Copyright	© ECMS2008
Printed: ISBN: 0-9553018-7-4	2008 International Conference
978-0-9553018-7-2	on High Performance Computing
	& Simulation (HPCS 2008)
CD: ISBN: 0-9553018-6-6	*joint CD – ECMS 2008 & HPCS 2008*
978-0-9553018-6-5	
Cover picture provided by	Loucas S. Louca
Printed by	Digitaldruck Pirrot GmbH 66125 Sbr.-Dudweiler, Germany

PROCEEDINGS OF THE 2008 INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING & SIMULATION (HPCS 2008)

June 3rd – 6th, 2008
Nicosia, Cyprus

Edited by: Waleed W. Smari

Organized by: The European Council for Modelling and Simulation [ECMS]

Technically Co-sponsored by:

Co-Sponsored by IEEE Germany, ASIM, EUROSIM, CASS, JSST, LSS, PTSK, TSS, [IEEE] Institute of Electrical and Electronics Engineers UKRI, The University of Cyprus

Hosted by:

University of Cyprus, Nicosia, Cyprus

The 2008 International Conference on High Performance Computing & Simulation (HPCS 2008)

June 3 - 6, 2008
Nicosia, Cyprus

Co-Sponsored by IEEE Germany, ASIM, EUROSIM, CASS, JSST, LSS, PTSK, TSS, The University of Cyprus

INTERNATIONAL TECHNICAL PROGRAM COMMITTEE - HPCS 2008

WORKSHOP ON SECURITY AND HIGH PERFORMANCE COMPUTING SYSTEMS (SHPCS 08)

SHPCS 08 Technical Program Committee:

Martin G. Jaatun, SINTEF ICT, Norway
Jean-Francois Lalande, LIFO, Université d'Orleans, France
Pil Joong Lee, Chungnam National University, South Korea
Mark Manulis, Université Catholique de Louvain, Belgium
Jose Antonio Onieva González, Universidad de Málaga, Spain
Alexander Pretschner, ETH Zürich, Switzerland
Waleed W. Smari, University of Dayton, USA
Willy Susilo, University of Wollongong, Australia
Toshihiro Tabata, Okayama University, Japan
Simone Tacconi, Polizia di Stato, Italy
Carolyn Talcott, SRI International, USA
Soon Tee Teoh, University of California, Davis, USA
Christian Toinard, LIFO, Universite d'Orleans, France
Luis Javier Garcia Villalba, Universidad Complutense de Madrid, Spain
Zonghua Zhang, INRIA, IRCICA, Villeneuve d'Ascq, France

WORKSHOP ON OPTIMIZATION ISSUES IN GRID AND PARALLEL COMPUTING ENVIRONMENTS (OPTIM 08)

Organizers: Pascal Bouvry, Girma Berhe, Bernabé Dorronsoro, Sébastien Varrette
University of Luxembourg, Luxembourg

OPTIM 08 Technical Program Committee:
Enrique Alba, University of Malaga, Spain
Farhad Arbab, CWI, Amsterdam, The Netherlands
Jacek Blazewicz, Technical University of Poznan, Poland
Azzedine Boukerche, University of Ottawa, Canada
Serge Chaumette, University of Bordeaux, France
Frédéric Guinand, University of Le Havre, France
Luc Hogie, INRIA Sophia-Antipolis, France
Djamel Khadraoui, CRP-Tudor, Luxembourg
Ricky Kwok, Colorado State University, Fort Collins, USA
Kittichai Lavangnananda, KMUTT, School of Information Technology, Bangkok, Thailand
An Le Thi Hoai, University of Metz, France
Anthony A. Maciejewski, Colorado State University, Fort Collins, USA
Carlos H.C. Ribeiro, Instituto Tecnológico de Aeronáutica, São José dos Campos – SP, Brazil
Jean-Louis Roch, INRIA Rhônes-Alpes, France
Steffen Rothkugel, University of Luxembourg, Luxembourg
Franciszek Seredynski, Polish Academy of Sciences, Warsaw, Poland
H.J. Siegel, Colorado State University, Fort Collins, USA
Peter Sturm, University of Trier, Germany
Michel Syska, I3S, University of Nice Sophia Antipolis, CNRS and INRIA, France
El-Ghazali Talbi, INRIA-Futurs, Lille, France
Nikos Vlassis, Technical University of Crete, Greece
Albert Y. Zomaya, University of Sydney, Australia

SPECIAL SESSION ON WEB SERVICES AND THE SEMANTIC GRID (WSSG 2008)

Organizer: Taha Osman
School of Computing & Informatics
Nottingham Trent University, Nottingham, U.K.

WSSG 08 Technical Program Committee:
David Al-Dabass, Nottingham Trent University, UK
Christophe Claramount, French Naval Academy, France
Jeff Pan, University of Aberdeen, UK
Evtim Peytchev, Nottingham Trent University, UK
Omer Rana, Cardiff University, UK
S. Reiff-Marganiec, Leicester University, UK
Gerald Schaefer, Aston University, UK
Vlad Tanasescu, Knowledge Media Institute (KMi), Open University, UK

SPECIAL SESSION ON PATTERN ANALYSIS AND RECOGNITION (PAR 08)

Organizer: Ventzeslav Valev, IAPR Fellow
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences, Sofia, Bulgaria

PAR 08 Technical Program Committee:
Mayer Aladjem, Ben-Gurion University of the Negev, Israel
Asai Asaithambi, University of South Dakota, USA
Jan-Olof Eklundh, Royal Institute of Technology, Sweden
Tin Kam Ho, Bell Laboratories, USA
Joachim Hornegger, Friedrich-Alexander University Erlangen-Nuremberg, Germany
Marek Kurzynski, Wroclaw University of Technology, Poland
Mariofanna Milanova, University of Arkansas at Little Rock, USA
Petia Radeva, Autonomous University of Barcelona, Spain
Bulent Sankur, Bogazici University, Turkey
Jose Ruiz Shulcloper, Advanced Technologies Applications Center, Cuba

HPCS 2008 ADDITIONAL REVIEWERS:

Mike McMahon, University of Nevada, Reno, Nevada, USA

HPCS 2008 PREFACE

Welcome to Cyprus, welcome to Nicosia, and welcome to HPCS 2008. This is the 6th edition of the HPCS conference since 2003. As it has matured into an esteemed venue for publication and discussion of knowledge in the corresponding areas, we are very happy and honoured to serve as General Co-Chairs of HPCS 2008. The conference program covers a wide spectrum of issues in modelling and simulation of high performance and large scale computing systems that play a key role in science and industry today.

Very hearty thanks go to the Program Chair, Professor Waleed W. Smari, and his colleagues for putting together such a broad, well designed and interesting program. A big thanks also goes to those who have set up all the related tutorials, special sessions, workshops and panels. We would also like to thank the HPCS 2008 local organizers from the University of Cyprus as well as ECMS.

We sincerely hope that you will enjoy the presentations and discussions; use the opportunity to meet interesting researchers, and appreciate the wonderful setting that the city of Nicosia offers to this conference. Last but not least, welcome back to HPCS 2009 next year.

Vladimir Getov	Gaetan Hains	Mads Nygård
London, United Kingdom	Paris, France	Trondheim, Norway

April 2008

HPCS 2008 PROGRAM

On behalf of the organizers and International Program Committee, I would like to welcome you to the 2008 International Conference on High Performance Computing and Simulation (HPCS 2008) held in Nicosia, Cyprus, June 3-6, 2008, in conjunction with ECMS 2008. This conference will provide a dynamic forum to address, explore, and exchange information, knowledge, and experiences in the state-of-the-art in high performance computing systems, their modelling and simulation, design and use, and impact. HPCS brings together researchers, scientists, engineers, practitioners, educators, and students from many nations and backgrounds to exchange their insights, breakthroughs, and research results about aspects of these systems and their technologies; to discuss challenges encountered in government, industry, and academe; and to seek new and innovative solutions. Additionally, we hope that the conference will present opportunities for many open technical interchanges in individual and group settings on key technology issues, during the conference and the potential for future collaborations among the participants, afterwards.

Current research in university, industry and research laboratories provides a new generation of HPC systems that rely on grid concepts, reconfigurability, autonomicity and pervasiveness to create fully interconnected communities of interest and practice with decision quality information in compressed time cycles. Through modelling and simulation, knowledge sharing and discovery, and just-in-time global grid-based information processing, individuals and groups will work together and make better, not just faster, decisions, services and products. It is hoped that the technologies and research presented in HPCS meetings will address the foundations and developments upon which these next generation systems will be built.

The conference this year consists of three keynote speeches, one plenary speech, three tutorials, one panel session, one posters session, 13 technical sessions, six breaks, three luncheons, and social events. For the first time, HPCS 2008 has its own proceedings volume. It contains 42 out of a total of 75 papers submitted, with an acceptance rate of 56%. Each paper in the main track was assigned to 4-5 reviewers and the majority of authors received at least 3-4 reviews back. Due to the TPC members' timely response, we were able to meet the deadlines we had planned for the track. Each of the workshops and special sessions handled their papers separately but maintained similar standards for paper evaluation and acceptance as much as possible. The program's technical papers represent works from academia, research laboratories, government, and industry.

On behalf of the Organizing and Program Committees, I would like to thank the many people who helped make this conference successful. I thank all authors who submitted their work to HPCS 2008 and who are presenting in Nicosia. Our excellent collections of papers and presentations were possible through the diligent work of the International Technical Program Committee. The ITPC members and reviewers did an exceptional job and we are grateful for their help in reviewing and evaluating the paper submissions. We would like to acknowledge the multi conference's three Keynote Speakers Profs. Geoffrey C. Fox, Felix Breitenecker, and Raymond Marie as well as our Plenary Speaker Prof. Wolfgang Gentzsch. For the first time, the conference this year has two workshops and two special sessions that were organized by Profs. Luca Spalazzi, Ratan Guha, Pascal Bouvry, Ventzeslav Valev, Drs. Girma Berhe, Bernabé Dorronsoro, Sébastien Varrette, and Taha Osman. We are thankful for their efforts and contributions. We keenly urge all participants to organize workshops and special sessions in their area of interest in future meetings and thus grow the community.

Also for the first time, the conference has three tutorials presented by Drs. Sandro Fiore, Salvatore Vadacca, Marcos Athanasoulis and Florence Reinisch and Prof. Geoffrey C. Fox. We thank our panellists Profs. Abachi, Bouvry, Getov, Fox, and Trinitis for their contributions to the HPCS 2008 Panel Session. This year, and with the outstanding efforts of the Posters Co-Chairs, we put together a good set of poster papers and presentations that we hope will contribute to some useful exchange and live discussions.

We wish to thank the European Council for Modelling and Simulation members for their hard work, support, and advice, which made the conference a success. We also wish to thank our hosts at the University of Cyprus in Nicosia, Cyprus for the wonderful arrangements, support, and services they have provided. We acknowledge all our co-sponsors, including the Biomedical High Performance Computing Leadership Summit and Harvard Medical School, Boston, Massachusetts, USA. And last but not least, we thank Ms. Martina-Maria Seidel, the HPCS 2008 Conference Manager for her continual support throughout the year to make this conference possible in every way.

I must also express my gratitude for the support, guidance, and encouragement I received from our General Chairs this year. In addition, I wish to thank all members of the Advisory Committee and the Organizing Committee without whom this conference and program would not have been possible.

We thank all of our attendees for making HPCS 2008 an extraordinary and enjoyable event. We hope you find this year's conference stimulating and worthwhile and look forward to seeing you at HPCS 2009.

Waleed W. Smari
HPCS 2008 Program Chair
Dayton, Ohio, USA
April 2008

TABLE OF CONTENTS

The 2008 International Conference on High Performance Computing & Simulation (HPCS 2008)
June 3 - 6, 2008, Nicosia, Cyprus

HPCS 2008 KEYNOTE SPEECH

Multicore Grids and the Data Deluge

Geoffrey C. Fox
Professor of Computer Science, Informatics, Physics
Pervasive Technology Laboratories
Indiana University, Indiana, USA

ABSTRACT

Technology advances suggest that the data deluge, network bandwidth and computers performance will continue their exponential increase. Computers will exhibit 64-128 cores in some 5 years. Consequences include a growing importance of data mining and data analysis capabilities that need to perform well on both parallel and distributed Grid systems. Parallelism needs to be extended from cluster to multicore architectures. Grids need to inherit the simplicity and broad support of Web 2.0 including mash-ups, gadgets and clouds. Clouds are virtual clusters forming a Grid that exports a system not a service interface. We look at possible scientific computing execution and programming environments that build on commodity Web 2.0 and multicore software concepts. Perhaps these will get good commercial support and finally allow attractive parallel and Grid software environments.

SPEAKER BIOGRAPHY

Geoffrey C. Fox (gcf@indiana.edu, http://www.infomall.org).
Professor Fox received a Ph.D. in Theoretical Physics from Cambridge University and is now professor of Computer Science, Informatics, and Physics at Indiana University. He is director of the Community Grids Laboratory of the Pervasive Technology Laboratories at Indiana University. He previously held positions at Caltech, Syracuse University and Florida State University. He has published over 550 papers in physics and computer science and been a major author on four books. Professor Fox has worked in a variety of applied computer science fields with his work on computational physics evolving into contributions to parallel computing and now to Grid and multicore chip systems. He has worked on the computing issues in several application areas – currently focusing on Defense, Earthquake and Ice-sheet Science and Chemical Informatics. He is involved in several projects to enhance the capabilities of Minority Serving Institutions.

The 2008 International Conference on High performance Computing & Simulation (HPCS 2008)
June 3 - 6, 2008, Nicosia, Cyprus

HPCS 2008 PLENARY SPEECH

Porting HPC Applications on Grids

Wolfgang Gentzsch
Duke University, RENCI Renaissance Computing Institute at UNC Chapel Hill, USA and the D-Grid Initiative, Germany

ABSTRACT

This presentation addresses the state-of-the-art in applications running on distributed, high performance and large scale computing systems, their modeling and simulation, design and use, and their impact. We will focus on compute and data intensive applications which have been ported to Grids recently. The presentation will include lessons learned and recommendations, and tries to answer specific questions like: Why do we need Grids and for which applications? Especially which applications benefit from Grids ? What are some of the success stories so far? What are the benefits of Grids compared with monolithic HPC systems? How to port and/or develop applications for Grids? The presentation will include some thoughts about current trends and future directions.

SPEAKER BIOGRAPHY

Wolfgang Gentzsch is adjunct professor of computer science at Duke University in Durham, and visiting scientist at RENCI Renaissance Computing Institute at UNC Chapel Hill, both in North Carolina. He is also a consultant for the German D-Grid Initiative. Recently, he was Vice Chair of the e-Infrastructure Reflection Group; Area Director of Major Grid Projects of the Open Grid Forum Steering Group; and he is a member of the US President's Council of Advisors for Science and Technology (PCAST-NIT). Before, he was Managing Director of MCNC Grid and Data Center Services in North Carolina; Sun's Senior Director of Grid Computing in Menlo Park, CA; President, CEO, and CTO of start-up companies Genias and Gridware, and professor of mathematics and computer science at the University of Applied Sciences in Regensburg, Germany. Wolfgang Gentzsch studied mathematics and physics at the Technical Universities in Aachen and Darmstadt, Germany.

HPCS 2008 TUTORIALS

TUTORIAL I

Grid Database Access, Management and Integration

Sandro Fiore and Salvatore Vadacca
Euro-Mediterranean Centre for Climate Change (CMCC) and
SPACI Consortium
Lecce, Italy

TUTORIAL DESCRIPTION

Grids encourage and promote the publication, sharing and integration of scientific data, distributed across Virtual Organizations. Scientists and researchers (from bioinformatics, astrophysics, etc.) work on huge, complex and growing datasets. The complexity of data management within a grid environment comes from the distribution, heterogeneity and number of data sources. Along with coarse-grained services (such as grid storages, replica services and storage resource managers), there is a strong interest on fine-grained services concerning, for instance, grid-database access and management. This tutorial will explain in detail Grid-Database Management Systems, with topics including basics on DBMS & Grids, database virtualization, data access and integration, security issues, performance issues, and interoperability with existing middleware (Globus, gLite, etc.). We present and discuss the state of major projects in the area, with focus on emerging and consolidated grid standards and specifications as well as production grid middleware. A demo on the Grid Relational Catalog (GRelC) Project will show real scenarios and use cases related to data access and integration. Examples concern bioinformatics (Italian LIBI Project), climate changes (Euro-Mediterranean Centre for Climate Change Data Grid CMCC-DataGrid), virtual clinical folders, accounting, monitoring and others. Both relational and XML databases are refered to.

TUTORIAL OUTLINE

- Basics on Database Management Systems & Grids
- Database virtualization and Grid-DB concept
- Existing and novel approaches for data access and integration
- Security issues, e.g., ACL, VO Membership management systems

- Performance issues, e.g. advanced delivery mechanisms/protocols in grid, streaming, compression
- Interoperability with existing middleware (Globus, gLite, etc.)
- Scalability issues
- Data Grid Portals and short demo on the GRelC Portal.

TARGET AUDIENCE

The targeted audience includes people interested in concepts related to database access, management and integration (both relational and XML) and grid environments (both gLite and Globus-based). Participants may, for instance, have background in bioinformatics (molecules/protein DBs), astrophysics (astronomic DBs), or climate research (metadata DBs for Earth Science, CMCC scenario).

REQUIRED BACKGROUND

Basics on Database Management Systems and query languages (SQL for RDBMS and XPath for XML DBs).

TUTORIAL DURATION

Two hours:

- Hour 1 – Basic concepts on Grid data management systems (S. Vadacca)
- Hour 2 – Advanced concepts on Grid data management systems, state of the art and future roadmap (S. Fiore)

INSTRUCTORS BIOGRAPHIES

Sandro Fiore was born in Galatina (ITALY) in 1976. He received a summa cum laude Laurea degree in Computer Engineering from the University of Lecce (Italy) in 2001, as well as a PhD degree in Informatic Engineering on Innovative Materials and Technologies from the ISUFI-University of Lecce in 2004. Research activities focus on parallel and distributed computing, specifically on advanced grid data management. Since 2004, he is a member of the Center for Advanced Computational Technologies (CACT) of the University of Salento and technical staff member of the SPACI Consortium. Since 2001 he has beens the Project Principal Investigator of the Grid Relational Catalog project (http://grelc.unile.it). Dr. Fiore was involved in the EGEE project (Enabling Grids for E-science) and is currently involved in the EGEE-II project and other national projects (LIBI). Since June 2006, he leads the Data Grid group of the Euro-Mediterranean Centre for Climate Change (CMCC) in Lecce (Italy). He is author and co-author of more than 40 papers in refereed journals/proceedings on parallel & grid computing and holds a patent on advanced data management.

Salvatore Vadacca was born in Galatina (LE) in 1982. He received summa cum laude bachelor and master degrees in Computer Engineering from the University of Lecce, Italy in 2003 and 2006, respectively. His research interests include data management; distributed, peer-to-peer and grid computing; as well as web design and development. Since 2003, he has been a team member of the GRelC Project. In 2006 he joined the Euro-Mediterranean Centre for Climate Change (CMCC) in Lecce, Italy, where he works in the Data Grid group.

The 2008 International Conference on High Performance Computing and Simulation (HPCS'08)
June 3-6, 2008, Nicosia, Cyprus

TUTORIAL II

Building Shared High Performance Computing Infrastructure for the Biomedical Sciences: Learnings from Biomed HPC 2007

Marcos Athanasoulis Dr.PH and Florence Reinisch MPH

Harvard Medical School
Boston, Massachusetts
USA

DESCRIPTION

In recent years high performance computing has moved from the sidelines to the mainstream of biomedical research. Increasingly researchers are employing computational methods to facilitate their wet lab research. Some emerging laboratories and approaches are based on a 100% computational framework. While there are many lessons to be learned from the computational infrastructure put into place for the physical and mechanical sciences, the character, nature and demands of biomedical computing differ from the needs of the other sciences. Biomedical computational problems, for example, tend to be less computationally intensive but more "bursty" in their needs. This creates both an opportunity (it is easier to meet capacity needs) and a challenge (job scheduling rules are more complicated to accommodate the bursts).

Harvard Medical School provides one of the most advanced shared high performance research computing centers at an academic medical center. In 2007, Harvard convened the first Biomedical High Performance Computing Leadership Summit to explore the issues in creating shared computing infrastructure for the biomedical sciences. We brought together over 100 leaders in the field to exchange ideas and approaches. Through special sessions and direct participant surveys a number of themes emerged around best practices in deploying shared computational infrastructure for the biomedical sciences. Based on prior experience and the summit findings, this workshop summarizes the approaches and ideas to providing a technical and process blueprint for organizations wishing to provide shared research computing research resources for groups small or large – from a few hundred CPUs and terabytes of data to thousands of CPUs and a petabyte or more of data.

TUTORIAL OUTLINE

The workshop includes the following topics:

- Summary of the current problems in Biomedical Sciences HPC
  - Image Processing
  - Simulation
  - 'Omics
  - Translational Research

- Data Centers and Hardware
    - Solving density problems
    - Power and cooling strategies
    - Blade servers and the multi-core machine
- Deployment Architectures
    - Approaches to system imaging
    - Supporting fault tolerant applications
    - Distributed storage, ready for production use?
    - Proprietary interconnects – cost/benefit analysis
    - Virtualization
- Job Scheduling
    - Approaches to time and resource based queues
    - Handling the challenges of parallel vs. distributed
    - Integrating "contributed" hardware
- Managing Storage Growth
    - SAN vs NAS
    - Distributed file systems
    - Archiving and near-line storage
    - New approaches to compression and de-duplication
- Organizational Challenges
    - How to ask for and get seed funding
    - Measuring performance and Return on Investment
    - Affiliating for group purchasing power
    - Workflow and support models
    - Working with the ego of the PI
    - Setting limits of services
- Putting it into Action
    - Online and offline resources
    - Communities and colleagues
    - Deployment planning tools

TARGET AUDIENCE

The target audience includes any researcher and or research IT core service provider who are interested in the challenges of providing shared high performance computing infrastructure to the biomedical sciences. From the postdoc who needs to set-up a modest compute cluster for their laboratory to the senior researcher who has been charged with providing world class infrastructure this tutorial will make them aware of the foundations and latest challenges of biomedical HPC.

REQUIRED BACKGROUND

While it is expected that tutorial attendees should be information technology professionals with a basic background in systems deployment and computer sciences, the session should prove valuable to anyone with an interest in the challenges and opportunities in creating high performance computing infrastructure for the biomedical sciences.

DURATION

The tutorial will take two hours. The bulk of the session will be devoted to the technical challenges and current issues in the field. In the final part of the session, participants will have the opportunity to present their plan for their home institution to the tutorial participants.

INSTRUCTORS BIOGRAPHIES

Dr. Athanasoulis is the chair of the Biomedical High Performance Computing Leadership Summit and Director of Client Services and Research Information Technology for Harvard Medical School where he oversees the IT service operations for the school and leads the development of high performance computing infrastructure to support biomedical and healthcare research. During his career, Dr. Athanasoulis has worked in both the public and private sector to improve the quality and efficiency of healthcare and research through information systems. Prior to joining Harvard Medical School, Dr. Athanasoulis was the Vice President of Product Development at RelayHealth Corporation, Inc., where he oversaw the continuing development and implementation of an advanced patient-provider communication system. As Chief Technology Officer at HealthCentral.com, he led the development of health information systems for more than 100 hospitals and health plans as well as a consumer portal that served millions of consumers. Dr. Athanasoulis has consulted to a wide variety of health care organizations including, UC San Diego, the Koop Foundation, the California Department of Health Services, San Francisco General Hospital, Alta Bates Hospital, the National Community Pharmacists Association and the UC Berkeley Wellness Guide. He is also the chief technical advisor for Healia, Inc. and co-founder of the Healthy Communities Foundation. He holds a master's degree in epidemiology and biostatistics and a doctorate in health informatics, both from UC Berkeley where he was a University Fellow.

Ms. Reinisch, is the Program Director for the Biomedical High Performance Computing Leadership Summit. She has more than twelve years of experience designing information systems in the biomedical sector. As program officer for the Healthy Communities Foundation she leads the implementation of a web based indicators project deployed in California and Washing State. She has served as co-investigator on multiple NIOSH – funded projects, including a four year study to evaluate the 1994 CDC Guidelines for the control of nosocomial transmission of tuberculosis risk among health care workers. Ms Reinisch was previously Director for the California Sharps Injury Prevention Program, where she developed a dynamic web application for the program, and served as co-investigator in a three-year CDC grant. She has expertise conducting surveillance and epidemiologic studies in occupational and environmental health, designing data management systems, statistical analysis, project management and web application deployment. She holds a Masters degree in Epidemiology and Biostatistics from University of California, Berkeley.

The 2008 International Conference on High Performance Computing and Simulation (HPCS'08)
June 3-6, 2008, Nicosia, Cyprus

TUTORIAL III

Using Multicore Chips for Scientific Computing

Geoffrey C. Fox
Professor of Computer Science, Informatics, Physics
Pervasive Technology Laboratories
Indiana University Indiana, USA

TUTORIAL DESCRIPTION

In this tutorial we review the status of generally available multicore chips including mainline chips from AMD and Intel as well graphics units from IBM (Cell) and NVIDIA. We discuss programming models and relation of these to those familiar on traditional distributed and shared memory parallel machines. We look at performance issues emphasizing those like memory bandwidth and shared cache usage that are distinct from those that dominate traditional large scale parallel applications. We discuss relation of multicore, cluster and Grid computing and examine role of services in unifying them. Examples from the SALSA project http://www.infomall.org/salsa will be used to illustrate ideas. The application focus will be linear algebra and data mining but other areas such as solution of differential equations will be discussed.

TUTORIAL OUTLINE

- Introduction to Multicore Chips
- Programming Models
- Relation to Other High Performance Distributed and Parallel Machines
- Performance of Multicore Systems
- Relation to Cluster and Grid computing
- Multicore Projects and Applications
- Conclusions on State of the Art and Future Directions

TARGET AUDIENCE

The target audience includes researchers, students, and practitioners who are interested in learning more about multicore design, development and use.

REQUIRED BACKGROUND

Knowledge of computer architecture and organization fundamentals. Knowledge of scientific computing.

TUTORIAL DURATION

The tutorial material will be presented in a 2 to 3-hour session.

INSTRUCTOR BIOGRAPHY

Geoffrey C. Fox (gcf@indiana.edu, http://www.infomall.org).
Professor Fox received a Ph.D. in Theoretical Physics from Cambridge University **a**nd is now professor of Computer Science, Informatics, and Physics at Indiana University. He is director of the Community Grids Laboratory of the Pervasive Technology Laboratories at Indiana University. He previously held positions at Caltech, Syracuse University and Florida State University. He has published over 550 papers in physics and computer science and been a major author on four books. Professor Fox has worked in a variety of applied computer science fields with his work on computational physics evolving into contributions to parallel computing and now to Grid and multicore chip systems. He has worked on the computing issues in several application areas – currently focusing on Defense, Earthquake and Ice-sheet Science and Chemical Informatics. He is involved in several projects to enhance the capabilities of Minority Serving Institutions.

The 2008 International Conference on High Performance Computing and Simulation (HPCS 2008)
June 3-6, 2008, Nicosia, Cyprus

HPCS 2008 PANEL SESSION

Challenges Facing Modeling and Simulation in HPC Environments

Moderator:

***Dr. Waleed W. Smari*, University of Dayton, Ohio, USA**

PANEL MEMBERS:

Dr. Geoffrey Fox,	Indiana University, USA
Dr. Vladimir Getov,	University of Westminster, Westminster, U.K.
Dr. Hamid Abachi,	Monash University, Australia
Dr. Pascal Bouvry,	Luxembourg University, Luxembourg
Dr. Carsten Trinitis,	UniversityTechnische Universität München, Germany

ABSTRACT:

In the past years, high performance computing environments have been used to solve challenging scientific modeling and simulation applications. Now high performance computing (HPC) technologies are becoming increasingly important as target platforms and have matured to the point of becoming a tool for impacting society at large in many major ways.

Panel members will present and discuss the main challenges facing HPC and simulation communities today and in the foreseen future. They will emphasize and possibly prioritize the importance of these problems and the difficulties of approaching them.

Furthermore, based on their experiences with different modeling and simulation applications using HPC technologies (such as cluster and grid platforms), our panelists will point out the main trends and new technologies that will impact these challenges, and the factors that they believe will play major roles in pushing these challenges forward.

PANELISTS SHORT BIOS:

Geoffrey C. Fox, distinguished scientific and director of Community Grids Lab. Pervasive Technology Labs at Indiana University. He earned his Ph.D. in Cambridge University (Theoretical Physics) in 1967. He nationally renowned for his work in the development and application of parallel computers, most recently served as director of the Computational and Information Science Laboratory at Florida State University. He was also the director of the Northeast Parallel Architectures Center at Syracuse University from 1990-2000. His research has led to two commercial spinoffs-

WebWisdom.com and Anabas Inc. Fox has worked with many distinguished scientists in his career; while at CalTech he worked with Nobel Prize winner in physics, Richard Feynman. Fox's current projects include developing the Online Knowledge Center for the Department of Defense High Performance Computing and Modernization Program, which is creating a peer-to-peer system to allow users to more easily access and update information on the department's high performance computers.

Vladimir Getov graduated from the Technical University of Sofia (Bulgaria) with distinction in 1975 and then earned his Ph.D. in Computer Science from the Bulgarian Academy of Sciences in 1980. As a Senior Research Fellow he spent several years leading both R&D and academic research projects in Bulgaria. During that time Dr Getov was Project Manager of the first Bulgarian IBM PC/XT compatible computer (1984). In 1989 he moved to England where he joined the Concurrent Computations Group at the University of Southampton. Since 1994 Dr Getov has been an academic staff member at the University of Westminster in London where he was awarded the titles Reader in 1998 and Professor in 2001. In 2003 Vladimir Getov was appointed Research Director of the Harrow School of Computer Science. In 2006 he was awarded Doctor of Science degree from the Bulgarian Academy of Sciences for research work and results in the area of "Methods and Systems for High Performance Computing with Java". Professor Vladimir Getov has been leading the Distributed and Intelligent Systems Group at the University of Westminster in London since 1996. He was a founding member of the Java Grande Forum in 1998 and has led the Java Grande Message Passing Group which produced the Message Passing For Java (MPJ) specification. Vladimir Getov was also co-chair of the Service Management Frameworks Group of the Open Grid Forum. In 2004, Professor Getov was elected Governor of the International Council for Computer Communication. He is a member of the CoreGrid Executive Committee and Leader of the European Institute on "Grid Systems, Tools, and Environments" of the EU CoreGrid Network of Excellence (Sept. 2004 - Aug. 2008). Professor Getov is also a Steering Committee Member of the John Vincent Atanasoff Initiative, working actively towards worldwide recognition of the inventor of electronic digital computing.

Carsten Trinitis received his Master in Electrical Engineering from Technische Universität München, Germany in 1990. He also received a Bachelor in Environmental Engineering in 1992, from Technische Universität München,. He earned his PhD in Electrical Engineering, Technische Universität München in 1998. Title of thesis: “Electric field optimization of three dimensional problems in high voltage enineering”. He also carried out Postdoctoral studies: Lehrstuhl für Rechnertechnik und Rechnerorganisation (Prof. Dr. A. Bode), Computer Science Department, TU München, in 1998, and from 2001 to 2005. Since 2005 he is a Senior Scientist at Lehrstuhl für Rechnertechnik und Rechnerorganisation TU München. Since 2002 he is also an Assistant Professor at the German Armed Forces University, Munich, Germany. Other activities include: (1998) Systems Engineer, MCG Company, Baldham, Germany, (1998 – 2001) Development Engineer, FORCE Computers GmbH, Neubiberg, Germany, (1992) Visiting Scientist, Worcester Polytechnic Institute, Worcester/MA, USA, (1994) Visiting Scientist, Jadavpur University, Calcutta, India, and **(**2005) Visiting Professor, University of Lomza, Poland. His research interests comprise computer architectures, microprocessor architectures, performance analysis of compute intensive applications, and history of computer science.

Hamid Abachi received his Ph.D. degree in Electrical and Computer Systems Engineering from University of Wales in Britain in 1981. He has been in academic life for more than 25 years. Hamid has also worked and gained a wide spectrum of practical experiences in heavy to light industries. From 1991 to present he has held a faculty position in the Department of Electrical and Computer Systems Engineering, at Monash University in Australia. He is Director of the International Program, and Director of Postgraduate (Coursework) studies as well as the Professional Development Programs. He is a member of the Editorial Board of the IEEE Systems Journal in the USA and WSEAS Transactions on Computer Research. He has been a keynote speaker at many international conferences. In addition, Hamid is also a member of Technical Program Committees and a reviewer of more than 60 international conferences where in a number of occasions he has been invited to serve as the conference chair. He is a Fellow of IET (formally IEE, The Institution of Electrical Engineers, UK) and a Fellow of IEAust (Engineers Australia). Hamid is also a Senior Member of the IEEE, USA. His prime research areas include the modeling and simulation of Parallel Processing Systems, Design of Advanced Computer Architectures, Fault-tolerant Distribution and Parallel Systems. He has many journal and international conference papers in these areas.

Pascal Bouvry earned his undergraduate degree in Economical & Social Sciences and his Master degree in Computer Science with distinction ('91) from the University of Namur, Belgium. He went on to obtain his Ph.D. degree ('94) in Computer Science with great distinction at the University of Grenoble (INPG), France. His research at the IMAG laboratory focused on mapping and scheduling task graphs onto Distributed Memory Parallel Computers. Next, he performed post-doctoral research on coordination languages and multi-agent evolutionary computing at CWI in Amsterdam, the Netherlands. Dr Bouvry gained industrial experience as manager of the technology consultant team for FICS (NASDAQ: SONE) a world leader in electronic financial services. Next, he worked as CEO and CTO of SDC, a Saigon-based joint venture between SPT (a major telecom operator in Vietnam), Spacebel SA (a Belgian leader in Space, GIS and Healthcare), and IOIT, a public research and training center. After that, Dr. Bouvry moved to Montreal as VP Production of Lat45 and Development Director for MetaSolv Software (NASDAQ: ORCL), a world-leader in Operation Support Systems for the telecom industry (e.g. AT&T, Worldcom, Bell Canada, etc.). Dr. Bouvry is currently heading the Computer Science and Communications (CSC) research unit of the Faculty of Sciences, Technology and Communications of Luxembourg University, and serving as Professor. Pascal Bouvry is also a member of the administration board of CRP-Tudor and a member of various scientific committees and technical workgroups (ERCIM WG, COST TIST, LIASIT, etc.).

The 2008 International Conference on High Performance Computing and Simulation (HPCS 2008)
June 3-6, 2008, Nicosia, Cyprus

HPCS 2008 POSTER ABSTRACTS

(Partial List)

Massively Parallel Simulations Of Astrophysical Plasmas

Bruno Thooris, E. Audit, A. S. Brun, Y. Fidaali, F. Masset, D. Pomarède, R. Teyssier
CEA/IRFU Saclay, Gif/Yvette Cedex FRANCE

ABSTRACT

The COAST (for Computational Astrophysics) project is a program of massively parallel numerical simulations in astrophysics involving astrophysicists and software engineers from CEA/IRFU Saclay. The scientific objective is the understanding of the formation of structures in the Universe, including the study of largescale cosmological structures and galaxy formation, turbulence in interstellar medium, stellar magnetohydrodynamics and protoplanetary systems. The simulations of astrophysical plasmas are performed on massively parallel mainframes (MareNostrum Barcelona, CCRT CEA France), using 3-D magnetohydrodynamics and N-body parallelized codes developed locally. We present in this paper an overview of the software codes and tools developed and some results of such simulations. We also describe the Saclay SD*vision* graphical interface, implemented in the framework of IDL Object graphics, our 3-D visualization tool for analysis of the computation results.

Application Of Novel Technique In Ripemd-160 Hash Function Aiming At High-Throughput

Harris Michail, V. Thanasoulis, D. Scinianakis, G. Panagiotakopoulos and C. Goutis
University Of Patras, Achaia, GREECE

ABSTRACT

Hash functions, form a special family of cryptographic algorithms that address the requirements for security, confidentiality and validity for several applications in technology. Many applications like PKI, IPSec, DSA, MAC's need the requirements mentioned before. All the previous applications incorporate hash functions and address, as time passes, to more and more users-clients and thus the increase of their throughput is necessary. In this paper we propose an implementation that increases throughput and operating frequency significantly and at the same time keeps the area small enough for the hash function RIPEMD-160. The deployed technique involves the application of spatial and temporal pre-computation to the conventional operation block. The proposed implementation leads to an implementation that achieves 35% higher throughput.

Presenter's Biography:

Harris Michail received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of computer security, hardware design and reconfigurable architectures.

Vassilis Thansoulis is an under-graduate student in the Department of Electrical .Eng, University of Patras, Greece. He is currently working on his thesis that lies in the domain of security.

Dimitris Schinianakis received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of of computer security, hardware design

George Panagiotakopoulos received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of embedded computers.

Costas Goutis (S'70-M'78) received B.Sc in Physics, Diploma in Electronic Engineering, M.Sc from the University of Heriott-Watt and Ph.D from the University of Southampton. He is currently a Professor with the ECE Department, University of Patras.

Communication Cost of Matrix Product on Super-Hypercube Architecture

Maryam Amiripour, Hamid Abachi
Department of ECSE, Monash University, AUSTRALIA

ABSTRACT

Processor allocation and the task scheduling technique in parallel processing systems play a significant role in improving the performance of a message-passing architecture. Adapting the right algorithms and further improvements in areas such as time complexity, execution time, speed up and synchronization mechanisms undoubtedly facilitates implementation of advanced applications on a parallel processing system. These applications include but are not limited to DNA computing, artificial immune systems and optical computing to name a few. This paper highlights the communication cost related to a Super-Hypercube topology for being a subclass of traditional Hypercube architecture. Furthermore, a particular reference is made to the mathematical modeling of Hypercube and Super-Hypercube architectures. Finally, graphical presentations are carried out based on mathematical calculations to address the advantage of Super-Hypercube topology.

Presenter's Biography:

Maryam Amiripour received her B.A. in Mathematics from Al-Zahra University in Iran in 1999. That was followed by a Post Graduate Diploma in Information and System Management form Queensland University in Australia in 2001.She is currently pursuing her PhD degree in Department of Electrical and Computer Systems Engineering at Monash University in Australia. Her area of research includes hardware design, modeling and simulation of advance parallel processing systems. The main parameters of her investigation include evaluation of performance, reliability, speed and cost analysis of massively parallel processing systems. She has a number of referred journal and conference papers in these areas. Her e-mail address is: maryam.amiripour@eng.monash.edu.au.

Hamid Abachi received his Ph.D. degree in Computer Engineering from University College Cardiff in Wales, Britain, in 1981. He has twenty five years of teaching, research and administrative experiences in international universities around the world. He is currently an Associate Professor in the Department of Electrical and Computer Systems Engineering at Monash University in Australia. He has more than 95 referred international publications including Journal and conference papers. He has served as a member of international program committee to more than 72 international conferences around the world. On a number of occasions has acted as the conference chairman and on many occasions as the session chairman at international conferences. He has also participated in the plenary sessions at international conferences. He has been a corecipient of the John Madsen Medal for his best Journal paper in the discipline of Electrical Engineering form the Institution of Engineers Australia (IEAust) in 2002, plus receiving a number of best paper awards in international conferences. In addition he has been invited as a keynote speaker at four international conferences. He is a Fellow of The Institution of Engineering and Technology (the IET, formerly IEE) in Britain, a Fellow of the Institution of Engineers in Australia (IEAust) and a Senior Member of IEEE in the USA. His research interests include design and simulation of parallel processing systems, modeling of advanced computer architectures, application of distributed multimedia computing in advanced Engineering Education. His e-mail address is: hamid.abachi@eng.monash.edu.au.

A Service Based Architecture for a Sensors Data Integrating System to Support Emergency Management Using Grid based Infrastructure

Enrico Tosti, Waleed Smari
Universita' Politecnica delle Marche, Ancona, ITALY & University of Dayton, Ohio, USA

ABSTRACT

Information is a critical aspect in any emergency management activity. Decision making and efficiency are improved when these are based on complete sets of information about the conditions of the struck area. Sensors integration in a grid-based architecture through a Service Oriented solution allows Real-time access to information via the World Wide Web. Thanks to the computational power of the grid architecture, the workload concerning the storing, analyzing, correlating and mining the large amount of sensed raw data can be split allowing a quick notification when a potential threat is detected. In this poster, we survey and classify types of sensors used in EMS and propose a grid based architecture that will support such systems.

Presenter's Biography:

Enrico Tosti is a Computer Engineering student at the Università Politecnica delle Marche, Ancona, Italy, currently performing research activities at the High Performance and Reconfigurable Computing Laboratory in the Department of Electrical and Computer Engineering, the University of Dayton, OH. This work is in part the subject of his Masters degree Thesis in Computer Engineering.

Waleed W. Smari received his Ph.D. degree in Electrical and Computer Engineering from Syracuse University, Syracuse, New York. Currently, he is an Associate Professor at the Department of Electrical and Computer Engineering, University of Dayton, Dayton, OH. His technical interests and specialties include Collaboration Sciences and Technologies, Human Centered Computing, High Performance Parallel and Distributed Processing and Networking, Performance Evaluation Methods and Modeling Techniques of Computing Systems, Information Systems and Engineering, Reconfigurable Computing and Digital Systems Design, and Computer Engineering Education. He has been active in research and scholarship in his areas of expertise, both at the national and international levels. He has served as PI on several research projects sponsored by government agencies and industry. He has edited several volumes and authored/co-authored over 60 publications. He is the Associate Editor of five international journals. He chaired international conferences and worked in others in various leadership capacities, such as program chair, exhibits chair, workshops chair, local arrangements chair. He was a Visiting Fellow at government labs as well as at a number of universities around the World. Dr. Smari has long teaching experience, both at the graduate and undergraduate university levels. Dr. Smari is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), a Member of the Association for Computing Machinery (ACM), the American Society for Engineering Education (ASEE), the International Society of Computers and Their Applications (ISCA), the International Association of Science and Technology for Development (IASTED), The Society for Computer Simulation International (SCS), The European Council on Modelling and Simulation (ECMS), and the VHDL International Users Forum (MASIC-VIUF).

GRelC DAIS: towards Data Access and P2P Integration in gLite-based Production Grids

Sandro Fiore, Salvatore Vadacca
University of Salento & Euro-Mediterranean Centre for Climate Change (CMCC), ITALY

ABSTRACT

In this poster, we describe in detail the Grid Relational Catalog (GRelC) Project, an integrated environment for grid database management, highlighting the vision/approach, architecture, components, services and technological issues. The key topic of this poster is the GRelC Data Access and Integration Service. The GRelC DAIS is a GSI/VOMS enabled web service addressing extreme performance, interoperability and security. It efficiently, securely and transparently manage databases on the grid across VOs, with regard to emerging and consolidated grid standards and specifications as well as production grid middleware (gLite & Globus). It provides a uniform access interface, in grid, both to access and integrate relational (Mysql, Oracle, Postgresql, IBM/DB2, SQLite) and non-relational data sources (XML DB engines such as eXist, XIndice and libxml2 based documents). Today the GRelC DAIS is part of the GILDA release (EGEE t-Infrastructure) and is candidate at the EGEE Respect Program, is part of the glite-based INFNGRID release since is tightly coupled with the gLite middleware, the EGEE architecture and the EGEE Training Infrastructure. Currently the GRelC DAIS is used as the Euro-Mediterranean Centre for Climate Change (CMCC) Data Grid framework.

Presenter's Biography:

Sandro Fiore was born in Galatina (LE) in 1976. He received a summa cum laude Laurea degree in Computer Engineering from the University of Lecce (Italy) in 2001, as well as a PhD degree in Informatic Engineering on Innovative Materials and Technologies from the ISUFI-University of Lecce in 2004. Research activities focus on parallel and distributed computing, specifically on advanced grid data management. Since 2004, he is a member of the Center for Advanced Computational Technologies (CACT) of the University of Salento and technical staff member of the SPACI Consortium. Since 2001 he has been the Project Principal Investigator of the Grid Relational Catalog project. Dr. Fiore was involved in the EGEE project (Enabling Grids for E-science) and is currently involved in the EGEE-II project and other national projects (LIBI). Since June 2006, he leads the Data Grid group of the Euro-Mediterranean Centre for Climate Change (CMCC) in Lecce (Italy). He is author and co-author of more than 40 papers in refereed journals/proceedings on parallel and grid computing and holds a patent on advanced data management.

Salvatore Vadacca was born in Galatina (LE) in 1982. He received summa cum laude bachelor and master degrees in Computer Engineering from the University of Lecce, Italy in 2003 and 2006, respectively. His research interests include data management; distributed, peer-to-peer and grid computing; as well as web design and development. Since 2003, he has been a team member of the GRelC Project. In 2006 he joined the Euro-Mediterranean Centre for Climate Change (CMCC) in Lecce, Italy, where he works in the Data Grid group.

SOPAS: A Low-Cost and Secure Solution for E-Commerce

M. Pasquet, D. Vacquez, C. Rosenberger
Laboratoire GREYC: ENSICAEN & Université de CAEN – CNRS, FRANCE

ABSTRACT

We present in this poster a new architecture for remote banking and e-commerce applications. The proposed solution is designed to be low cost and provides some good guarantees of security for a client and his bank issuer. Indeed, the main problem for an issuer is to identify and authenticate one client (a cardholder) using his personal computer through the web when this client wants to access to remote banking services or when he wants to pay on a e-commerce site equipped with 3D-secure payment solution. The proposed solution described in this paper is MasterCard Chip Authentication Program compliant and was experimented in the project called SOPAS. The main contribution of this system consists in the use of a smartcard with a I^2C bus that pilots a terminal only equipped with a screen and a keyboard. During the use of services, the user types his PIN code on the keyboard and all the security part of the transaction is performed by the chip of the smartcard. None information of security stays on the personal computer and a dynamic token created by the card is sent to the bank and verified by the front end. We present first the defined methodology and we analyze the main security aspects of the proposed solution.

Presenter's Biography:

Marc PASQUET is an assistant professor at ENSICAEN, France. He obtained his Master degree from ENSAM (Ecole Nationale Supérieure des Arts et Métiers) in 1977. He worked for 13 years for different companies belonging to the signal transmission field and 15 years for the banking sector in the field of electronic payment. He joined ENSICAEN (National Engineer School of Caen in France) in 2006 where he is now leading researchs in the field of electronic payment.

Delphine Vacquez is a Research & development engineer at ENSICAEN, France. She obtained her Master degree in 2006 from the University of Caen (computer science department). She has been working in the field of secure and contacless e-payment.

Christophe Rosenberger is a Full Professor at ENSICAEN, France. He obtained his Ph.D. degree from the University of Rennes I in 1999. He works at the GREYC Laboratory. His research interests include evaluation of image processing and biometrics. He is involved recently on e-transactions applications.

A Proposal for Securing A Large-scale High-Interaction Honeypot

Jérémy Briffaut, Jean-François Lalande, Christian Toinard
University of Orléans, Security and Distributed Systems (SDS) Team, FRANCE

ABSTRACT

This poster presents the design of a secured high-interaction honeypot. The challenge is to have a honeypot that welcomes attackers, allows userland malicious activities but prevents from system corruption. The honeypot must be scalable to authorize a large amount of malicious activities and to analyze those activities efficiently. The hardening of the honeypot is proposed for two kinds of host. The first class prevents system corruption and has never to be reinstalled. The second class assumes system corruptions but easy reinstallation is available. A first cluster enables to deploy a wide range of honeypots and security sensors. A second cluster provides an efficient auditing facility. The solution is totally based on open source software and has been validated during one year. A statistical analysis shows the efficiency of the different sensors. Origin and destination of attacks are given.

Presenter's Biography:

Jeremy Briffaut has just finished his Phd thesis in december 2007. One of the contribution of his thesis is the PIGA Intrusion Detection System that is used as one of the sensors deployed in the High-Interaction Honeypot. He is currently a temporary researcher at Ensi de Bourges.

Jean-François Lalande is an Assistant Professor in the SDS team. He then made his PhD at INRIA of Sophia-Antipolis, in the Mascotte project (CNRS/INRIA/UNSA). His research interests were the combinatory optimization inside optical and satellite networks. He is currently working on the verification of security policies and intrusion detection using policy graph analysis.

Christian Toinard is a Professsor in the Laboratoire d'Informatique de l'Université d'Orléans (LIFO).
He received a Degree in Engineering from INSA in 1985 and a Degree in Engineering from ENSIMAG in 1986. After having worked in software industry, he made a PhD at the University of Paris VI in 1992. He is currently inc harge to charge to set up and develop research activities in security. The research developed had lead to the creation of the Security and Distributed System team.

HPCS 2008 SPONSORS

Co-sponsored by:

- IEEE Germany Section
- ASIM; German Speaking Simulation Society
- EUROSIM; Federation of European Simulation Societies (ASIM, AES, DBSS, CROSSIM, CSSS, FRANCOSIM, HSS, ISCS, SIMS, SLOSIM, UKSIM)
- CASS; The Chinese Association for System Simulation
- JSST; The Japanese Society for Simulation Technology
- LSS; The Latvian Simulation Society
- PTSK; The Polish Society of Computer
- TSS; The Turkish Simulation Society
- University of Cyprus, Cyprus

In Cooperation with the IEEE Computer Society Technical Committee on Parallel Processing (TCPP) (Pending)

HPCS 2008 EXHIBITORS

The 2008 International Conference on High Performance Computing and Simulation (HPCS 2008)
June 3-6, 2008, Nicosia, Cyprus

Technical Papers

Parallelization of Simulation and HPC Application

COMPARISON OF SPATIAL DATA STRUCTURES IN OPENMP–PARALLELIZED STEERING

Alexander Wirz, Claudia Leopold, Björn Knafla
University of Kassel, Research Group Programming Languages / Methodologies
Wilhelmshöher Allee 73, 34121 Kassel, Germany
wirz@student.uni-kassel.de, {leopold,bknafla}@uni-kassel.de

KEYWORDS

Parallelization of Simulation; Partitioning, Mapping, and Scheduling

ABSTRACT

With the invent of multicore CPUs, parallelization becomes an important issue in the programming of computer games. We consider steering, i.e., the coordination of a large number of agents to simulate swarm-like behavior. Steering and in particular its neighborhood search component are compute-intensive and profit from parallelization. In this paper, we combine OpenMP threads with the use of spatial data structures, considering three structures: k-d tree, grid, and cell array with binary search. In addition to parallelization, we tuned the performance by e.g. optimizing cache usage. The focus of the paper is an experimental comparison of the three structures, based on a parallel version of the OpenSteer library. In nearest-neighbor search and updates, we found none of the structures to be superior to the others, and provide a detailed account of their pros and cons.

1. INTRODUCTION

As parallel architectures are becoming commonplace, novel application areas arise. One of them is computer games in which, among others, artificial intelligence components can profit from faster hardware. In this paper, we consider steering, a technique that models the movement of computer-controlled characters (agents) as a combination of simple steering behaviors. Examples include obstacle avoidance and alignment to the movement direction of neighbors. From individual behaviors, a seemingly intelligent group behavior emerges. Steering is deployed in games such as Fable and Alarm for Cobra 11.

We carried out experiments with OpenSteer (Reynolds, 1999, 2004), an open-source C++ library that implements steering behaviors, and its associated testbed OpenSteerDemo that supports rapid prototyping of combinations of the behaviors. Two scenarios were considered: 1) a group of pedestrians following a path and thereby avoiding collisions, and 2) a swarm of birds moving according to rules of separation, alignment and cohesion. OpenSteerDemo has been parallelized with OpenMP threads in previous work (Knafla and Leopold, 2007), and we refer to this parallel version.

As found by profiling, the clearly most compute-intensive component of steering is neighborhood search. Therein, each agent computes the c nearest from among those agents who are within a certain radius around the agent. A straightforward algorithm considers all pairs of agents and requires time $O(n^2)$ for n agents.

Nearest-neighbor search can be speeded up by using spatial data structures, which are well-known from areas such as database management systems and computer graphics, where they are used for various types of geometric queries. In general, these structures organize data according to their coordinates, and thus store neighbors closer to each other. We use the structures to store agent data, which allows us to restrict nearest-neighbor search to only a subset of agents.

Several types of spatial data structures have been suggested in the literature and can be classified into flat and hierachical structures (Velho et al., 2002). This paper investigates k-d tree, grid, and cell array with binary search, of which the k-d tree is a hierarchical and the others are flat data structures. We consider two variants of the grid, based on hashing and modulo functions, respectively. A potentially infinite world is assumed for all structures, i.e., coordinates are not bounded.

Our application runs a sequence of simulation steps. In each step, it searches for the nearest neighbors of all agents, and updates the states of all agents. As agents move, states and in particular coordinates change dynamically. A specific requirement of our application is parallel nearest neighbor queries and parallel updates, i.e. multiple agents issue queries at the same time. Data structures that support these operations have been implemented with C++. Performance was tuned to make each structure as efficient as possible. For instance, we grouped queries from close agents to improve cache usage.

Our main contribution is an experimental comparison of the performance of the three structures. Briefly stated, there was no clear winner. Although the flat data structures clearly outperformed the k-d tree, they require careful application-specific tuning. The k-d tree, in contrast, is more robust towards changes in the application setting. Of the flat structures, the cell array with binary search is on advantage if agent density is high, and inferior other-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

wise. Only of the two grid variants, the modulo grid was a clear winner.

Sect. 2 of this paper starts with background on steering, OpenSteer, and OpenSteerDemo, including OpenMP parallelization. Then, Sect. 3 introduces the three data structures. We explain the update and nearest-neighbor search operations, including parallel implementation. Moreover, Sect. 3 comprises an analytical comparison among the structures. Sect 4 is devoted to experiments, covering experimental setting, results, and discussion. Finally, Sect. 5 concludes the paper.

2. STEERING AND OPENSTEER

Steering is an artificial intelligence technique that is used in the implementation of computer games. It can be deployed to simulate swarms, flocks and other groups of agents, for which a complex group behavior emerges from simple individual behaviors of the group members. These individual behaviors are composed of elementary behaviors that may include (Schnellhammer and Feilkas, 2004):

- *separation*: keep some minimum distance from neighbors to avoid collisions,
- *cohesion*: stay near the center of neighbors to form a group, and
- *alignment*: adjust direction and velocity of movement with neighbors, to provide for a coordinated action.

Each elementary behavior is represented by a steering vector that is computed by vector operations from the agent's own state (position, orientation, velocity), the states of its neighbors, and possibly further information on the local environment (e.g. obstacles). The steering vectors of different behaviors are combined, e.g. by weighted summation, to compute the agent's new state.

OpenSteer and OpenSteerDemo can be used to develop and tune steering behaviors. In particular, OpenSteerDemo allows a user to write plugins for certain simulation scenarios. This paper refers to two scenarios that we adopted from Reynolds (Reynolds, 1999, 2004):

- *Pedestrian:* Here, agents (pedestrians) walk through a two-dimensional world, following the path in Fig. 1 from *a* to *g* and backwards, repeatedly. To avoid collisions, each agent observes its environment and computes steering vectors according to pathfollowing, agent- and obstacle-collision avoidance rules. Note that only nearest neighbors (as described below) are taken into account in these computations. Thus, steering avoids but does not preclude collisions, the remaining collisions are taken care of by other components of a game.
- *Boids:* This scenario assumes a three-dimensional world. Boids stands for birds-like objects or bird-oids, i.e., the plugin simulates a swarm of birds who move according to the separation, cohesion, and alignment rules described above. As in the pedestrian plugin, the computation of steering vectors is based on nearest neighbors only.

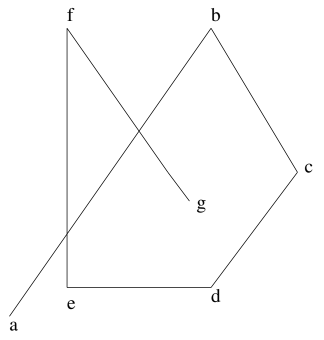


Figure 1: Path for pedestrian plugin

The term nearest neighbor needs further explanation. In both scenarios, it is assumed that an agent can only see its local environment, which is modeled by a fixed-size circle (for pedestrian), or a fixed-size sphere (for boids) around the agent, respectively. An agent must fulfill two requirements to be a nearest neighbor of another agent: 1) it must be situated within maximum distance d from that agent, and 2) it must belong to the c closest neighbors of that agent. Obviously, the nearest-neighbor relation is not symmetric.

OpenSteerDemo runs a main loop as it is typical for games. Each step of the iteration simulates one discrete time step, and is subdivided into an update stage and a graphics stage. The former logically moves all agents, and the latter renders the new state of the game world. We use the term frame rate to denote the number of main loop steps that can be carried out per second. Thus, frame rate is a measure for the speed of the application, and a high frame rate corresponds to a low running time.

In previous work by the same authors, the application has been refactored and parallelized (Knafla and Leopold, 2007). The parallel version deploys a user-configurable number of OpenMP threads, each of which simulates multiple agents. In the parallel version, the update stage has been split up into a simulation sub-stage and a modification sub-stage, which are separated by barriers. The simulation sub-stage starts with a nearest-neighbor search for all agents, and then computes and combines steering vectors based on the states of neighbors. During the modification sub-stage, agents update the world state by writing their new individual states to a shared data structure. Since the simulation sub-stage comprises read accesses only, and the writes of the modification sub-stage refer to disjoint data, each of the two stages can internally be run in parallel, without any need for synchronization.

In the base version of OpenSteerDemo (Knafla and Leopold, 2007), the running time of each step is dominated by the time for nearest-neighbor search. This version deploys a straightforward search algorithm that we denote as *brute-force* in the following. It assumes agents, or more specifically the data that represent the agents' states, to be stored in a list, and deploys a doubly-nested loop. The outer loop runs through all agents to invoke nearest-neighbor search, and the inner loop runs through all agents to select the c closest neighbors.

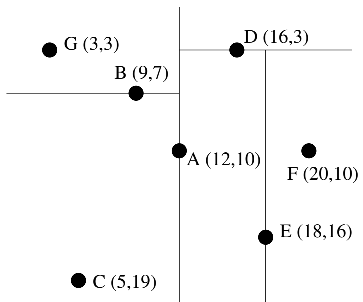
Obviously, in the inner loop, it would be sufficient to consider agents that are reasonably close. This can be achieved with spatial data structures that allow to identify close agents without touching the others. The following section describes three structures that we experimented with in this paper: k-d tree, grid, and cell array with binary search.

3. PARALLEL DATA STRUCTURES AND IMPLEMENTATION

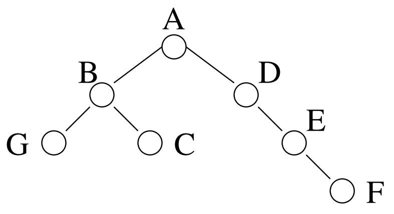
k-d tree

The k-d tree (Bentley, 1975) is a special type of binary search tree for data with k-dimensional keys. In our case, keys represent agent positions, and $k = 2$ (for pedestrian) or $k = 3$ (for boids). We refer to the homogeneous variant, in which data are held by both internal nodes and leaves. As in any search tree, a node's key is larger than the keys of all nodes in its left subtree, and less than or equal to the keys in its right subtree. The special feature of k-d trees is the definition of this less-than relation. Depending on node level, it discriminates by one of the k coordinates: first coordinate for the root, second coordinate for nodes at level one, and so on, cyclically.

Figure 2 gives an example for $k = 2$. Here, subtrees of `A` are discriminated by x-coordinate, subtrees of `B` and `D` by y-coordinate, and subtrees of `G`, `C` and `E`, if existent, by x-coordinate, again.



(a) Points in a two-dimensional space



(b) The corresponding k-d tree

Figure 2: Example for k-d tree

A k-d tree may be balanced or not, and hence its height may vary from O(log$n$) to O(n), where n denotes the number of nodes. Operations *search* and *insert* are straightforward: As in standard binary search trees, they start from the root and recursively branch into the left or right subtree, depending on a comparison between the given node's key and that of the root. Of course, the comparison must be based on the respective coordinate, and thus the level needs to be taken care of. As in standard search trees, a newly inserted node becomes a leaf.

For *delete*, let us denote the node to be deleted by p, and assume its position to be known from a previous search. If p is a leaf, delete is trivial. Otherwise, without loss of generality, let p discriminate its subtrees by x. Then, the contents of p is replaced by the contents of q, the node with smallest x-coordinate from among the nodes in p's right subtree (if there are multiple such nodes, any can be taken). In case of an empty right subtree, the left and right subtrees are exchanged before selecting q. After having replaced the contents of p by that of q, the algorithm deletes q by applying the same scheme recursively.

The algorithm for *nearest-neighbor search* takes as input a center point, which is the position of an agent, and the radius of the search region, which is a circle or sphere around it. Starting with the root, it checks for each node on its way through the tree whether it is inside the search region. If it is, the corresponding agent is a candidate for nearest neighbor, and the algorithm compares it to the others found so far to decide whether it is among the c closest. If the agent is outside the search region, the value of its discriminator coordinate is compared to the maximum and minimum values that this coordinate may take for the search region, and the search proceeds with either the left, right, or both subtrees, accordingly.

Parallel invocation of nearest-neighbor search for multiple agents does not require any changes to the data structure, as all accesses are reads. We allocate frequently used data structures such as lists of nearest neighbors once per thread instead of once per nearest neighbor search, to avoid expensive system calls from a parallel region.

After each simulation step, the positions of all agents need to be updated. Unfortunately, k-d trees are ill-suited for this operation since they have a static structure, i.e., movement requires rearrangement of the tree. Of course, any update can be realized by a delete followed by an insert, but these operations are expensive and nontrivial to parallelize (JáJá, 1992). Therefore, we decided to rebuild the tree after each simulation step, which has the additional advantage that the generated tree is balanced. Experiments showed this variant to be faster than inserts/deletes.

Grid

A grid subdivides the world into disjoint cells (Samet, 2006). We refer to equal-sized cells that correspond to rectangles and cuboids, respectively. Grid cells are indexed with k-dimensional vectors, where k denotes the

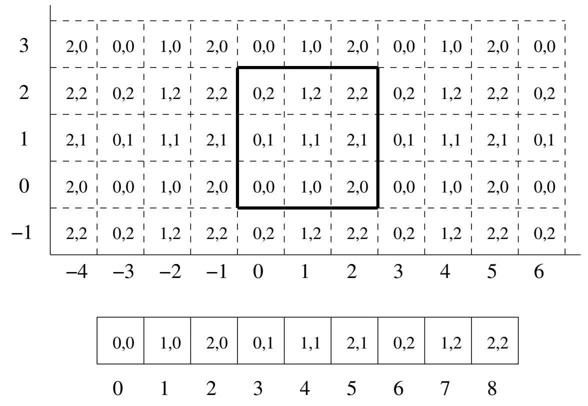


Figure 3: Assignment from grid cells to buckets with modulo function. The selected region is marked in bold. There is a one-to-one correspondence between cells of the region and buckets, as illustrated in the lower part of the figure.

dimensionality of the world. From an agent's coordinates, the index of the grid cell holding the agent is computed by integer divison through cell side lengths. If cells have extent $10 \times 10 \times 5$, for instance, an agent at coordinates $(23,7,17)$ belongs to grid cell $(2,0,3)$.

From the assumption of an unbounded world, there is an infinite number of grid cells. The grid data structure groups them into a finite number *m* of buckets and stores them in an array. Each element holds a list (implemented by an STL `vector`) of those agents that belong to a grid cell assigned to the bucket.

Note that the assignment of agents to buckets comprises two steps. First, agents are assigned to grid cells by dividing their coordinates by the cell side lengths, and then grid cells are assigned to buckets by another function. For the second step, we considered two options: hashing and modulo. In both cases, the bucket index is computed from an agent's coordinates within constant time. For hashing, we used function

$$\text{hash}(x,y,z) = (x \cdot p_1 \quad \text{xor} \quad y \cdot p_2 \quad \text{xor} \quad z \cdot p_3) \,\%\, m$$

where (x,y,z) is the index of the grid cell, and $p_1 = 73856093$, $p_2 = 19349663$ and $p_3 = 83492791$ are primes. The function has been taken from Teschner et al. (Teschner et al., 2003) to balance the number of grid cells per bucket.

For the modulo function, a core grid is spanned on a finite region of the world. Core cells have a one-to-one correspondence to buckets. Cells outside core are mapped to core by a modulo operation. Figure 3 illustrates the assignment. For example, grid cells $(1,2)$ and $(-2,-1)$ are mapped to bucket 7.

In both variants of the grid, *search* is realized by first computing the agent's bucket, and then searching this bucket linearly. *Insert* and *delete* start selecting the bucket, as well. Then, insert appends the agent to the corresponding list, whereas delete first locates the agent and then replaces it by the last entry.

Nearest-neighbor search selects the set of grid cells that intersect with the search region by simple geometric calculations, and runs through all buckets that contain at least one grid cell of interest. In both the hashing and modulo variants, the algorithm traverses each bucket only once if the search region overlaps with multiple cells of the same bucket. In the hashing variant, a bitset with one entry per bucket is used for bookkeeping. The modulo variant does not consider individual cells, but directly computes the buckets from agent positions.

For performance tuning, we modified the parallel version of OpenSteer, which has an outermost `for` loop that runs through all agents, splits this loop into equal-sized chunks, and maps each chunk to a thread. In the tuned version, the outermost loop runs through buckets, and an equal-sized chunk of buckets is assigned to each thread. Threads process buckets one after another. Since each bucket holds a group of grid cells, and agents from the same grid cell are close, nearest neighbor queries issued for agents from the same bucket have search regions with much overlap. Thus, when processing these queries, a similar set of buckets needs to be considered, which can likely be kept in cache. The improvement in locality comes at the price of worse load balancing, as buckets differ in their number of agents. Nevertheless, in experiments we observed a performance increase by about 35%.

Updates are simple writes of coordinates as long as an agent stays within the same cell. If it crosses cell boundaries, it must be deleted from the old cell, and inserted into the new one. As with nearest-neighbor search, each thread is responsible for multiple buckets. To deal with boundary crossings, it maintains a private list of agents moving away. At the end of the update phase, all private lists are processed sequentially. It would have been possible to parallelize this step as well, but not worthwhile as only 0.5 to 2 % of agents are affected.

Cell Array with Binary Search

Like a grid, the cell array with binary search subdivides the world into disjoint cells and assigns them to buckets (Mauch, 2003). Unlike a grid, it includes only $k-1$ dimensions into the partitioning, but uses the last coordinate to sort entries of each bucket.

Figure 4 depicts a two-dimensional world divided into cells of side lengths 10. As in the modulo variant of grids, cells of a selected region (here 0 to 30 on the x-axis) directly correspond to buckets. To assign an agent to a bucket, first the index of the cell holding the agent is calculated by dividing the agent's coordinates by the cell side lengths, and then the cell is mapped to the selected region by the modulo operation. Within each bucket, agents are sorted by their *k*th coordinate.

Search is accomplished by first selecting the bucket, and then carrying out a binary search. For *insert* and *delete*, data have to be moved to create or fill a gap, which is less efficient than in grids.

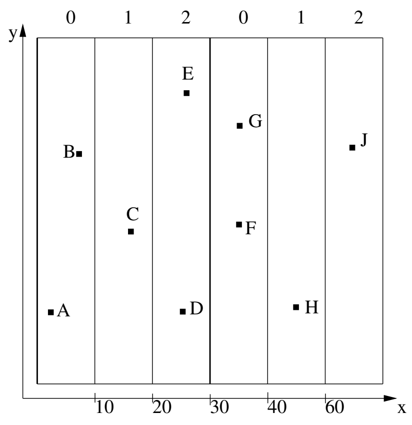


Figure 4: Cell array with binary search

For *nearest-neighbor search*, as in grids, the algorithm selects all buckets that intersect with the search region, but chooses buckets based on coordinates of only $k-1$ dimensions. When searching a particular bucket, it makes use of the sorted order. As illustrated in Fig. 5, the search region defines minimum and maximum coordinates in the kth dimension for agents of interest ($y_{\min}$ and $y_{\max}$). In each bucket considered, the algorithm carries out a binary search to locate $y_{\min}$, and then investigates consecutive agents until their kth coordinate exceeds $y_{\max}$.

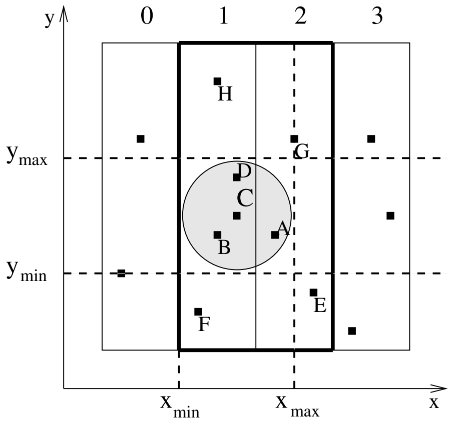


Figure 5: Nearest-neighbor search in a two-dimensional cell array.

Updates are realized as in grids, except that buckets are unsorted after modifications. Thus the grid algorithm is followed by a parallel sorting step in which each thread sorts multiple buckets. Despite our data being presorted, the STL sorting algorithm turned out to be faster than insertion sort, and was therefore deployed.

Analytical Comparison

Table 1 analytically compares the data structures, referring to the operations of interest in OpenSteerDemo. Formulas assume that the operations are applied to all agents. The table uses the following abbreviations:

- n: number of agents
- m: number of buckets
- f: average number of agents in the search region
- g: average number of grid cells in the search region
- h: average number of agents in grid cells that intersect with the search region

	Buildup	**Nearest-Neighbor**	**Update**
Brute-F.	$O(n)$	$O(n^2)$	$O(n^2)$
k-d Tree	$O(n \cdot \log n)$	$O(n(\log n + f))$	$O(n \cdot \log n)$
Grid	$O(m+n)$	$O(n(g+h))$	$O(n)$
Cell A.	$O(m+n^2)$	$O(n(g \cdot \log(n/m) + h))$	$O(n \cdot \log n)$

Table 1: Average running times of operations buildup, nearest-neighbor search, and update for different data structures

Formulas have been taken from Samet (Samet, 2006) and Mauch (Mauch, 2003). The table shows that all spatial data structures outperform the brute-force algorithm. Differences between the structures are minor. Note that part of the formulas can not be directly compared as, for instance, h differs between grid and cell array even for identical scenarios. The formulas refer to a sequential implementation. As we will see in Sect. 4, the running time of our application is dominated by nearest-neighbor search. For this operation, analytical speedup with p threads is linear for all structures, since threads process different agents independently.

4. EXPERIMENTS

Both the pedestrian and boids scenarios were run with 1000 agents, selecting $c = 7$ neighbors. We set the search radius to 12 units in the pedestrian plugin, and to 9 units in the boids plugin. In the boids scenario, we bounded the world by a sphere of radius 50 units, with a bird leaving the sphere immediately re-entering it at the diametrically opposite point. Agent positions were initialized by randomly placing pedestrians onto the path, and randomly placing birds in a sphere of radius 20 units, respectively.

We run experiments for 7200 simulation steps of OpenSteerDemo and report averages over three such runs. OpenSteerDemo allows to separately measure running times for different stages, and we made use of this feature. Experiments were carried out on a dual-processor dual-core AMD Opteron 270 machine with 2GB memory. OpenSteer / OpenSteerDemo was compiled by the Intel compiler version 10.0.23 with options `-std=c99 -O2 -inline-level=2 -openmp -openmp-report1 -fp-model fast -xW`.

	1 Thread	2 Threads	4 Threads
Brute-Force	66,94	34,54	18,39
k-d tree	59,22	31,39	16,30
Grid (Hashing)	41,38	20,97	11,94
Grid (Modulo)	28,48	14,75	7,67
cell array	24,10	13,32	6,77

Table 2: Running time of nearest-neighbor search for pedestrian plugin (in seconds).

	1 Thread	2 Threads	4 Threads
Brute-Force	40,60	21,72	11,71
k-d Tree	24,73	13,91	7,63
Grid (Hashing)	15,66	8,53	4,89
Grid (Modulo)	10,62	6,02	3,40
Cell Array	10,93	5,84	3,32

Table 3: Running time of nearest-neighbor search for boids plugin (in seconds).

Tables 2 and 3 list running times for nearest-neighbor queries. Note that values are for 7200 simulation steps with 1000 queries each.

The values in tables 2 and 3 refer to tuned variants of the flat data structures, in which we selected side lengths of cells and number of buckets as depicted in Table 4. Note that our data structures can handle an unbounded world, but in both scenarios the world is limited. Therefore, we decided for a one-to-one correspondence between buckets and grid cells for the modulo grid. For the hashing grid, such a one-to-one correspondence could not be achieved.

	Side length of cells		Number of buckets	
	Pedestrian	Boids	Pedestrian	Boids
Grid (H.)	15	18	80	500
Grid (M.)	15	15	343	343
Cell Array	10	10	100	100

Table 4: Selected parameters after performance tuning

A comparison of tables 2 and 3 reveals that nearest-neighbor search always takes longer in the pedestrian than in the boids scenario. This difference is partly due to the larger search radius, for which more nodes or buckets need to be investigated. It can even be observed for the brute-force algorithm, because of the need to select the *c* closest from among the neighbors in the search region. Moreover, the boids scenario more uniformly distributes agents in space, such that less birds than pedestrians are candidate for nearest neighbor.

Both tables clearly demonstrate that spatial data structures speed up nearest-neighbor search. In both cases,

- flat data structures outperform the k-d tree
- the modulo variant of the grid outperforms the hashing variant
- the cell array with binary search is slightly faster than the modulo grid.

Differences between the data structures are partly due to differences in the number of agents that need to be looked at during the search. This presumption was verified experimentally by counting agents (Wirz, 2008). One reason the hashing grid is inferior to the modulo grid is the need to search all grid cells assigned to a bucket, in contrast to the one-to-one correspondence in the modulo grid.

The number of agents to be looked at is about the same for k-d tree and modulo grid. Nevertheless, the modulo grid is faster, which is due to the realization of buckets by STL vectors, in contrast to the pointer-based structure of the k-d tree.

In the cell array, only about half the number of neighbors as in the modulo grid need to be looked at. The performance gain is less, though, since the cell array requires an additional operation: locating $y_{\min}$ by binary search. Whether or not the expense for this operation pays depends on the number of agents in a bucket.

As mentioned above, cell side lengths have been tuned for the flat structures, since we observed a major impact on running times. If side lengths are small, many cells and the corresponding buckets need to be considered; if cells are large, selected cells may hold many agents outside the search region. Thus, optimal values depend on both agent density and search radius. Experiments proved our hypothesis that a one-to-one correspondence between cells and buckets performs best in a bounded world.

Tables 2 and 3 show that parallelization of nearest-neighbor search was successful for all data structures, confirming the analysis in Sect. 3. Speedups range between 3.1 and 3.7 with 4 threads for the different data structures and scenarios.

Parallelization of the updates was less successful, as tables 5 and 6 show. We observed speedups of 1.3 to 2.2 with 4 threads for the flat data structures (Wirz, 2008), and slowdowns for the k-d tree. For the k-d tree, rebuilding the tree was about twice as fast as deleting and reinserting the nodes. As tables 5 and 6 show, the cell array performed best for the updates, as well. The hashing grid is slightly faster than the modulo grid, and the k-d tree is slowest.

The low speedups may be due to not having spent much time on tuning updates, since their impact on the overall running time is low. The fraction of running time spent in different phases of the application is illustrated in Fig. 6. It shows that spatial data structures significantly speed up the simulation phase (sum of search, update and simulation sections) to the point that further efforts are only worthwhile for the graphics phase. Results for the pedestrian scenario are similar and have been omitted for brevity.

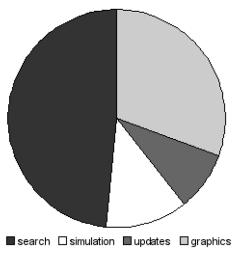
In our application, the speedups can be used to either increase the frame rate, or to handle more agents. Table 7 shows how many agents can be handled at a frame

	1 Thread	2 Threads	4 Threads
k-d Tree	3,19	4,40	4,52
Grid (Hashing)	1,51	1,06	0,70
Grid (Modulo)	1,42	1,47	1,09
Cell Array	1,13	0,96	0,69

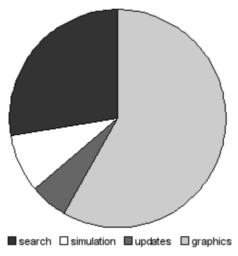
Table 5: Running time of update for pedestrian plugin (in seconds)

	1 Thread	2 Threads	4 Threads
k-d Tree	2,24	3,20	3,38
Grid (Hashing)	1,52	1,16	0,78
Grid (Modulo)	1,40	1,09	0,97
Cell Array	1,08	0,95	0,72

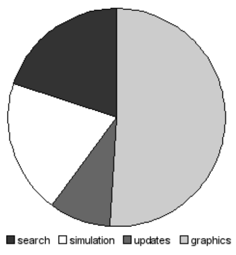
Table 6: Running time of update for boids plugin (in seconds)



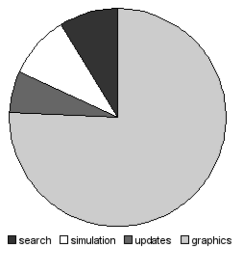
(a) Brute-Force (1 Thread)



(b) Brute-Force (4 Threads)



(c) Cell Array (1 Thread)



(d) Cell Array (4 Threads)

Figure 6: Fraction of running time spent in different phases for boids plugin

rate of 30 frames per second with the various data structures. Increases are significant and demonstrate the success of parallelization in combination with spatial data structures.

Finally comparing all data structures, the flat structures perform much better than the k-d tree, but have the drawback of requiring tuning. In particular, they can not deal with frequent changes in search radius. The k-d tree is robust towards such changes, and still performs better than the brute-force algorithm. Of the flat structures, the cell array appears somewhat faster than grids, but this outcome depends on scenario. The cell array is on advantage if cells are densely populated, but for sparsely populated cells grids may be the better choice. Of the two grid variants, the modulo grid was the clear winner.

	Brute-F.	k-d Tree	Grid (M.)	Cell A.
Ped.	2850	3100	4400	4700
Boids	3300	5000	7800	8200

Table 7: Maximum number of agents that can be simulated at 30 frames per second with 4 threads

5. CONCLUSIONS

The paper has shown that use of spatial data structures can significantly speed up steering in both sequential and parallel settings. Experiments were based on OpenSteer / OpenSteerDemo, and considered three structures: k-d tree, grid, and cell array with binary search. There was no clear winner, although a certain advantage for the cell array could be observed.

Future work should consider more data structures, especially more hierarchical structures such as range trees (Samet, 2006), other scenarios, and data structures for a bounded world.

REFERENCES

Bentley, J. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

JáJá, J. (1992). *An Introduction to Parallel Algorithms*. Addison-Wesley.

Knafla, B. and Leopold, C. (2007). Parallelizing a Real-Time Steering Simulation for Computer Games with OpenMP. In *Proc. Parallel Computing (ParCo)*, pages 219–226.

Mauch, S. (2003). *Efficent Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology.

Reynolds, C. W. (1999). Steering Behaviors For Autonomous Characters. In *Proc. Game Developer Conference*, pages 763–782.

Reynolds, C. W. (2004). OpenSteer Website. `http://opensteer.sourceforge.net`.

Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann.

Schnellhammer, C. and Feilkas, T. (2004). Steering behaviors. `http://www.steeringbehaviors.de`.

Teschner, M. et al. (2003). Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization*, pages 47–54.

Velho, L., Gomes, J., and Figueiredo, L. H. (2002). *Implicit Objects in Computer Graphics*. Springer.

Wirz, A. (2008). Parallele Räumliche 3D Datenstrukturen für Nachbarschafts-Abfragen. Master's thesis, Universität Kassel.

TRANSACTION-ORIENTED SIMULATION IN AD HOC GRIDS: DESIGN AND EXPERIENCE

Gerald Krafft and Vladimir Getov
Harrow School of Computer Science
University of Westminster
Watford Rd, Northwick Park, Harrow HA1 3TP, U.K.
E-mail: G.Krafft@gmx.net, V.S.Getov@wmin.ac.uk

KEYWORDS
Parallelization of simulation, grid and cluster computing, reliable parallel and distributed algorithms.

ABSTRACT

In this paper we analyse the requirements of performing parallel transaction-oriented simulations within loosely coupled systems like ad hoc grids. We focus especially on the space-parallel approach to parallel simulation and on discrete event synchronisation algorithms that are suitable for transaction-oriented simulation and the target environment of ad hoc grids. To demonstrate our findings, a Java-based parallel simulator for the transaction-oriented language GPSS/H is implemented on the basis of the most promising shock-resistant Time Warp (SRTW) synchronisation algorithm and using the grid framework ProActive. The analysis of our parallel simulator shows that the SRTW algorithm can successfully reduce the number of rolled back transaction moves but it also reveals circumstances in which the SRTW algorithm can be outperformed by the normal Time Warp algorithm. Finally, possible improvements to the SRTW algorithm, based on experiments using the Grid'5000 platform, are proposed in order to avoid such problems.

1. INTRODUCTION

Transaction-oriented simulation is a special case of discrete event simulation that uses two types of objects, stationary objects called blocks and mobile objects called transactions that move through the model and change the state of the blocks. The movement of a transaction can be delayed or blocked by a stationary object but otherwise transactions always move at a certain simulation time meaning that the simulation time does not progress while a transaction is moved. Because such a movement of a transaction describes an action with a specific timestamp that changes the state of the model it is equivalent to an event in discrete event simulation. Inferred from this equivalency most aspects and findings for discrete event simulation can also be applied to transaction-oriented simulation and vice versa. Transaction-oriented simulation is best suited for the simulation of systems that are based on units of traffic competing for the use of specific resources which covers a wide area of applications. Typical examples include communication systems, transportation systems, manufacturing systems and general queuing systems. The best-known transaction-orineted simulation language is GPSS and its extended version GPSS/H (Schriber 1991).

Parallel and distributed computing offers a way to reduce the runtime of large and complex computer simulations as for instance required in engineering, military, biology and climate research. But the growing complexity of simulation models can reach the limits of today's high-performance parallel computer systems, that in addition also induce a very significant cost factor. Grid platforms can provide large-scale computing at lower costs by allowing several organisations to share their resources. But traditional grid infrastructures are relatively static environments that require a dedicated administrative authority and therefore are not well suited for transient short-term collaborations of small organisations with fewer resources. Ad hoc grids enable the design of such dynamic and transient resource-sharing infrastructures (Smith et al. 2004) that allow even small organisations or individual users to form grid environments on demand. They make grid computing and grid resources widely available to small organisations and mainstream users allowing them to perform resource demanding computing tasks like complex computer simulations.

There are several approaches to performing simulations distributed across a parallel computer system. The space-parallel approach (Fujimoto 1993) is one of these approaches that is robust, applicable to many different simulation types and can be used to speed up single simulation runs. It requires for the simulation model to be partitioned into relatively independent sub-systems that are assigned to logical processes (LPs). Each LP may then be processed on a different node. Synchronisation between these LPs is still required because the model sub-systems are usually not fully independent. A lot of past research has concentrated on different synchronisation algorithms for parallel simulation. Some of these are only suitable for certain types of parallel computers such as shared memory systems.

This work investigates the possibility of performing parallel transaction-oriented simulation in ad hoc grid environments with the main focus on the aspects of parallel simulation. Potential synchronisation algorithms

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

and other simulation aspects are analysed in respect of their suitability for transaction-oriented simulation and ad hoc grids as the target environment and the chosen solutions are described and reasons for their choice given. To demonstrate the solutions a Java-based parallel simulator for the transaction-oriented language GPSS/H is implemented and evaluated using a set of simple example simulation models.

2. SYNCHRONISATION ALGORITHMS

2.1. Requirements

The choice of synchronisation algorithm can have a significant influence on how much of the parallelism that exists in a simulation model will be utilised by the parallel simulation system. An overview of the two main groups of synchronization algorithms described as conservative and optimistic alogirthms can be found in (Das 1996). Conservative algorithms utilise the parallelism less well than optimistic algorithms because they require guarantees, which in most cases are derived from additional knowledge about the behaviour of the simulation model, like for instance the communication topology or lookahead attributes of the model. For this reason conservative algorithms are often used to simulate very specific systems where such knowledge is given or can easily be derived from the model. For general purpose simulation systems optimistic algorithms are better suited as they can utilise the parallelism within a model to a higher degree without requiring any guarantees or additional knowledge. The best-known optimistic algorithm is Time Warp (TW) (Jefferson 1985). But in many situations TW can show an over-optimistic behaviour leading to uncommitted simulation progress being undone as a result of rollbacks. Subsequent research has therefore focused on limiting the optimism in TW if required and in a self adapting way.

Another important aspect of choosing the right synchronisation algorithm is the relation between the performance properties of the expected parallel hardware architecture and the granularity of the parallel algorithm. In order for the parallel algorithm to perform well in general on the target hardware environment the granularity of the algorithm, i.e. the ratio between computation and communication has to fit the ratio of the computation performance and communication performance provided by the parallel hardware.

Considering the target environment of ad hoc grids and the goal of designing and implementing a general parallel simulation system based on the transaction-oriented simulation language GPSS/H we concluded that the best suitable synchronisation algorithm is an optimistic or hybrid algorithm that has a coarse grained granularity. The algorithm should require only little communication compared to the volume of computation it performs. At the same time the algorithm needs to be flexible enough to adapt to a changing environment, as this is the case in ad hoc grids. A further requirement is that the algorithm can be adapted to and is suitable for transaction-oriented simulation.

2.2. Algorithm Selection

A promising algorithm found for these requirements is the SRTW algorithm (Ferscha and Johnson 1999). This algorithm has some similarities with the elastic time algorithm (Srinivasan and Reynolds 1995) and also the adaptive memory management algorithm (Das and Fujimoto 1997) but at the same time is suitable for loosely coupled distributed systems like grids. Similar to the elastic time algorithm state vectors are used to describe the current states of all LPs plus a set of functions to determine the output vector but the SRTW algorithm does not require a global state. Instead each LP separately tries to optimise its parameters towards the best performance. Similar to the adaptive memory management algorithm the optimism is controlled indirectly be setting artificial memory limits but for the SRTW algorithm each LP will limit its own memory instead of using an overall memory limit for the whole simulator.

The SRTW algorithm is described by (Ferscha and Johnson 1999) as a fully distributed approach to controlling the optimism in TW that requires no additional communication between the LPs. It is based on the TW algorithm but extends each LP with a control component called logical process control component (LPCC) that constantly collects information about the current state of the LP using a set of sensors. These sets of sensor values are then translated into sets of indicator values representing state vectors for the LP. The LPCC will keep a history of such state vectors using a clustering technique so that it can search for past state vectors that are similar to the current state vector but provide a better performance indicator. An actuator value will be derived from the most similar of such state vectors that is subsequently used to control the optimism of the LP.

As the SRTW algorithm was designed for discrete event simulation its sensors and indicators had to be adapted to the equivalent values in transaction-oriented simulation.

3. SIMULATOR DESIGN ISSUES

3.1. End of Simulation

Another important aspect that had to be considered is the detection and correct handling of the simulation end. A transaction-oriented simulation is complete when the defined end state is reached, i.e. the termination counter reaches a value less or equal to zero. When using an optimistic synchronisation algorithm for the parallelisation of transaction-oriented simulation it is crucial to consider that optimistic algorithms will first execute all local events without guarantee that the causal order is correct. They will recover from wrong states by performing a rollback if it later turns out that the causal order was violated. Therefore, any local state reached by an optimistic LP has to be considered provisional until a global virtual time (GVT) message has been received that guarantees the state. In addition, it is most likely that at any point in real time each of the LPs has reached a different local simulation time so that after an end state has been reached by one of the LPs, which is guaranteed by a GVT, it is important to synchronise the states of all LPs. Thus, the combined end state from all model partitions is equivalent to the model end state that would have been reached in a sequential simulator.

The mechanism suggested here leads to a consistent and correct global end state of the simulation considering the problems mentioned above. For this mechanism the LP reaching a provisional end state is switched into the provisional end mode. In this mode the LP will stop to process any further transactions leaving the local model partition in the same state but it will still respond to and process control messages like GVT parameter requests and it will receive transactions from other LPs that might cause a rollback. The LP will stay in this provisional end mode until the end of the simulation is confirmed by a GVT or a received transaction causes a rollback with a potential re-execution that is not resulting in the same end state. While the LP is in the provisional end mode additional GVT parameters are passed on for every GVT calculation denoting the fact that a provisional end state has been reached and the simulation time and priority of the transaction that caused the provisional end. The GVT calculation process can then assess whether the earliest current provisional end state is guaranteed by the GVT. If this is the case then all other LPs are forced to synchronise to the correct end state by rolling back using the simulation time and priority of the transaction that caused the provisional end and the simulation is stopped.

3.2. Suitable Cancellation Technique

Transaction-oriented simulation has some specific properties compared to discrete event simulation. One of these properties is that transactions do not consume simulation time while they are moving from block to block. This has an influence on which of the synchronisation algorithms are suitable for transaction-oriented simulation but also on the cancellation techniques used. If a transaction moves from LP1 to LP2 then it will arrive at LP2 with the same simulation time that it had at LP1. A transaction moving from one LP to another is therefore equivalent to an event in discrete event simulation that when executed creates another event for the other LP with exactly the same timestamp. Because simulation models can contain loops, as it is for instance common for models of quality control systems where an item failing the quality control needs to loop back through the production process, this specific behaviour of transaction-oriented simulation can lead to endless rollback loops for certain cancellation techniques. Besides the original cancellation technique introduced by (Jefferson 1985) that is known as aggressive cancellation another cancellation technique called lazy cancellation was suggested by (Gafni 1985). Figure 1 compares the rollback behaviour of aggressive cancellation and lazy cancellation in respect of such a loop within the simulation model.

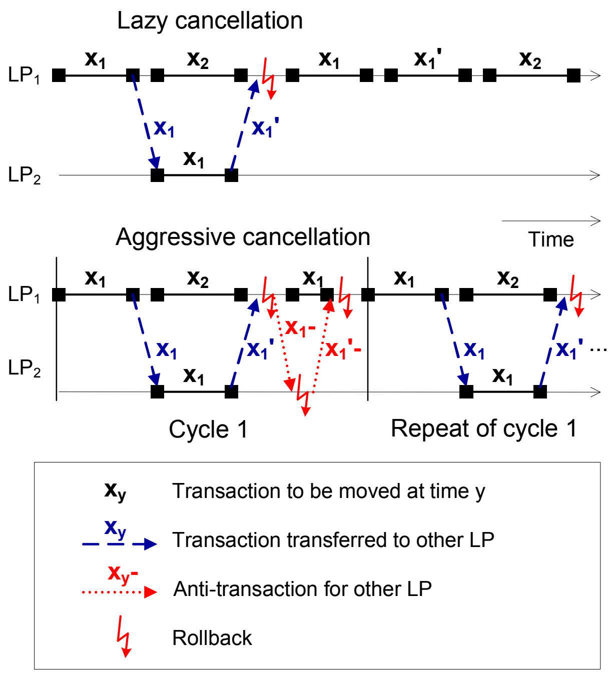


Figure 1: Cancellation in Transaction-Oriented Simulation

It shows the movement of transaction x1 from LP1 to LP2 but without a delay in simulation time the transaction is transferred back to LP1. As a result LP1 will be rolled back to the simulation time just before x1 was moved. At this point two copies of transaction x1 will exist in LP1. The first one is x1 itself which needs to be moved again and the second is x1' which is the copy that was send back from LP2. This is the point from where the execution differs between lazy cancellation and aggressive cancellation. In lazy cancellation x1 is processed again resulting in the same transfer to LP2. But because x1 was sent to LP1 already it will not be transferred again and no anti-transaction will be sent. From here LP1 just proceeds moving the transactions in its transaction chain according to their simulation time (transaction priorities are ignored for this example). Apposed to that in aggressive cancellation the rollback results in an anti-transaction being sent out for x1 immediately which causes a second rollback in LP2 and another anti-transaction for x1' being sent back to LP1. At the end both LPs will end up in the same state in which they were before x1 was processed by LP1. The same cycle of events would start again without any actual simulation progress.

In order to avoid the described endless rollback loops lazy cancellation needs to be used for parallel transaction-oriented simulation.

4. IMPLEMENTATION

The parallel transaction-oriented simulator was implemented using the Java-based grid environment ProActive (INRIA 2000) that is very well suited for ad hoc grids. The overall architecture of the parallel simulator follows the Master-Slave approach. Figure 2 shows the simplified architecture of the parallel simulator including its main components.

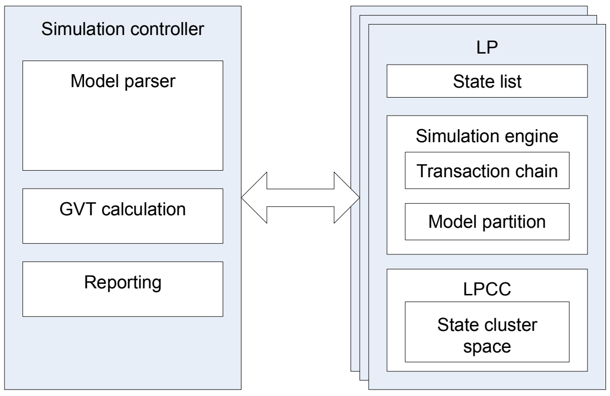


Figure 2: Architecture Overview

The main parts of the parallel simulator are the simulation controller and the LPs. The simulation controller steers the overall simulation. It is created when the user starts the simulation and will use the model parser component to read the simulation model file and parse it into an object structure representation of the model. After the model is parsed the simulation controller creates LP instances, one for each model partition. The simulation controller and the LPs are implemented as ProActive Active Objects so that they can communicate with each other via remote method calls. Communication will take place between the simulation controller and the LPs but also between the LPs themselves, for instance, in order to exchange transactions. Note that the communication between the LPs is not illustrated in Figure 2. After the LPs have been created and initialised, they will receive the model partitions that they are going to simulate from the simulation controller and the simulation is started. The main component of each LP is the simulation engine, which contains the transaction chain and the model partition that is simulated. The simulation engine is the part that is performing the actual simulation. It is moving the transactions from block to block by executing the block functionality using the transactions. The implementation of the LPs follows the optimistic TW algorithm. It use state checkpointing, i.e. each LP contains a state list that stores historic simulation states allowing them to perform rollbacks if required. Lazy cancellation is implemented in order to propagate the cancellation of transactions already sent to other LPs. There are other lists within the simulation engine that are not shown in Figure 2, for instance the list of transactions received and the list of transactions sent. The LPs are extended further to operate according to the SRTW algorithm as desribed by (Ferscha and Johnson 1999). This is archived by adding an LPCC and specific sensors into each LP. The sensor values are periodically read by the LPCC and converted into indicators. The most important indicators for the SRTW algorithm are the number of events committed per second and the average number of events in use. The first one describes the simulation progress and the second is artificially limited by the LPCC in order to reduce the optimism if required. For transaction-oriented simulation the equivalent indicators are the number of committed transaction moves per second and the average number of uncommitted transaction moves consisting of transaction moves that have been performed but not yet committed and transaction moves that are scheduled to be performed. The LPCC stores each indicator set in a cluster space and then uses it to find a similar past indicator set that promises better performance and to derive a new actuator value from the indicator set found. An option to disable the LPCC allows for the simulator to either operate in SRTW mode or in TW mode.
The simulation controller will perform GVT calculations necessary to establish the overall progress of the simulation and to allow LPs to reclaim memory through fossil collection. GVT calculation will also be used to confirm a provisional simulation end state that might be reached by one of the LPs.
When the end of the simulation is reached then the simulation controller will ensure that the partial models in all LPs are set to the correct and consistent end state and it will collect and assemble information from all LPs to output the post simulation report.

5. SIMULATION RESULTS

The performance of a distributed simulation depends on several factors such as the simulation model and how it is partitioned, the hardware performance of the nodes processing the LPs plus any additional loads on these nodes and the performance of the communication channels between the LPs. It also depends on how efficient the synchronization algorithm can utilise the parallelism within the model. This includes whether an optimistic simulation leads to many and possibly cascaded rollbacks because of over-optimistic processing. Over-optimistic processing can itself be a result of the simulation model or different and changing processing speeds of the LPs.
The evaluation of the simulator therefore concentrates on whether the SRTW algorithm can limit the optimism compared to TW if required and on other effects it has on simulation performance. The two example simulation models used were deliberately kept very simple in order to evaluate specific aspects of the algorithms. Each

example simulation model was run once in SRTW mode and once in TW mode. The validation runs were performed on the Grid'5000 platform with each LP and the simulation controller using a separate node of the Azur cluster at the Sophia Antipolis site. This cluster consists of IBM eServer 325 machines with two 2.0GHz AMD Opteron 246 CPUs per node. Communication between the nodes took place through a gigabit Ethernet link.

5.1. Reduction of Rolled Back Transaction Moves

The simulation model used for the first evaluation is shown in Figure 3. It contains two partitions each simulated by a separate LP. Both partitions have a GENERATE block and a TERMINATE block but in addition partition 1 also contains a TRANSFER block that with a very small probability of 0.001 sends some of its transactions to partition 2. The whole model is constructed so that partition 2 is ahead of partition 1 regards simulation time, achieved through the different configuration of the GENERATE blocks, and that occasionally partition 2 receives a transaction from partition 1. Because partition 2 is ahead in simulation time this will lead to rollbacks in partition 2. The simulation stops after 120000 transactions have been terminated in the second partition. This model attempts to emulate the common scenario where a distributed simulation uses nodes with different performance parameters or partitions that create different loads so that during the simulation the LPs drift apart and some of them are further ahead in simulation time than others leading to rollbacks and re-execution.

```
PARTITION Partition1,120000
GENERATE 1,0
TRANSFER 0.001,Label1
TERMINATE 0
PARTITION Partition2,120000
GENERATE 3,0,5000
Label1 TERMINATE 1
```

Figure 3: Simulation Model 1

The model was simulated once in SRTW mode and once in TW mode by enabling or disabling the LPCCs within all LPs of the simulator. We found that in SRTW mode the LPCC of LP2 successfully reduces the number of rolled back transaction moves compared to the TW mode by limiting the number of uncommitted transaction moves using the actuator. Figure 4 shows the actuator values set by the LPCC during the simulation. For some of the processing intervals no actuator value was set because the LPCC could not find a past state vector with a better performance indicator.

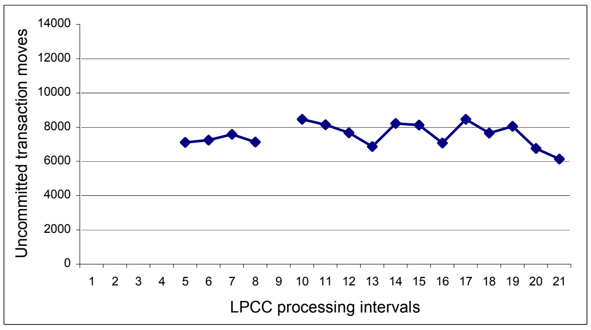


Figure 4: LP2 Actuator Value Graph

The actual reduction in the number of rolled back transaction moves within LP2 over the whole simulation can be seen in Table 1. It shows that the simulation run using SRTW required 74232 less rolled back transaction moves, which is around 20% less compared to the simulation run using TW. As a result the total number of transaction moves performed by the simulation was also reduced leading to a better average simulation performance of 5502 time unites per second in SRTW mode compared to 4637 time unites per second in TW mode.

Table 1: LP2 Processing Statistics

LP statistic item	SRTW	TW
Total committed transaction moves	119561	119961
Total transaction moves rolled back	308790	383022
Total simulated transaction moves	428789	503021

5.2. TW Outperforming SRTW

During the testing of the parallel simulator we found that in some cases the TW algorithm can outperform the SRTW algorithm. This second evaluation demonstrates this in an example. The simulation model is very similar to the one used in the first evaluation. It contains two partitions with the first partition transferring some of its transactions to the second partition but this time the GENERATE blocks are configured so that the first partition is ahead in simulation time compared to the second. The simulation is finished after 5000 transactions have been terminated in one of the partitions. The complete simulation model can be seen in Figure 5.

```
PARTITION Partition1,5000
GENERATE 1,0,4000
TRANSFER 0.3,Label1
TERMINATE 1
PARTITION Partition2,5000
GENERATE 1,0
Label1 TERMINATE 1
```

Figure 5: Simulation Model 2

As a result of the changed GENERATE block configuration and the first partition being ahead of the second partition in simulation time, all transactions received by partition 2 from partition 1 are in the future for partition 2 and no rollbacks will be caused. But it will lead to a steady increase of the number of outstanding transactions within partition 2 pushing up the indicator for the number of uncommitted transaction moves during the simulation.
The first simulation run was performed with the LPCC enabled, i.e. in SRTW mode. The significant effect of the simulation run is that the LPCC in LP2 starts setting actuator values in order to steer the local simulation processing towards a past state that promises better performance but because the number of uncommitted transaction moves within the second partition increases as a result of the transactions received from partition 1 the actuator limits set by the LPCC tend to be lower than the current number of uncommitted transaction moves resulting in the actuator limit being exceeded and the LP being switched into cancelback mode. The cancelback mode forces the LP2 to temporarily stop processing transactions and to cancel back some of the transactions received from LP1 leading to the overall simulation progress being slowed down. Figure 6 shows the actuator values applied by the LPCC in LP2. For most of the simulation the actuator was set to a value of around 2000 uncommitted transaction moves. All intervals that had an actuator value set led to the LPCC switching the LP2 into cancel back mode resulting in a significantly reduced rate of committed transaction moves in LP2 as shown in Figure 7.

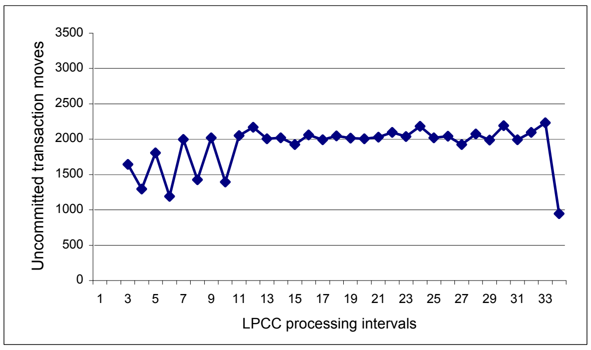


Figure 6: LP2 Actuator Value Graph

The second simulation run of this model was performed with the LPCC disabled, i.e. in TW mode. There were no rollbacks during the simulation and none of the LPs were artificially slowed down leading to an optimum average simulation performance for the model and setup of 165.7 time units per second. In SRTW mode the average simulation performance for the same model was dramaticaly reduced to 27.7 time units per second.

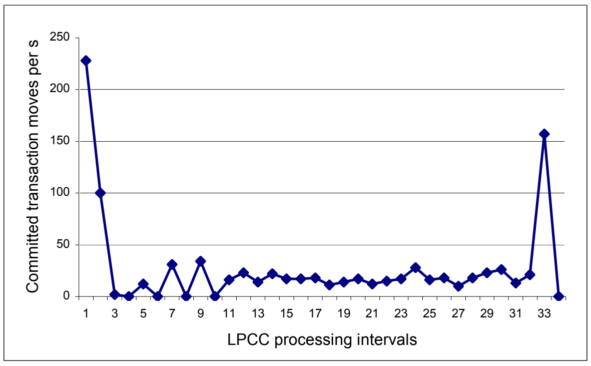


Figure 7: LP2 Simulation Progress Rate

6. DISCUSSION AND FUTURE WORK

The evaluation using the first example simulation model demonstrated that the SRTW algorithm can successfully reduce the amount of rolled back simulation work compared to TW. Although the number of rolled back transaction moves was only reduced by around 10% for the example simulation model used, much higher reductions can be expected for more complex simulation models that result in cascaded rollbacks.
The second example simulation model revealed a problem of the SRTW algorithm, implemented as described in (Ferscha and Johnson 1999). The TW algorithm provides ideal performance for this model because no rollbacks were caused and therefore none of the simulation work needed to be undone. Any superior synchronization algorithm would be expected to perform this simulation model with similar performance like TW. However, the SRTW algorithm performs significantly worse than the TW algorithm because it attempts to limit optimism in an LP that is already behind in respect of simulation time and that is not causing any rollbacks leading to a significant slowdown of the overall simulation progress. The SRTW algorithm fails in this scenario because the adaptive optimization within the LPs is purely based on local information that is not always sufficient for the algorithm to make the correct decision. Making the LPs aware of their position within the global progress window (GPW) as suggested by (Tay et al. 1997) could be a way of avoiding such problems. The SRTW algorithm already requires GVT calculations in order to establish the values of its performance indicators and this GVT calculation could easily be extended to calculate the global furthest time (GFT) giving all LPs an additional indicator of their position within the GPW.
In future work the parallel simulator could be extended to use a changed SRTW algorithm that is aware of the GPW and perhaps also implement the adaptive throttle scheme as described by (Tay et al. 1997) so that further comparisons of these algorithms are possible. Future studies could also investigate how the SRTW algorithm or an improved version of it reacts to communication delays that are likely to occur in Internet based grids.

7. CONCLUSION

We briefly discussed the requirements for a synchronisation algorithm suitable for ad hoc grid environments as well as transaction-oriented simulation. Further requirements for parallel transaction-oriented simulation were analysed and possible solutions suggested. The SRTW algorithm was chosen as a promising algorithm that fulfils the requirements. The algorithm was adapted to transaction-oriented simulation and a parallel simulator was implemented using the grid environment ProActive. Our parallel simulator can operate in SRTW mode as well as in TW mode allowing a comparison of the two algorithms for different transaction-oriented simulation models.
The evaluation of the parallel simulator showed that the SRTW algorithm can successfully reduce the number of rolled back transaction moves, which for simulations with many or long cascaded rollbacks will lead to a better simulation performance. But it also revealed a weakness of the SRTW algorithm. Because LPs try to optimise their properties based only on local information it is possible for the SRTW algorithm to perform significantly worse than the TW algorithm. Future work on this simulator could improve the SRTW algorithm by making the LPs aware of their position within the global progress of the simulation.

ACKNOWLEDGMENT

This research work was funded partially by the European Commission under the research and development project GridCOMP (Contract IST-2006-034442). Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS, RENATER and other contributing partners (https://www.grid5000.fr).

REFERENCES

Das, S.R. 1996. “Adaptive protocols for parallel discrete event simulation”. *Proceedings of the 28th conference on Winter simulation* (Coronado, CA, USA), ACM Press, New York, 186-193.

Das, S.R. and R.M. Fujimoto. 1997. “Adaptive memory management and optimism control in time warp”. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 7(2), ACM Press, New York, 239-271.

Ferscha, A. and J. Johnson. 1999. “Shock resistant Time Warp”. *Proceedings of the thirteenth workshop on Parallel and distributed simulation* (Atlanta, Georgia, USA), IEEE Computer Society, Washington, DC, 92-100.

Fujimoto, R.M. 1993. “Parallel and Distributed discrete event simulation: algorithms and applications”. *Proceedings of the 25th conference on Winter simulation* (Los Angeles, USA), ACM Press, New York, 106-114.

Gafni, A. 1985. “Space Management and Cancellation Mechanisms for Time Warp”. Ph.D. dissertation TR-85-341. Dept. of Computer Science, University of Southern California.

INRIA. 2000. *ProActive - Programming, Composing, Deploying on the Grid.* [online] Available from: http://proactive.inria.fr/ (Date viewed 10 February 2008).

Jefferson, D.R. 1985. “Virtual time”. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7(3), ACM Press, New York, 404-425.

Schriber, T.J. 1991. “An Introduction to Simulation Using GPSS/H”. John Wiley & Sons.

Smith, M.; T. Friese; B. Freisleben. 2004. “Towards a Service Oriented Ad-Hoc Grid”. *Proceedings of the 3rd International Symposium On Parallel and Distributed Computing* (Cork, Ireland), IEEE Computer Society, Washington, DC, 201-208.

Srinivasan, S. and Jr.P.F. Reynolds. 1995. “NPSI adaptive synchronization algorithms for PDES”. *Proceedings of the 27th conference on Winter simulation* (Arlington, Virginia, USA), ACM Press, New York, 658-665.

Tay, S.C.; Y.M. Teo; S.T. Kong. 1997. “Speculative parallel simulation with an adaptive throttle scheme”. *Proceedings of the eleventh workshop on Parallel and distributed simulation* (Lockenhaus, Austria), IEEE Computer Society, Washington, DC, 116-123.

AUTHOR BIOGRAPHIES

GERALD KRAFFT was born in Schwedt/Oder, Germany and first went to the University of Wismar, where he studied Computer Science and obtained his degree in 1998. He subsequently relocated to London, U.K. where he works as a senior software developer and joined the University of Westminster for a postgraduate degree in Advanced Computer Science, which he completed with distinction in 2007. He has a particular interest in parallel and distributed systems and computer simulation. His Web-page is http://perun.hscs.wmin.ac.uk/~gerald/.

VLADIMIR GETOV leads the Distributed and Intelligent Systems Group at the University of Westminster in London. His current research interests are focussed on component-oriented design of Grid platforms and applications, autonomous distributed systems, parallel architectures and performance, mixed-language high-performance programming environments with Java, and hybrid programming models and paradigms. Professor Getov has over 100 publications including edited volumes, articles or chapters in books, journal and conference papers, technical reports, as well as invited and tutorial lectures, seminars, design prototypes, etc. His Web-page is http://perun.hscs.wmin.ac.uk/~vsg/.

Automatic Development of High Performance Multi-Physics Simulators

Félix Christian Guimarães Santos, Email: flxcgs@yahoo.com.br
Eduardo Roberto R. de Brito Junior, Email: errbj@yahoo.com.br
José Maria Bezerra, Email: zemaria@ufpe.br
Federal University of Pernambuco
Department of Mechanical Engineering
Rua Acadêmico Hélio Ramos, s/n - Recife - PE 50740-530 - Brazil

***Abstract*—MPhyScas - Multi-Physics and Multi-Scale Solver Environment - is a computational system aimed at supporting the automatic development of simulators for coupled problems, developed at the Department of Mechanical Engineering of the Federal University of Pernambuco - Brazil. It provides a framework, which is flexible enough to accommodate representations for all levels of computation that can be found in simulators based on the finite element method. MPhyScas is built on a set of a powerful language of patterns supporting abstractions for solution algorithms; phenomena, geometric entities; phenomenon-phenomenon and phenomenon-geometry relationships and others, together with a library of low level entities - like finite elements, reference finite elements, numerical integration tools, and so on. In despite of its completeness in what regards all stages of a multi-physics simulation, the current version of MPhyScas produces sequential simulators only. Thus, it does not support any kind of communication between its computational entities besides those defined by direct references (pointers). In this work we present the architecture of an improvement of MPhyScas, called MPhyScas-P (MPhyScas Parallel), which can be used for the automatic development of either sequential or parallel simulators. We take an advantage of the architecture in layers of MPhyScas in order to define a hierarchical parallel computational scheme in such a way that communication procedures are automatically identified, localized and built. That hierarchy also provides a natural way of defining data structures and access dynamics for all memory levels, providing simpler ways of dealing with non-uniform memory access patterns. Some preliminary results obtained with a prototype will be shown and analyzed.**

***Index Terms*—Finite element method, Simulator, Multi-physics, Coupled phenomena**

I. Introduction

MPhyScas (Multi-Physics Multi-Scale Solver Environment) is an environment dedicated to the automatic development of simulators based on the finite element method. The term multi-physics can be defined as a qualifier for a set of interacting phenomena defined in space and time. These phenomena are usually of different nature (deformation of solids, heat transfer, electromagnetic fields, etc.) and may be defined in different scales of behavior (macro and micro mechanical behavior of materials). A multi-physics system is also called a system of coupled phenomena. If two phenomena are coupled, it means that a part of one phenomenon's data depends on information from other phenomenon. Such a dependence may occur in any geometric part, where both phenomena are defined. Other type of data dependence is the case where two or more phenomena are defined on the same geometric component and share the geometric mesh. Multi-physics and multi-scale problems are difficult to simulate and the building of simulators for them tend to be very costly in terms of time spent in the programming and testing of the code. The main reason for that is the lack of reusability. A detailed discussion can be found in [1]-[2].

Usually, simulators based on the finite element method can be cast in an architecture of layers. In the top layer global iterative loops (for time stepping, model adaptation and articulation of several blocks of solution algorithms) can be found. This corresponds to the overall scenery of the simulation. The second layer contains what is called the solution algorithms. Each solution algorithm dictates the way linear systems are built and solved. It also defines the type of all operations involving matrices, vectors and scalars, and the moment when they have to be performed. The third layer contains the solvers for linear systems and all the machinery for operating with matrices and vectors. This layer is the place where all global matrices, vectors and scalars are located. It is also responsible for the definition of the finer details for the assembling of matrices and vectors. The last layer is the phenomenon layer, which is responsible for computing local matrices and vectors at the finite element level and assembling them into global data structures.

The definition of those layers is important in the sense of software modularization. But it does not indicate neither how entities belonging to different layers interact nor what data they share or depend upon. That is certainly very important for the definition of abstractions, which could standardize the way those layers behave and interact. The architecture of MPhyScas presents a language of patterns in order to define and represent not only a set of entities in each layer - providing the needed layer functionalities - but also the transfer of data and services between the layers. Thus, MPhyScas is a framework that binds together a number of computational entities, which were defined based on that language of patterns, forming a simulator. Such a simulator can easily be reconfigured in order to change solution methods or other types of behavior [3]-[4]. Almost every single piece of code that constitute MPhyScas computational entities in a simulator can be reused in the building of other different simulators. This makes the simulators produced by MPhyScas strongly flexible, adaptable and maintainable.

However, the original architecture of MPhyScas provides

Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

support for the automatic building of sequential simulators only. For instance, it does not have abstractions that could automatically define the distribution of data and procedures and their relationships across a cluster of PC's. In this work we briefly present MPhyScas architecture for sequential simulators (called MPhyScas-S from now on) in order to proceed with the main part of the work, i.e., the definition of a new parallel architecture. This new architecture, called MPhyScas-P, should satisfy a number of new requirements, including the support for the parallel execution of the simulators in clusters of PC's. MPhyScas-S is a framework with the support of an extended finite element library and a knowledge management system. MPhyScas-P uses the same extended library and knowledge management system from MPhyScas-S (with minor differences). It also makes use of the concept of layers already used in MPhyScas-S in order to define a hierarchical parallel structure. Such a hierarchy is useful for the automatic definition of synchronization schemes; data partition and distribution procedures; inter-process communication patterns and data management across several levels of memory. Procedures are automatically specialized for the pre-processing; the simulation and the post-processing phases, depending on the hierarchical distribution and on the characteristics of the hardware being used. Also, only two types of communication between processes during a simulation, and patterns are defined for their representation.

This work is organized as follows: first the architecture of MPhyScas-S is briefly described paving the ground for describing MPhyScas-P's architecture. Next, comments on some relevant related work are provided in order to build a context for this article. After that is done, the architecture of MPhyScas-P is presented, followed by a description of its behavior (in the context of the work load and flow requirements). In the end some conclusions are drawn. Our purpose here is more descriptive than analytic. However, whenever needed some comments will be provided in order to make things a bit clearer. In the end some conclusions are drawn.

II. The Architecture of MPhyScas-S

The architecture of MPhyScas-S [5], [4] establishes a computational representation for the computational layers using some design patterns (see Figure 1), where the **Kernel** Level represents the global scenery level, the level of the solution algorithms is represented by the **Block** Level, the level of solvers is represented by the **Group** Level and the phenomena level is represented by the **Phenomenon** Level. The definition of this structure is aimed at improving the quality of simulators design. The defined architecture attempts to fill in the existing gap in the development of FEM simulators for multi-physics and multi-scales problems. The main requirements of this architecture are: (i) Flexibility in the development of simulators; (ii) Extensibility of simulators through the integration of components and (iii) Improved reusability of code, data and models.

The architecture of MPhyScas-S is shown in Figure 2. The Static Library allows the maintenance of data employed in the building of simulators and simulations. Those data includes: methods (mesh generation, numerical integration,

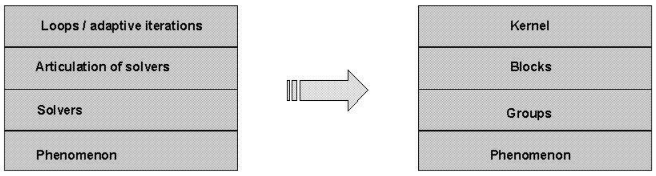


Fig. 1. Computational representation for the layers of the simulator

for instance), functions (constitutive parameters, for instance), algorithms and phenomena. The Pre-Processor produces **Data for Simulation** and builds the **Simulator** using the **Static Library**. The **Data for Simulation** represents the input data in a simulation as it is used by the **Simulator**. The **Simulator** is responsible for the execution of a simulation. The **Simulator** uses the **Data for Simulation** and produces the **Results of the Simulation**. The **Viewer** uses the **Results of the Simulation** and the **Data for Simulation** to produce the visualization of the simulation results.

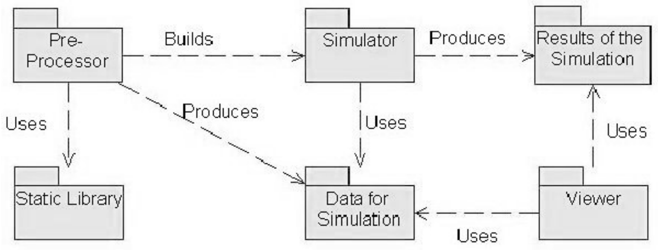


Fig. 2. Architecture of the MPhyScas

The **Simulator** is considered as a pattern [6], [3] and [7] (see Figure 3), which, simply speaking, is a workflow in the form of a DAG (Direct Acyclic Graph) and divided into four layers, which strongly follows the slices of concerns for each layer, already cited in the introduction:

i) **Kernel**: it is responsible for initialization procedures (transferred to **Blocks** in the lower level); for global time loops and iterations; for global adaptive iterations and for articulation of activities to be executed by the **Block**s in the lower level. The **Kernel** stores system data related to the parameters for its loops and iterations;
ii) **Block**: it is responsible for the transfer of incoming demands from the **Kernel** to its **Group**s in the lower level (initialization procedures, for instance); for **Block** local time loops and iterations (inner loops and iterations inside a global time step, restricted to groups of phenomena); for procedures inside time stepping schemes; for **Block** local iterations (restricted to some groups of phenomena, like in a Newton-Raphson iteration, for instance); for **Block** local adaptive iterations (restricted to some groups of phenomena); for operations with global quantities (transferred to **Group**s in the lower level, which are the owners of global quantities). The **Block**s serve the **Kernel** level. Each **Block** is responsible for a certain number of **Group**s, which can not be owned by other **Block**. All demands from a **Block** to the lower level should be addressed to its **Group**s. The **Block**s store system data related to parameters for their own loops and iterations

and parameters for their procedures;

iii) **Group**: it is responsible for the transfer of incoming demands from its **Block** to **Phenomena** in the lower level (initialization procedures, for instance); for the assembling coordination and solution of systems of linear algebraic equations (the method used depends on the solver component); for operations with global quantities (by demand from its **Block**), for articulation of activities to be executed by its **Phenomena** in the lower level (basically concerned with computation and assembling of global matrices and vectors). The **Group**s serve their respective **Block**s. Each **Group** is responsible for a certain number of **Phenomena**, which can not be owned by other **Group**. All demands from a **Group** to the lower level should be addressed to its **Phenomena** only. The **Group**s store global matrices, vectors and scalars and store the **GroupTask**s, which are objects encapsulating standard procedures, where articulation of the **Group**'s **Phenomena** are needed. The **GroupTask**s are programmable and their data are standard pieces of information, depending only on the type of the **GroupTask**.

iv) **Phenomenon**: it is responsible for the computation of local matrices, vectors and scalars (**Phenomenon** quantities); for operations involving matrices and vectors at the finite element level and their assembling into given global matrices and vectors. The **Phenomena** serve their respective **Group**s. The **Phenomena** store data related to constitutive parameters or other parameters, which are specific of the respective Phenomenon; store the geometry where the respective **Phenomenon** is defined (different **Phenomena** may share a geometry or a part of it); store **WeakForm**s, which are tools for computing and assembling quantities defined on a certain part of the geometry. A **WeakForm** may be active or not. Only active **WeakForm**s can be used during a simulation. A **WeakForm** may store parameters, which are related to specific simulation data (for instance, functions for the definition of boundary conditions or parameters needed for the computation of a quantity, which should be given together within a simulation data set). The **Phenomenon** should store methods, which are tools to be used in certain **Phenomenon** specific tasks. For instance, those tasks can be generation of geometric and **Phenomenon** meshes, numerical integration at the element level, shape functions, etc.

The simulation starts with the execution of the root of the **Kernel**, which uses services provided by a set of **Block**s, which in turn uses services from a set of **Group**s. Each **Group** owns a set of **Phenomenon** objects, which are used to perform the production of local matrices and vectors and the assembling of them into given (by the **Group**) global matrices and vectors.

The states that define the configuration of each **Phenomenon** object are stored in the respective **Group** object, where solvers are located. This is convenient due to the fact that the **Group**'s layer is responsible not only to assemble and solve algebraic systems, but also to operate with scalars, vectors and matrices in response to requests from a **Block** (see Figure 5).

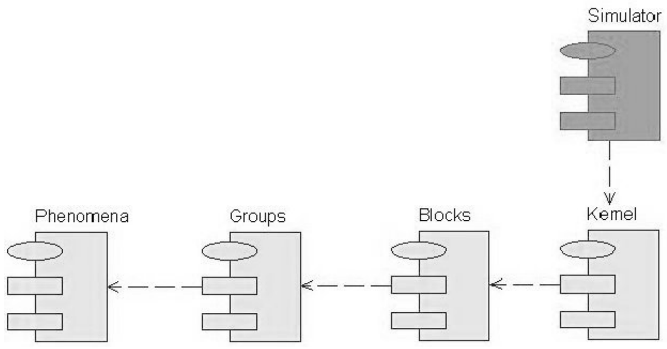
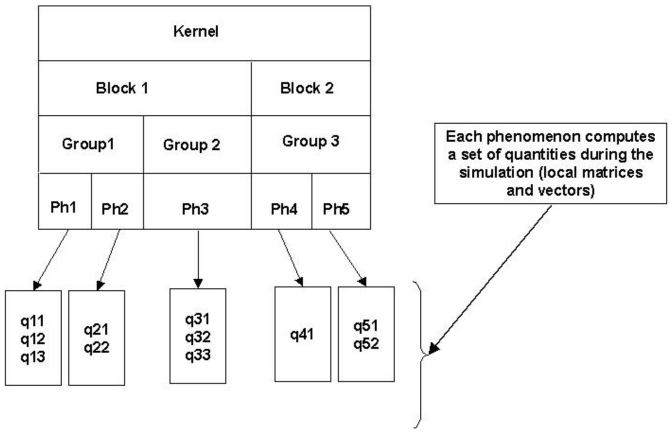


Fig. 3. Simulator diagram



Kernel
Block 1
Block 2
Group1
Group 2
Group 3
Ph1
Ph2
Ph3
Ph4
Ph5
q11 q12 q13
q21 q22
q31 q32 q33
q41
q51 q52
Each phenomenon computes a set of quantities during the simulation (local matrices and vectors)

Fig. 4. Each Phenomenon object is able of computing a set of quantities during a simulation

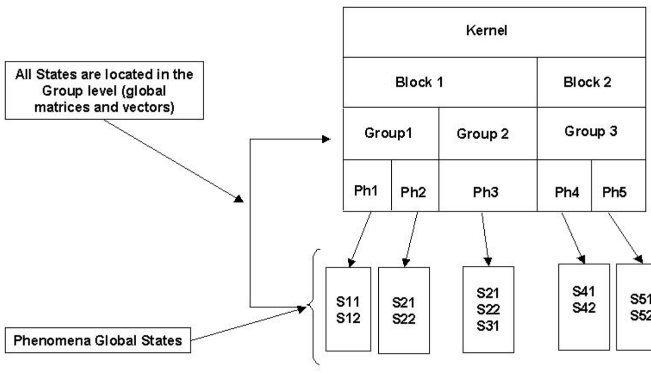


Fig. 5. Each Phenomenon object has its own set of states, which is stored in its Group object

A quantity that a **Phenomenon** object can compute and assemble may be coupled to other **Phenomenon**'s states (one or more) as it is depicted in Figure 6. MPhyScas-S provides all the machinery to make this procedure automatized following the specification of some data related to the place where coupling occur; handlers for the states and a reference to the coupled **Phenomenon** object, which should be given to the object responsible for the computation.

For further detailed information on MPhyScas-S see [4]. In the next section we provide some relevant related work.

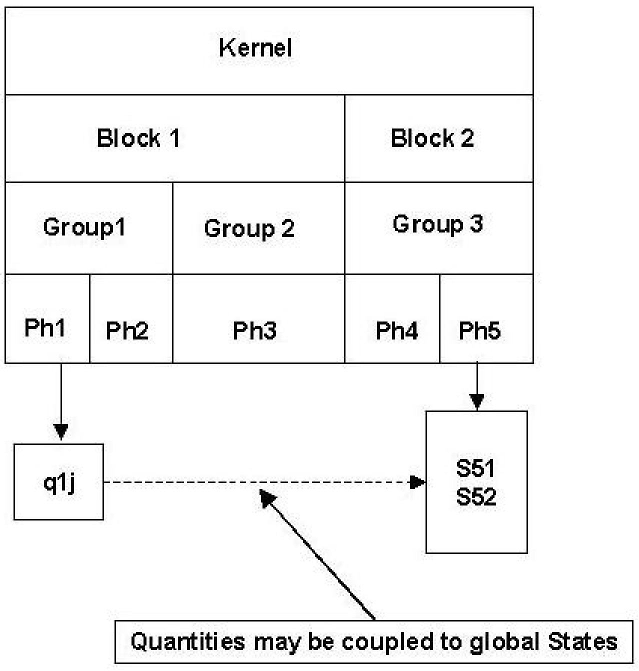


Fig. 6. A quantity computed by a Phenomenon object may be coupled to a state from other Phenomenon object

III. Related Work

Definition and building of computational frameworks that support programming of simulator for multiphysics problems has been a very active area in the last decade. For the sake of providing a simpler context for the present work, we classify current research efforts into only two classes: (i) libraries, which, besides providing important abstractions supporting data, procedures and relationships for coupled physics simulation, do not provide a structural guidance (architecture) for the building of simulators, and (ii) frameworks, which provide a deep structure of abstractions and patterns in the form of an architecture. We will comment more on the second class, since MPhyScas is a framework with those characteristics. As examples of class (i) we cite the Component Template Library (CTL) [8] and Comsol [9]. CTL provide abstractions that support the building of solutions algorithms for loose and tight coupling (procedures in the level of Group and Phenomena). It is an implementation of the component concept with an RMI semantic similar to CORBA or Java-RMI components, which can be used to build complex parallel simulators. It is sophisticated in the sense that allows component in several languages (C, C++, FORTRAN) and different communication models (RMI, MPI, PVM, threads) among other features. Comsol is a commercial package and not much of its internal behavior is publicly exposed. Nevertheless, it provides abstractions that allow users to define coupled procedures through handlers to vector fields and provide encapsulated access to high performance computing. It has a sophisticated GUI with CAD and visualization modules. However, besides varying levels of support for HPC, neither one of them provides structural guidance for simulators (levels Kernel through Phenomena), leaving that task to the user.

In the class of frameworks (class (ii)), one can find powerful packages, such as Uintah [10], Cactus [11] and Sierra [12]. They are substantially different and have been applied to extremely sophisticated simulations. Since the architecture of MPhyScas is closer to Sierra's architecture, we will comment on this framework with more detail. Uintah Computational Framework and Cactus Framework consist of a set of software components and libraries that facilitate the solution of Partial Differential Equations (PDEs) on Structured AMR (SAMR) grids using hundreds to thousands of processors. Although they do not provide a structure for simulators as done by Sierra and MPhyScas, they are in this class due to how they bind components together; define and use coupling information and provide access to high performance (e.g. parallel) processing through a shared service infrastructure. That characterizes them as having a deep interoperability system. Uintah uses CCA (Common Component Architecture) [13] for designing and describing components interfaces. It does not have a pre-defined structure for simulators. Thus, in order to provide framework functionality it defines on top of its primary set of abstractions another set of components and supporting libraries, which targets the solution of PDE's on massively parallel architectures. This set is called Uintah Computational Framework (UCF). UCF builds a graph called **TaskGraph**, which describes the data dependencies between the various computation steps in a simulator. Also, it defines a simulator's workflow as a direct acyclic graph of **Task**s. Communication between Tasks is made through a DataWareHouse, which provides the illusion that all memory is global. If a Task correctly describes its data dependencies, then the data stored in the DataWareHouse will match the desired data (variable and region of state). Communication is scheduled by a local scheduling algorithm that approximates the true globally optimal communication schedule. Because of the flexibility of single-assignment semantics, the UCF is free to execute tasks close to data or move data to minimize future communication [10].

The following nice summary of Cactus structure is found in [14]: ”The Cactus **Flesh** acts as the coordinating glue between modules that enables composition of the modules into full applications. The Flesh is independent of all modules, includes a rule based scheduler, parameter file parser, build system, and at run time holds information about the grid variables, parameters, methods in the modules and acts as a service library for modules. Cactus modules are termed **Thorns** and can be written in Fortran 77 or 90, C or C++. Each thorn is a separate library providing a standardized interface to some functionality. Each thorn contains four configuration files that specify the interface between the thorn and the Flesh or other thorns (variables, parameters, methods, scheduling and configuration details). **Drivers** are a specific class of Cactus Thorns that implement the model for parallelism. Each solver thorn is written to an abstract model for parallelism, but the Driver supplies the concrete implementation for the parallelism” (see also [11]).

Sierra framework provides a structural guidance in layers composed of (from top to bottom) **Application**, **Procedure**, **Region** and **Mechanics**. Application articulates user-provided algorithms in order to establish high-level activities. It uses services from a set of Procedures, which can freely articulate Regions using a set of user-provided algorithms. A Region defines activities, which are related to a fixed geometric region, for which a mesh is provided. Those activities are defined by user-provided algorithms. It uses services provided by a set of Mechanics. In order to perform the desired work, a Mechanics uses a set of MechanicsInstances and a set of user-provided algorithms. A Mechanics implements procedures related to a specific physics - defined on a subset of its Region's mesh - and its MechanicsInstances are responsible for atomistic operations defined on a subset of its Mechanics' mesh. A Mechanics may use another set of Mechanics, building more layers downwards. This may be used in multiscale computations, where a lower level Mechanics is used to compute constitutive data for a MechanicsInstance of its parent Mechanics [12]. If a Mechanics A needs data from another Mechanics B (provably defined in another Region), the Advanced Services of Sierra provide means to transfer mesh-dependent data from one mesh to the other. The result is then stored in the Region of Mechanics A. The SIERRA Framework Core Services manage the parallel distribution of mesh objects for an application. Management of a parallel distributed mesh is defined in three parts: (i) policies and distributed mesh sets, relations, and data structures; (ii) parallel operations that do not modify the distributed-mesh data structures, and (iii) operations that modify the distributed-mesh data structures. Sierra has a sophisticated management system for parallel operations, which is strongly supported by its defined topology. As far as the authors are concerned Sierra supports only SPMD (single process multiple data) type of parallel processing.

Clearly, it is possible to provide a parallel between the architecture of MPhyScas and that of Sierra. Application relates to Kernel; Procedure relates to Block; Region relates to Group; Mechanics relates to Phenomenon and MechanicsInstance relates to WeakForm (geometry-related atomistic piece of code). However, there are several and important notes that should lead to marked differences:

i) MPhyScas is strongly concern-oriented, instead of procedure-oriented or context-oriented. Concerns are more easily mapped into requirements and architectures. Also, concerns are related to the fundamentals of the classes of problems being tackled and are less vulnerable to programming traditions and limitations. Adequate separation of concerns may lead to more reusable, maintainable and adaptable code. Therefore, MPhyScas was built to satisfy a nested set of concerns (functional and non-functional) related to the numerical approximation to solutions of partial differential equations (mainly those solutions defined by the finite element method). For instance, layers in MPhyScas are slices of the code following a set of concerns already cited before. Sierra is more procedure-oriented.

ii) The specification for the Kernel (simulator's scenery) is more detailed than that for Sierra's Application. It has only one shallow algorithm limited by the designed responsibilities of the layer of Blocks. Of course, that algorithm can be freely designed, but should satisfy the concerns specified by what we call the Scenery of the Simulation (it can be understood as a general specification for the simulator, which gets more refined when requirements for the other layers are detailed).

iii) MPhyScas's Block is quite similar to Sierra's Procedure in their generality (their behavior is determined solely by their algorithms). However, one Block never shares a Group with other Block. This constraint does not apply to the relationship between Procedure and Region in Sierra framework. The justification for both Procedure and Block is the need for the articulation (in adaptive iterations, nonlinear solver iterations, and other situations) of sets of solvers involving different phenomena (physics). As Sierra does it, MPhyScas limited the depth of this layer in a slice of a generic simulator, where activities related to time stepping methods, nonlinear iterations and other processes are defined and articulated for one fixed set of subsets of phenomena . However, in MphyScas each Block also has the responsibility to define its **cone of influence** in a disjoint way. The reason is that MPhyScas would like to support dynamic exchange of components in all levels of computation. Therefore if two Blocks were allowed to establish relationships to the same Group, concerns related to both would get messed up, making it extremely difficult to define the parts of code affected by changes in one Block.

iv) A Region in Sierra is based on operations (Mechanics and algorithms) and vector fields, which are defined on a geometric entity and its geometric mesh. Transfer of vector fields from one mesh to the other is provided by the Advanced Services of Sierra. All vector fields in a Region are defined in subsets of the same mesh. The motivation for the Group in MPhyScas is based on a set of Phenomenon objects and their data, which participate in formation of linear monolithic algebraic systems. Thus, all data needed to assemble and solve those algebraic systems are stored in the Group. Thus, there might be Phenomenon objects in a Group defined on different meshes. That becomes manageable because a Group does not know anything neither about geometry domains nor about meshes. The transfer of vector fields between meshes is performed by a especially designed Phenomenon (instantiated and executed as a normal Phenomenon object). Those pieces of information are located in the respective Phenomenon objects. There are special data structures that allow two Phenomenon objects to share the mesh of a geometric entity, whenever they are defined in that geometric part [4]. All matrices and vectors are stored in the Group and their relationship with the Group's Phenomenon objects is described by user-provided data. The location of matrices and vectors in the Group was motivated by the location of linear solvers in the Group. All dependencies between Phenomenon objects are resolved in the Phenomenon layer. Besides the solvers, a Group is entirely programmable; does not depend on other user-defined algorithm and does not share Phenomenon objects with other Group (therefore providing the influence cone). There are other differences, but the cited references are able of providing further information.

v) Mechanics in Sierra encapsulate procedures related to a particular physics. It articulates its MechanicsInstances and algorithms in order to provide the computation of quantities and the assembling of them. MPhyScas provide those functionalities with Phenomenon objects and their activated WeakForm objects. A Phenomenon object accepts algorithms for activities such as numerical integration, error estimation, mesh adaptation (geometric and phenomenon meshes), shape functions (trial and test), mesh generation (geometric and phenomenon meshes). A Phenomenon has two types of meshes: geometric and phenomenon. Phenomenon meshes describe the distribution of polynomial order of approximation over the geometric mesh. It seems Sierra does not support p and h-p adaptivity, because it does not data structures for such procedures. This certainly would complicate transfer procedures and the way coupled vector fields are used in Sierra. That is supported by MPhyScas and is one of the concerns, which was considered when placing meshes at the Phenomenon layer.

Both architectures (MPhyScas' and Sierra's) were developed independently. The first definition of the MPhyScas' architecture was published in 2001. Nevertheless, they present a similar structure with some marked differences (mainly in the execution graph and placement of some data structures), which get sronger in MPhyScas-P, where the tree structure of the architecture of MPhyScas is used in the control of the simulator execution on a cluster of PC's (see next section). It is important to note that while MPhyScas-P is in the beginning of its development, Sierra is already a mature, complex, fully developed system with far more functionalities than MPhyScas-P. MPhyScas-P is being developed to be applied in a production environment (analysis of material degradation for the petroleum industry).

IV. The Architecture of MPhyScas-P

In modern clusters of PC's one can identify at least four hierarchical levels of different procedures and/or memory usage:

i) **Cluster Level**: it is composed of all processes running in all machines being used in a simulation.

ii) **Machine Level**: it is composed of all processes running in one individual machine among all those used in a simulation

iii) **Processor Level**: it is composed of all processes running in one individual processor among all those running in one individual machine.

iv) **Process Level**: it is composed of one single process running in one individual processor (provably multi-core) among all other processes in this same processor. It can be divided into two groups:

  iv.i) **Core Sub-level**: it is composed of all parts of the code from one individual process, which is not strongly hardware specific.

  iv.ii) **Software-Hardware (SH) Sub-level**: it is composed of all parts of the code from one individual process, which is strongly hardware specific (cache management, fpga acceleration, etc.)

Whenever the architecture of a computational system allows for a hierarchy of procedures, it may be a good idea to define a hierarchy of processes in such a way that few of them would accumulate some very light management tasks. The benefits for this strategy include: (i) procedures can be hierarchically synchronized (from coarse to fine grain), reducing management concerns and increasing correctness; (ii) since locality concerns change along the hierarchy levels, memory management can become more and more specialized from top to bottom. Communication processes can also benefit from locality knowledge; (iii) The hierarchy allows for the encapsulation of concerns, making it easier the design of exchangeable components. Besides the natural benefit of this aspect, it also allows for the adaptation of the code to new hardware and software technologies, without incurring in heavy reprogramming in all levels of the hierarchy.

A. Interprocess Communication Process

Next we provide a description of how interprocess communication is considered in MPhyScas-P architecture. Communication between processes is a very important issue for problems with only one physics and gets even more crucial for multi-physics problems. The cause for that is the fact that for those types of problems communication is not needed only to complete information during linear algebra operations, but also to transfer information (vector fields and meshes) from one phenomenon to the other. Furthermore, such a data dependence between phenomena does not occur only on the boundaries between two mesh components, but on any geometric entity. There are other complication factors that contribute to make things even worse. Changing solution algorithms means that the linear systems, which are going to be solved, may be dramatically changed. For instance, changing from a monolithic scheme to an operator splitting one will require the solutions of several different coupled linear systems (inside an iteration loop) instead of only one.

The architecture of MPhyScas-S has satisfactorily solved those problems related to data dependence and sharing between Phenomenon objects for the sequential processing. It is also able of representing solution algorithms in such a way that the entire simulator can be adapted with a minimum amount of reprogramming (maximum amount of reuse). However, the essential difference, when considering parallel processing, is the appearance of interprocess communication procedures, which can be of four types:

i) **Communication during linear algebra operations**: The inclusion of code parallelization as a requirement implies that interprocess communication is needed during linear algebra operations. For instance, parallel matrix-vector multiplication requires interprocess communication in order to complement the job done by each process.

ii) **Communication along process hierarchy**: The establishment of the hierarchy of procedures will require some processes to assume some kind of leadership depending on the layer, where they are located. This will require interprocess communication throughout the hierarchy.

iii) **Communication for coupled information - I**: MPhyScas-S has dealt with coupling dependences between different phenomena already, but in parallel processing this type of dependence can become very complex. For instance, this is the case whenever the interface between two coupled phenomena coincides with a boundary between two components of the mesh partition. That means that vector fields and respective meshes data have to be transferred from one process to the other.

iv) **Communication for coupled information - II**: If two coupled phenomena have different geometric meshes, all components in one mesh partition may be different from all components in the other partition. Thus, there will be a need for interprocess data transfer from one phenomena to the other, whenever coupled quantities have to be computed.

Well-thought distribution schemes and data representation abstractions can eliminate both types of communication for coupled information. This can be done by copying the coupled vector fields and meshes data, to the processes where they are needed. Those copies will be updated whenever needed (communication of type (i) only). Processes will have to be given a larger memory space, but that can be made a minimum. The important thing is that the whole data set of a coupled mesh will not be transferred between two processes when a coupled quantity is to be computed. Thus, only types (i) and (ii) will be needed. They will be called communications of Type-I and Type-II, respectively.

In order to simplify the presentation of the MPhyScas-P's architecture some requirements have to be made: (i) If two or more phenomena are coupled in one geometric entity, then they share the same geometric mesh on that geometric entity. We will not consider transfer of data between different meshes in this work; (ii) All phenomena have to be represented in each process with a nonempty geometric mesh; (iii) Only three hierarchical levels will be considered in this work: Cluster, Machine and Process Levels. We do not differentiate the running processes by their processors in the same machine. Furthermore, we will not divide a process code into Core and Software-Hardware Sub-levels.

Requirements (i) and (iii) are made for simplification of explanation, since if they were not made, some details about mesh partition and distribution and the use of software-hardware procedures would be needed, blurring the center piece of this work. Requirement (ii) is needed because of the lack of a better alternative: we are making an option for a SPMD scheme. A MPMD scheme would require an automatic analysis of the solution algorithm in order to decide what procedure branches could be executed in parallel. Such a solution algorithm analysis is still the subject of an ongoing work.

B. Logical and Topological Views

There are two views of the MPhyScas-P's architecture:

i) **Logical View**: the logic of MPhyScas-P's workflow is the same as the MPhyScas-S', that is, it has the same levels of procedures (Kernel, Blocks, Groups and Phenomena), all relationships between them are preserved and all procedures within each layer are technically the same (besides the fact that data are now distributed). Therefore the relationships among entities in the different levels of MPhyScas-P is also a DAG (direct acyclic graph). Thus, we are able of cloning a suitable modification of MPhyScas-S to all processes in a SPMD scheme. In this sense, one can imagine that MPhyScas-P is MPhyScas-S with distributed data and a hierarchical synchronization scheme (see Topological View below).

ii) **Topological View**: the topology of the procedures in the workflow of MPhyScas-S is implemented in MPhyScas-P in a hierarchical form with the aid of a set of processes, which are responsible for the procedures synchronization. There are three types of leader processes (see Figure 7):

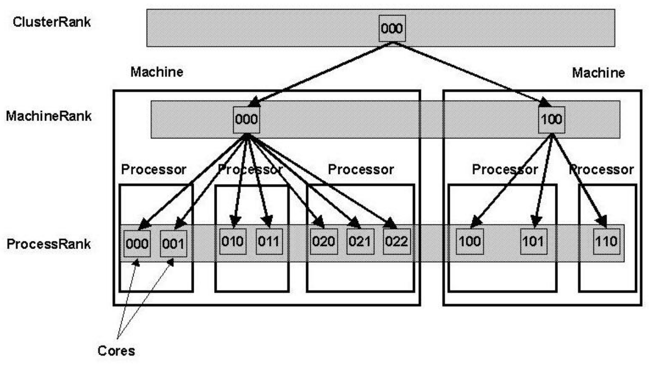


Fig. 7. Hierarchy of the simulator in MPhyScas-P

ii.i) **ClusterRank Process**: it is responsible for the execution of the **Kernel** and to synchronize the beginning and the end of each one of its level's tasks, which requires demands to lower level processes. In a simulation there is only one ClusterRank process (for instance, the process with rank equal to zero in an MPI based system). Figure 8 depicts the relationship between a ClusterRank process and the simulator layers.

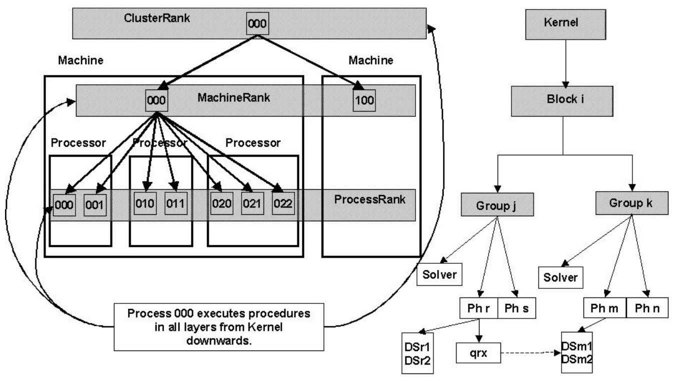


Fig. 8. Layers with procedures executed by clusterRank in MPhyScas-P

ii.ii) **MachineRank Processes**: one process is chosen among all processes running in an individual machine to be its leader. Thus, there is only one MachineRank process per machine. It is responsible for the execution of procedures in the **Block** level and to synchronize the beginning and the end of each one of its level's tasks, which requires demands to lower level processes. ClusterRank is also the MachineRank in its own machine. Figure 9 depicts the relationship between a MachineRank process and the simulator layers.

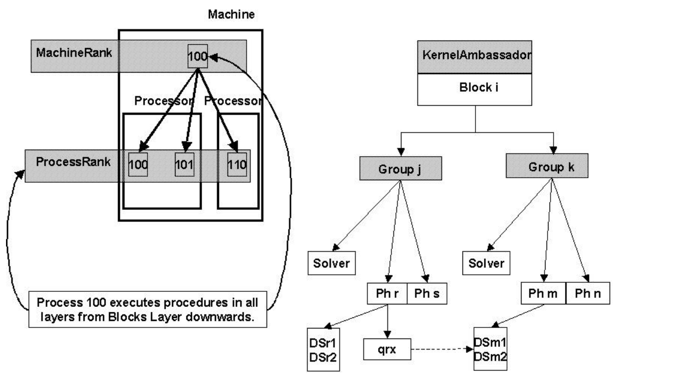


Fig. 9. Layers with procedures executed by machnineRank processes in MPhyScas-P

ii.iii) **ProcessRank Processes**: it is responsible for the execution of the procedures in the **Group** level. The ClusterRank and all MachineRank processes are also ProcessRank processes. Figure 10 depicts the relationship between a ProcessRank process and the simulator layers.

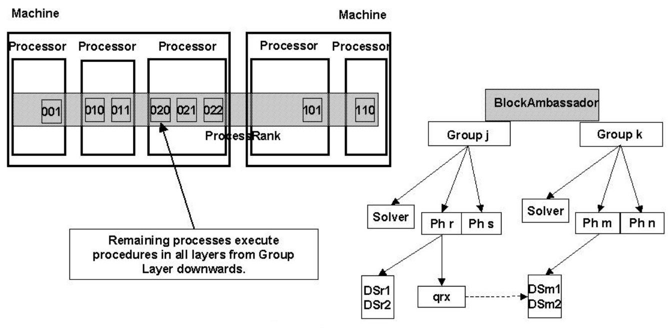


Fig. 10. Layers with procedures executed by processRank processes in MPhyScas-P

Knowing that MPhyScas-S transfer commands from the **Kernel** level down to the **Phenomenon** level in the form of a tree structure, it can be seen that ClusterRank only demands services from all MachineRanks, which only demands services from all of its ProcessRanks. Since the activities in one level returns to the level immediately above after they are accomplished (with the exception of the **Kernel** level) there are natural ways of synchronizing each activity (for instance, using barriers after each demand to the respective lower level has been executed). The heavy computational load is located in the ProcessRank processes. Since all processes are also ProcessRanks and the extra management duties of the leader processes are extremely light, there is no waste of processing power. Furthermore, there is certainly an advantage with the tremendous simplification in the synchronization tasks. Note that the activities in **Group** and **Phenomenon** levels are left for a finer granularity of management. In both levels there are well localized CPU intensive operations, which could be accelerated with a suitable software optimization and the use of hardware devices (for instance, fpga's).

As it has been seen, MPhyScas-P can be considered as MPhyScas-S running in different processes with distributed data. Besides the natural differences between sequential and parallel programs, there is also a specialization of some of the processes, which is important in the synchronization activities. However, when coming to the more demanding parts of the computation, all processes will participate as well. Those parts are coded almost exactly in the same as they are in MPhyScas-S. In what follows we explain the main procedures executed by the preprocessor and by the simulator.

V. Computational Work Load and Flow in MPhyScas-P

In this section we summarize several aspects of the main activities related to the simulator building, the preprocessing of user data and the simulation. In what follows we will describe the following activities: (i) definition and instantiation of the simulator; (ii) Input of simulation data; (iii) Preprocessing; (iv) Simulation execution; (v) Mesh partition and (vi) Visualization.

A. Simulator definition and instantiation

Simulator objects are complex computational entities and are built following a set of user data (actually, meta-data). Simulators in MPhyScas-P architecture do not behave the same in all processes. Therefore, the preprocessing builds simulators able of instantiating different behaviors . Behavior instantiation will be performed depending on the role of each running process, that is, ClusterRank, MachineRank and ProcessRank type processes will behave differently, since they have different management duties. However, in the present implementation, they will perform virtually the same procedures, when it comes to activities at the Group and Phenomenon levels (the most computationally intensive procedures). The definition of a simulator behavior in each process comprises the following activities:

a. ClusterRank (Rank Zero): (i) Interacts with user in order to build/configure the simulator; (ii) Identifies all other processes as either MachineRank or ProcessRank and provides a tag to each one of them; (iii) Format simulator specification data for distribution to each MachineRank process; (iv) Distributes simulator specification data to all MachineRank processes.
b. MachineRanks: (i) Receive simulator specification from ClusterRank; (ii) Format simulator specification data for distribution to its ProcessRanks processes; (iii) Distribute simulator specification data to all its ProcessRanks processes.

c. ProcessRanks: Receive simulator specification from its MachineRank process.
d. All processes: Instantiate simulator (processes from one hierarchical level to another have different simulation instantiation mechanism).

B. Simulation data input

Input of simulation data in MPhyScas-P is exactly the same as for MPhyScas-S (for more information see [4]). However, since processes are specialized - depending on where they are placed in the hierarchy of the simulator - the transfer of simulation data start with the ClusterRank and goes down the hierarchy down to the ProcessRanks. The input procedures are:

a. ClusterRank: (i) Interacts with user in order to input simulation data: (i.1) Geometry; (i.2) Phenomenon types; (i.3) Relation phenomenon $\times$ geometry; (i.4) Quantity to be activated for each phenomenon object; (i.5) Group data;(i.6) Phenomenon data; (i.7) Complementary data for the definition of the preprocessor behavior; (ii) Formats simulation data for distribution to all MachineRanks; (iii) Distributes simulation data to all MachineRanks.
b. MachineRanks: (i) Receive simulation data from ClusterRank; (ii) Format simulation data for distribution to its ProcessRanks; (iii) Distribute simulation data to its ProcessRanks.
c. ProcessRanks: Receive simulation data from its MachineRank.
d. All processes: Instantiate preprocessor object.

C. Preprocessing

Preprocessing is an activity responsible for the building of data structures for the simulation data in a way that can be understood by the simulator. Not only that, of course, because part of the user data is transformed severely, before becoming available for the simulator. Those tasks can be very computationally demanding and can be performed either sequentially - with the result being distributed afterwards - or in parallel. One of such an example is mesh generation. In MPhyScas-P the preprocessing is also especialized, depending on the process type along the hierarchy. In any case, the idea is that the processes in each level will perform part of the preprocessing and will send subsets of raw data together with subsets of already preprocessed data to processes in the lower level. This helps not only load balancing, but also the simplification of procedures.

1) Preprocessing Dynamics: The dynamics of the preprocessing activities can be described through the actions taken at each level of computation:

a. ClusterRank: (i) **If** preprocess is sequential: (i.1) Preprocess whole simulation data including mesh generation and partition; (i.2) Format preprocessed simulation data to all MachineRanks; (i.3) Distribute preprocessed simulation data to all MachineRanks, or **else** (i.1) Preprocess the whole simulation data in parallel with all other processes (communication with other processes depends on the methods used, i.e., mesh generation)
b. MachineRanks: (i) **If** preprocess is sequential: (i.1) Receive preprocessed simulation data from ClusterRank; (i.2) Preprocess a small part of its simulation data; (i.3) Format preprocessed simulation data for distribution to its ProcessRanks; (i.4) Distribute preprocessed simulation data to all its ProcessRanks, or **else** (i.1) Preprocess the whole simulation data in parallel
c. ProcessRanks: (i) **If** preprocess is sequential: (i.1) Receive preprocessed simulation data from rank machine; (i.2) Preprocess a small part of its simulation data; or **else** (i.1) Preprocess the whole simulation data in parallel
d. Notes: (i) The Preprocessor object is actually a very complex machine. It encapsulates a great variety of other objects, which were instantiated following data (choices) given by the user; (ii) This object is specialized depending whether the node is a ClusterRank or a MachineRank or a ProcessRank; (iii) It is not the intention of this paper to go into details about the preprocessing stage. Nevertheless, short explanations about mesh generation and distribution will be needed; (iv) A third party mesh generation in parallel for MPhyScas should require that: (iv.1) ClusterRank starts the process and distributes data to be performed in parallel with all MachineRanks; (iv.2) Then all MachineRanks will process the data a little bit more and then redistribute them to all ProcessRanks; (iv.3) Since ClusterRank and all MachineRanks are also ProcessRanks, the heaviest work will be done after the data is spread among all processes; (v) When the mesh generation is sequential, only ClusterRank executes the mesh generator and then makes the partition and distribution of the mesh; (vi) Being able of using a third party mesh generator is also a requirement for MPhyScas-P. Thus, it is wrapped inside an object, which is also responsible to transfer data in and out of the mesh generator.

2) Preprocessing activities: The following activities comprise the main activities in the preprocessing. For clarity purposes, we assume that the mesh generation is done sequentially by the ClusterRank:

a. ClusterRank: (i) Instantiate Phenomenon objects; (ii) For each Phenomenon object: (ii.1) Build GeomGraph; (ii.2) Build PhenGraph; (ii.3) Activate quantities; (ii.4) Build relationship Phenomenon $\times$ Phenomenon based on coupled quantities; (ii.5) Establish mesh sharing relationship; (ii.6) Instantiate methods; (iii) Relate Phenomenon objects with simulator Groups; (iv) For each Group: (iv.1) Build GroupTask objects and load their data; (iv.2) Build QuantityGroup objects with their GroupTask objects; (iv.3) Instantiate methods; (v) Generate geometric meshes; (vi) Generate phenomenon meshes for each Phenomenon; (vii) Partition each one of the geometric meshes and respective phenomenon meshes among MachineRank processes; (viii) Partition GeomGraphs following geometric mesh partition; (ix) Partition PhenGraphs following the partition of the respective GeomGraphs; (x) Build Phenomenon objects for each partition; (xi) Format data (Group data and Phenomenon data for each partition) to be sent to the MachineRanks processes; (xii) Distribute preprocessed data to MachineRanks processes.

b. MachineRank: (i) receive data from ClusterRank; (ii) Instantiate Phenomenon objects; (iii) For each Phenomenon object: (iii.1) Build GeomGraph; (iii.2) Build PhenGraph; (iii.3) Activate quantities; (iii.4) Build relationship Phenomenon $\times$ Phenomenon based on coupled quantities; (iii.5) Establish mesh sharing relationship; (iii.6) Instantiate methods; (iv) Relate Phenomenon objects with simulator Groups; (v) For each Group: (v.1) Build GroupTask objects and load their data; (v.2) Build QuantityGroup objects with their GroupTask objects; (v.3) Instantiate methods; (vi) Recover geometric meshes; (vii) Recover phenomenon meshes for each Phenomenon; (viii) Partition each one of the geometric meshes and respective phenomenon meshes among its ProcessRank processes; (ix) Partition GeomGraphs following geometric mesh partition; (x) Partition PhenGraphs following the partition of the respective GeomGraphs; (xi) Build Phenomenon objects for each partition; (xii) Format data (Group data and Phenomenon data for each partition) to be sent to its ProcessRank processes; (xiii) Distribute preprocessed data to its ProcessRank processes

c. ProcessRank: (i) receive data from its MachineRank; (ii) Instantiate Phenomenon objects; (iii) For each Phenomenon object: (iii.1) Build GeomGraph; (iii.2) Build PhenGraph; (iii.3) Activate quantities;(iii.4) Build relationship Phenomenon $\times$ Phenomenon based on coupled quantities; (iii.5) Establish mesh sharing relationship; (iii.6) Instantiate methods; (iv) Relate Phenomenon objects with simulator Groups; (v) For each Group (v.1) Build GroupTask objects and load their data; (v.2) Build QuantityGroup objects with their GroupTask objects; (v.3) Instantiate methods; (vi) Recover geometric meshes; (vii) Recover phenomenon meshes for each Phenomenon; (viii) Build Phenomenon objects

D. Simulation execution

The execution of the simulation requires synchronization in all levels of the hierarchy. We will not describe this mechanism in detail. However, it is important to mention that the execution of tasks at a given level, which requires tasks to be executed by other processes in the lower level, is used as a synchronization point for all processes involved. The main procedures can be viewed below:

a. ClusterRank: (i) Interacts with the user in order to start simulation; (ii) Starts simulation by executing the Kernel driver (it is an object): (ii.1) Whenever the Kernel driver calls the execution of a procedure at the Block level, it should broadcast a message with the needed data to all MachineRanks; (ii.2) Execute its own Block level as requested by the Kernel driver (ClusterRank acts as a MachineRank); (ii.3) Upon the end of the execution of the procedure in the Block level, ClusterRank broadcasts a message to all MachineRanks for synchronization purposes.

b. MachineRanks: (i) Receive message from ClusterRank to execute a procedure in the Block level; (ii) Execute the required procedure; (iii) Whenever a procedure in a Block object demands the execution of a procedure - operation of type BLAS I, II or III or the execution of a GroupQuantity object - at the Group level, it should broadcast a message with the needed data to all its ProcessRanks; (iv) Upon the end of the execution of the procedure in the Group level, MachineRank broadcasts a message to all ProcessRanks for synchronization purposes; (v) At the end of the procedure, MachineRank sends a message answering the synchronization broadcast sent by ClusterRank

c. ProcessRanks: (i) Receive message from its MachineRank to execute a procedure in the Group level; (ii) Execute the required procedure; (iii) At the end of the procedure, ProcessRank sends a message answering the synchronization broadcast sent by its MachineRank.

d. Notes: It is noticeable that the described hierarchical execution in parallel allows also for parallelization schemes of type MPMD (multiple processes multiple data), because it links components in a DAG (direct acyclic graph) structure. The DAG structure allows for automatic and dynamic analysis, load balancing, algorithm partitioning and scheduling of the execution of all its parts on a given set of processors. This is currently being pursued and will be published elsewhere.

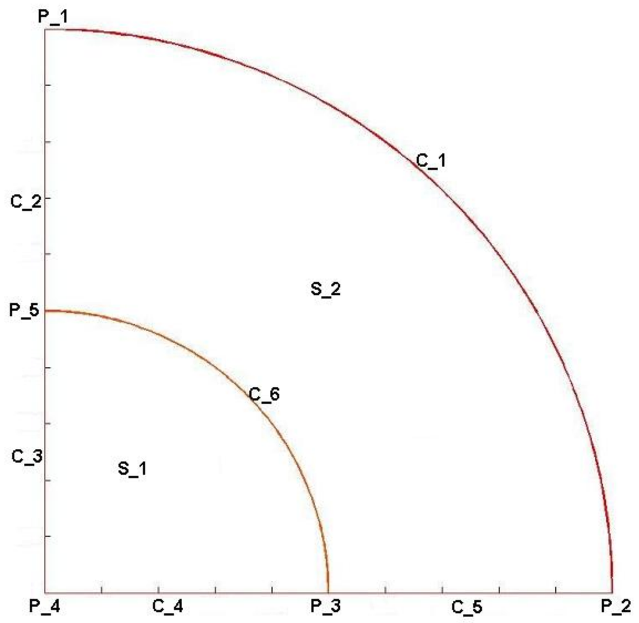


Fig. 11. Geometric domain

E. Further notes

In this subsection we present some notes to further clarify some issues.

1) Mesh partition and distribution: For what follows, consider the geometric domain in Figure 11 and its geometric

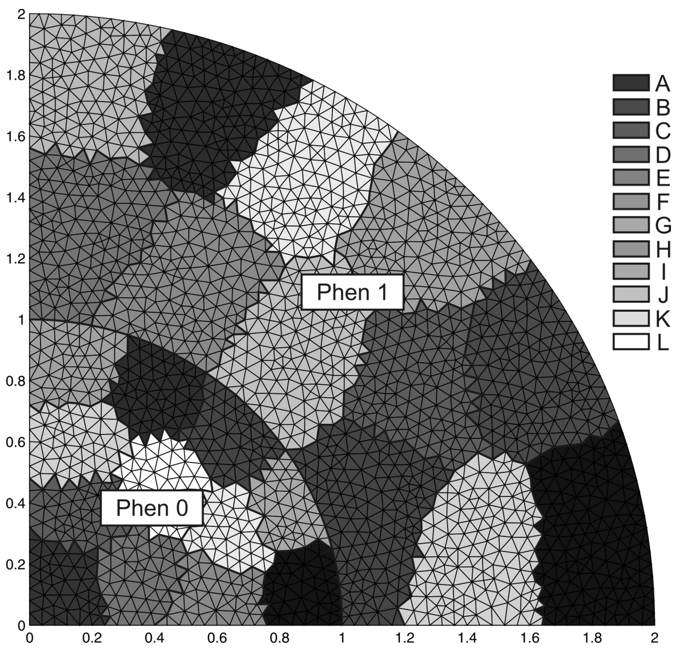


Fig. 12. Partitioned geometric mesh

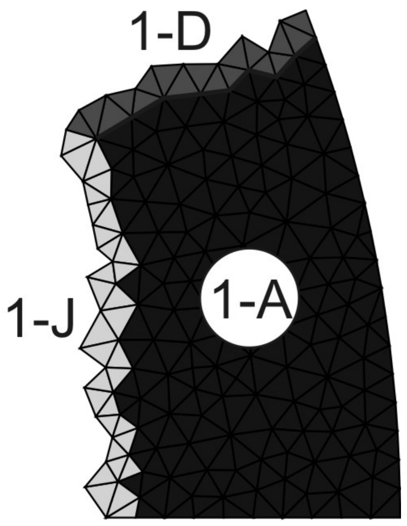


Fig. 13. Partition component in contact with the outer boundary

mesh 12, which is partitioned for twelve processes. Assume that phenomena Ph_1 and Ph_2 act on S_1 and S_2, respectively. Note that both domains were partitioned into twelve parts (processes i, i = A, . . ., L) in order for each process to contain both Phenomenon objects with nonempty meshes.

a. MPhyScas architecture associate procedures (quantity computations and other tasks) to geometric entities [15], [4]. They are performed at the Phenomenon level, but their execution and required parameters are established at the Group level. Those procedures and the geometry are then organized in the form of two graphs, the GeomGraph and the PhenGraph, which have the same structure. However, while the GeomGraph encapsulates a geometric entity (points, curves, surfaces, volumes) at each one of its nodes (GeomGraphNodes), the PhenGraph encapsulates the procedures (called WeakForm) at each one of its nodes (PhenGraphNodes), which are to be computed on the geometric entities of the respective GeomGraphNodes.
b. The partition of the mesh represents a partition of the geometry and thus requires an associated graph partition. The current geometric entities will then be partitioned - after their mesh partition - and new geometric entities will be formed (including those on the mesh interface between two mesh parts), see Figures 13 and 14. Note

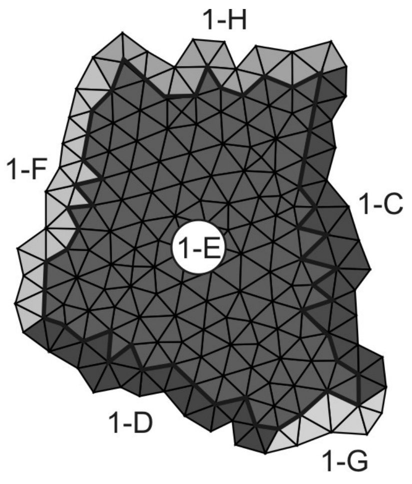


Fig. 14. Partition component without contact with the outer boundary

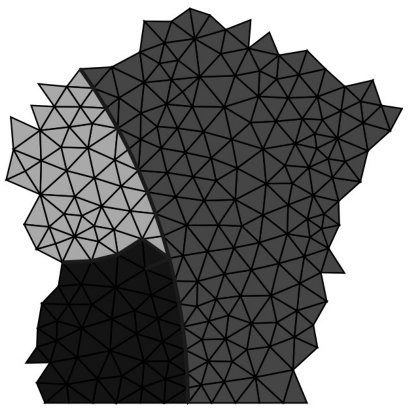


Fig. 15. Neighboring partition components, such that the interface is an existent geometric entity

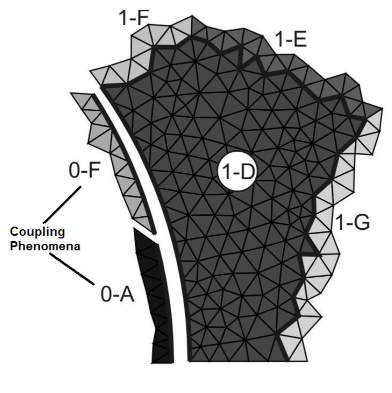


Fig. 16. Added CouplingPhenomenon to the process on one side

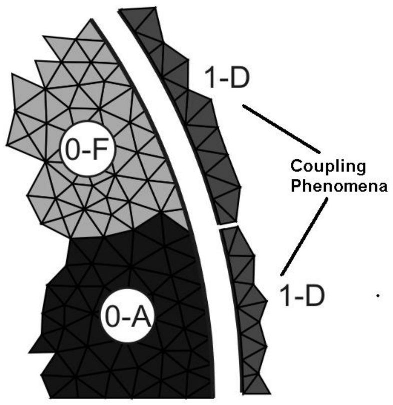


Fig. 17. Added CouplingPhenomena to the process on the other side

that ghost elements are always included in the parts of a partition.

c. The partition of already existent geometric entities will produce new geometric entities (Figure 13), which will inherit all WeakForms from the former. However, the new geometric entities obtained at the interface between two partitions will be given new WeakForms, depending on the solution algorithm implemented in the simulator. Nevertheless, the vector fields restricted to those brand new geometric entities will represent the data to be exchanged between the neighboring processes.

d. Following item (c) above, new GeomGraphs and PhenGraphs will be generated for each partition. It is important that a process P can retrieve the id's of its neighboring processes and the connectivity of the mesh nodes at each interface between P's mesh and its neighboring mesh parts. Those pieces of information can also be localized at the PhenGraphNode associated to the GeomGraphNode, which contains the geometric interface between two mesh parts. Two processes are neighbors if their geometric meshes have a nontrivial intersection.

e. It may happen that an interface between two mesh parts is also a part of an existent geometric entity at the contour of the geometric domain, Figure 15. Suppose that this geometric entity divides the geometry into two regions, where one phenomenon acts on one side and a different one acts on the other (for instance, part A and part D in Figure 15 and 16). In this case, if there is exchange of data between both phenomena during simulation, then, special procedures should take place. This is so because one cannot afford the transfer of coupling data (vector fields and Phenomenon meshes) across processes each time one Phenomenon object needs data from other Phenomenon on the boundary of its geometric domain.

f. In order to tackle the problem described in item (e) above, consider that both meshes were partitioned among all processors in such a way that each process has all Phenomenon objects with nonempty mesh parts. Suppose now that Ph_0 computes a quantity q_a on the interface between S_1 and S_2 (that is, curve C_6), where it needs data from phenomenon Ph_1. Consider process A, which contains two mesh partitions, $A-0$, for Ph_0 and $1-A$ for Ph_1. Consider now process D, which contains two mesh partitions, $0-D$, for Ph_0 and $1-D$ for Ph_1. Note that the interface between $0-A$ and $1-D$ coincides with a part of the curve C_6. Thus, the computation of q_a by process A will need information from Ph_1, which is not in process A (note that $1-A$ is far from that curve). The solution is, then, to add to process A another Phenomenon object called CouplingPhen (in the same Group as Ph_0), containing the copy of the geometric entity, where the coupling occur (interface $0-A$-$1-D$), together with its mesh and related vector fields, Figure 17. Also, this object should know process D's id and handler to its locale in Ph_1 in D. In this way, whenever required, the CouplingPhen object will update - through communication between both processes - only the vector field data from process D related to the geometric part $1-D$. Mesh data (geometric and phenomenon meshes) is already local to process A and need not be transferred.

g. The structure of CouplingPhen is exactly the same as a regular Phenomenon object. The difference is that the instantiation of CouplingPhen is dynamic and does not need data from the user. Also, the coupling information needed in the computation of a coupled quantity (like q_a in the above example) is automatically built from the original information (provided by the user), which linked the computation of q_a by Ph_0 with data from Ph_1 on the curve C_6.

h. Note that geometric interfaces between two parts can be either a point, or a mesh curve, or a mesh surface. This is so because those entities can be shared by more than one partition. Actually, points (in 2-D and 3-D) and mesh curves (in 3-D) can be interfaces between many partitions at the same time. This makes the above story a little more demanding, but we will not go into further details. The main picture is already set.

i. After the mesh partition and distribution is finished, the preprocessing procedures will generate the new GeomGraphs and PhenGraphs for each process. Then, all CouplingPhen objects are instantiated. That is the last thing the preprocessor will do before the simulation. The simulator automatically schedules the updating requests to CouplingPhen objects.

j. Mesh partition and distribution are far more complex when two or more Phenomena objects - defined on the same geometric entity - do not share the same geometric mesh. It is when data transfer between meshes should take place. This case is quite important, but will not be considered here.

2) Visualization and other types of external interfaces: MPhyScas does not provide costume-made visualization machinery for simulation data and results. However, it does provide interfaces to third party visualization software. An interface should be implemented for each new visualization software to be used. Also, the post-processing procedures are implemented as Phenomenon objects, that is, all calculations and format exchange of quantities to be visualized/analyzed are implemented as coupled WeakForm's in Phenomenon objects. The execution of the visualization software can be done through MPhyScas interface, although it can also be done separately. Not only visualization events can be considered. Phenomenon objects can also encapsulate (through their WeakForm's) a variety of different types of interferences in the simulation. For instance, interruptions and coupling between simulation and laboratory experiments can be implemented with this strategy. The simulation algorithm will dictate when and where in the simulation those interferences will become active.

VI. Conclusions

We presented the architecture of MPhyScas-P, a framework aimed at supporting the automatic development of high performance simulators for multi-physics problems. This architecture inherits from MPhyScas-S (the sequential version) all its workflow representation, with the obvious difference that MPhyScas-P is distributed in a hierarchical way. Although MPhyScas-S has already a fully functional prototype, MPhyScas-P has a prototype (using MPI) currently being tested. Besides those qualities that MPhyScas-S has already demonstrated (strong reusability, maintainability, adaptability and correctness), MPhyScas-P provides also another nice feature: due to the DAG (direct acyclic graph) structure of its workflow, its code can be dynamically analyzed and reconfigured in such a way that MPMD schemes could be used. At last but not least, it is important to notice that we are dealing with very complex problems, with very complex solution algorithms. We think that if MPhyScas-P would be able to alleviate the burden of programming and changing code for those types of problem, our task would be fulfilled. We are currently building a graphic user interface coupled to a DBMS in order to manage the use of components for MPhyScas-S and MPhyScas-P and are planning in using an interface description language in order to describe all interfaces of components. One candidate being considered is SIDL from the Common Component Architecture (CCA) [13].

References

[1] F. C. G. Santos, E. R. R. J. Brito, and J. M. A. Barbosa, "Simulao do problema de evoluo do dano em uma barra elasto-viscoplstica com acoplamento termomecnico empregando grafo de interface genrica (gig)," 7° *Congresso Iberoamericano de Engenharia Mecnica*, 2005.

[2] F. C. G. Santos, E. R. R. J. Brito, and J. M. A. Barbosa, "Coping with data dependence and sharing in the simulatin of coupled phenomena," *International Congress on Computational and Applied Mathematics - ICCAM*, 2006.

[3] F. Santos, M. Lencastre, and M. Vieira, "Workflow for simulators based on finite element method," *Proceedings of the International Conference on Computational Science (ICCS)*, 2003.

[4] F. Santos, E. R. R. J. Brito, J. M. A. Barbosa, J. M. B. Silva, and I. H. F. Santos, "Toward the automatic development of simulators for multi-physics problems," *International Journal of Modelling and Simulation for the Petroleum Industry*, vol. 1, no. 1, 2007.

[5] M. Lencastre, *Conceptualisation of an Environment for the Development of FEM Simulators.* Doutorado em ciências da computação, Universidade Federal de Pernambuco, Recife, Pernambuco, 2004.

[6] F. Santos, M. Lencastre, and I. Rodrigues, "Fem simulator based on skeletons for coupled phenomena," *Proceedings of the 2nd Latin American Conference on Pattern Languages of Programming*, 2002.

[7] F. Santos and M. Lencastre, "An approach for fem simulator development," *Journal of Computational ans Applied Mathematics*, 2006.

[8] R. Niekamp, "Software component architecture," *http://congress.cimne.upc.es/cfsi/frontal/doc/ppt/11.pdf.*

[9] *http://www.comsol.com/.*

[10] S. Parker, "A component-based architecture for parallel multi-physics pde simulation," *Future Generation Computer Systems*, no. 22, p. 204216, 2006.

[11] A. G. L. G. M. J. R. T. S. E. S. J. Goodale, T., "The cactus framework and toolkit: Design and applications," *Vector and Parallel Processing - VECPAR '2002, 5th International Conference, Springer*, 2003.

[12] H. C. Edwards, "Sierra framework version 3: Core services theory and design," *Sandia National Laboratory, report SAND2002-3616*, November 2002.

[13] "Common component architecture home page," *http://www.cca-forum.org/.*

[14] T. S. G. M. Graybill, B., "Hpc application software consortium summit - concept paper," *http://www.cct.lsu.edu/ gallen/Reports/HPCASC_March2007.pdf*, March 25-26, 2008.

[15] F. C. G. Santos, E. R. R. J. Brito, and J. M. A. Barbosa, "Phenomenon computational coupling relationship between phenomena on multi-physics simulation," *20th European Conference on Modelling and and Simulation*, 2006.

DISTRIBUTED REAL-TIME RAILWAY SIMULATOR

Mihai Hulea, Camelia Avram, Tiberiu Letia, Dana Muresan, Sergiu Radu
Technical University of Cluj-Napoca, C. Daicoviciu street, nr. 15, Cluj-Napoca, Romania
{mihai.hulea, camelia.avram, tiberiu.letia}@aut.utcluj.ro

KEYWORDS
Railway traffic, real time simulation, distributed systems.

ABSTRACT

The problem of control and management of railway transportation is a complex task with major outcomes in a modern society. Trains are suitable for transporting peoples and goods with a good trade-off between cost and rapidity. A railway system consists of a network of tracks, list of stations, safety devices (signals, sensors, etc) and a set of trains. Trains moves from one station to other along the network. Distributed applications are applicable in case of large systems.

INTRODUCTION

Railways are perceived as the most efficient means of mass passenger transportation. In the case of land based freight movements, railways are usually at a competitive advantage relative to road transport for the non-urban medium to long distances, bulk and containerized tasks. The operations of railway systems involve multi-disciplinary practices, ranging from business to transport operations and engineering (Ho et al 2004).
Many railway systems are still state-owned, but the privatization of various extents has been going on in many countries (Zimmermann et al 2003). The newly evolved private companies assume different roles within the operational chart of a railway system. Indeed, more than one company may take on the same role and compete with each other, which is one of the supposed advantages of privatization. A number of non-privatized railway lines are also contemplating decentralization in some way so that local authorities or contracting companies are running the rail services.
As a result, there are many parties, such as track owners and service providers, working together (collaborating and/or competing) to provide such services that the overall business and engineering objectives are considered and balanced, as well as fulfilling their own interests and duties. In order to study the behavior of this system with multiple, interactive and autonomous parties, agent-oriented technology offers the framework for modeling. Each party is represented by an independent agent who is equipped with its objectives, intelligence and autonomy (Ho et al 2004; Cuppari et al 1999; Faber et al 2006).

The increase demand of short-term train schedules by Transport Operators highlights the necessity of automated tools for train traffic decision support. When the number of trains running on a railway line and the availability of tracks are known on day by day basis, decision support systems can help in maximizing the demand granting and optimizing the traffic flow. (Cuppari et al 1999)
The issue in railway simulation is to let trains move through the network based on their time table and to check if deadlocks appears. Also the simulator can be used to check the performances of different dynamic control algorithms before implement them in the real field. The simulators can be used also for training the railway operators.
As stated in (Ferschea 2005) the fundamental approaches for simulating systems are continuous vs. discrete; event and time driven vs. event driven. In continuous event simulation the state change continuously in time, while in a discrete event simulation the events happen instantly at a fixed moment in time. The proposed approach of this paper is a time driven simulation in which the simulator update the state of the system at fixed point in time based on a given pulse clock. For solving very large railway simulation problems a distributed architecture approach must be considered. The main reasons for this are: scalability, performance and reliability.
The paper proposes a distributed time driven generic simulation system suitable for testing various routing and planning algorithms. By generic is meant that the simulator is able to take a configuration as input in form of the XML structures therefore not being tied by a static railway structure. The simulator updates time in discrete steps. One basic requirement needed to be fulfilled by the time update method is to synchronize it's time update interval with other simulator instances. It should not be possible for one simulator to progress faster than other simulators. Taking into account the discrete nature of the proposed simulator, each entity in the system should update his state on each time step. Tow possible approaches can be identified to accomplish this. This first approach is a parallel update where all entities update states concurrently. With this method extra care should be taken in order to synchronize updating threads and to prevent data inconsistency. The second approach is a sequential method where all entities are updated one after another. The proposed simulation architecture uses the sequential method.
The simulator provides a structure communication module through which other applications (like

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

controller, monitoring and information and advisory system) can send commands to the railway objects (switches, signals, etc.) and read the current state of the different railway objects (sensors, signals, trains, etc). The system also integrates a GPS emulator which can be queried in order to obtain current train coordinates and speed. The simulator instances communicate through a communication and coordination module.
The issue of railway simulation is to virtually let trains run through the network and to check whether the timetable is verified or stable (H. Schlenker 2005).

STATE OF THE ART

Modeling behavioral characteristics is a key issue of Rail way Intelligent Safety Guarantee System (RISGS) presented in (Cai et al 2006). A multi-agent simulation system in RISGS is considered and the proposed model is evaluated in a simulated experiment. This model incorporates the philosophy of object oriented concepts, available Petri-nets analysis tools and multi-agent techniques. The analysis results indicate this method can provide effective judgments without deadlocks, and improve complicated characteristics description about intelligence, autonomy and security.
Using parallel processing reduce the computing time and therefore is suitable for building real-time simulators and present different issues related to solving a power distribution system with parallel computing based on a multiple-CPU server and they concentrate, in particular, on the speed-up performance (Fun et al 2000). The model presented in [Bon, 2000] was initially based on the track analyzer method developed by Volpe National Transportation Systems Center, but was significantly enhanced in order to provide accurate predictions over a wider variety of vehicle behavior and to fulfill the real time constraints.
Several models are described in literature and some programming constraints for the routing and scheduling of trains running through a junction are taken into account (Rodriguez 2005). The model uses input data from relevant time events of train runs calculated by a simulator. The model can be integrated into a decision support system used by operators who make decisions to change train routes or orders to avoid conflicts and delays.
Hybrid Petri nets can be used to simulate the traffic and to evaluate the performance of the control system. (Kaakai et al 2007) propose a simulation model based on hybrid Petri nets able to carry out performance evaluation procedures in order to increase the traffic safety.

DESIGN OF THE SIMULATOR

Railway structure modeling

XML Schema is used to describe the structure of the document containing the railway network structure. In this section the elements used to describe the network are presented.
The root of the XML document is the *railmap* element which contains all other elements which define a railway network.
The following types of railway elements are defined: gate, connectionGate, segment, signal, switch and sensor. Each element has a unique id and is stored in the element ID attribute. Also all elements have tow attributes (posX and posY) which store the location of the elements and are used for displaying them inside the simulator graphical user interface.

A gate represents a connection point between tow segments. A special type of *gate* is a *connectionGate* which is used to define connectivity between segments located in the networks managed by different simulators. This type of gates are located at the border of the railway network, and when the train pass a *connectionGate* it will be transferred to the remote network in a position which correspond to the remote *connectionGate* identified by the local *connectionGate*.
A segment represents a railway track and is delimited by tow gates named *gate1* and *gate2*. A convention is maid that the moving direction from *gate1* to *gate2* is the positive direction and is coded with the value 1 while moving from *gate2* to *gate1* is considered negative direction and is noted with -1.
The railway switch is modeled by *switch* element. A *switch* element is composed of a set of gates and has at a moment of time one active connection which allows a train to pass from one segment to another segment. The switch can be commuted in order to change the connected segments.

The *signal* element models a railway semaphore. The signals can have 2 possible states: green state and red state. Signals are attached to segments and are identified by the gate where are located and by the direction from which they can be seen.
The *sensor* element models a presence sensor which can be installed on any position on the road network. The sensor is activated when the train passes over it.
Based on the presented XSD schema, the XML files are created representing various railway network structures. When a simulator instance is launched it will load a railway network located in a XML file.

System Architecture

The paper proposes a distributed simulator architecture in which each station in a road network is handled by one simulator. The simulators are interconnected and communicate between each others. The UML component diagrams in Figure 1 presents the general simulator software architecture.

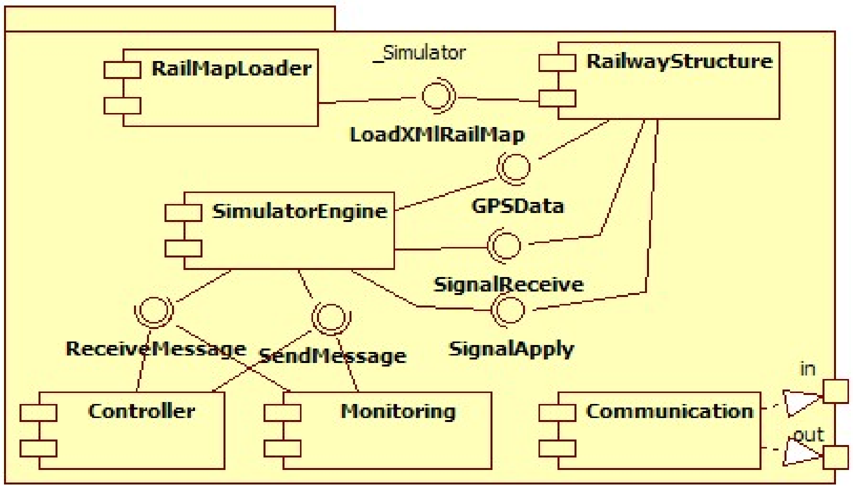


Figure 1: Simulator general software architecture.

The railway network structure is loaded from an XML file by the configuration module. The core of the configuration module is the RailMapLoader which uses a XML parser for interpreting the structure XML file and creating a structure of objects representing the railway network. The UML diagram presented in the Figure 2 describes the structure of objects used to model the railway network.

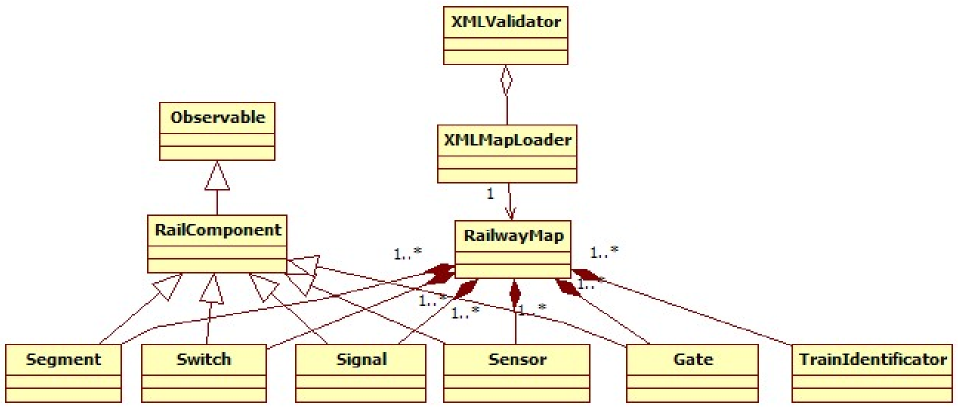


Figure 2: Railway structure modeling classes

The RailwayMap class is a container class which stores the structure of the railway network and the current state. The simulator engine accesses the internal data and alters the state of the network through a set of function defined in the following modules: StructureQuer, DataTransformtion, SingalReceive and SignalApply.

The communication between simulator instances is realized through CommServer and CommClient modules which use TCP\IP protocol in order to exchange messages. The serialization mechanism is used for sending messages. When a train reach the border of the network controlled by a simulator it will be sent to the corresponding remote simulator as stated in the network structure connection rules (this rules are set in the connectionGate elements presented in the previous section).

Moving trains and controlling structure states inside the network are accomplished by the simulator engine using the TrainMove and CollisionDetection modules. On each step the next position of the trains will be calculated taking into account the following aspects: the dynamic train characteristics and the current status of the railway objects (the state of signals, semaphores and switches). The CollisionDetection implements an algorithm for detecting collision between trains which are moving on the same segment.

The communication model

The communications between distributed simulators are implemented at tow levels. At the first level it is implemented a communication layer between simulator engines. When a train leaves a zone controlled by a simulator it will be sent through the simulator engine connection to the corresponding neighbor simulator. At the second level it is implemented a communication layer between controllers. At this level messages are exchanged between controllers in order to make path reservations. In Figure 3 the communication between simulators is presented.

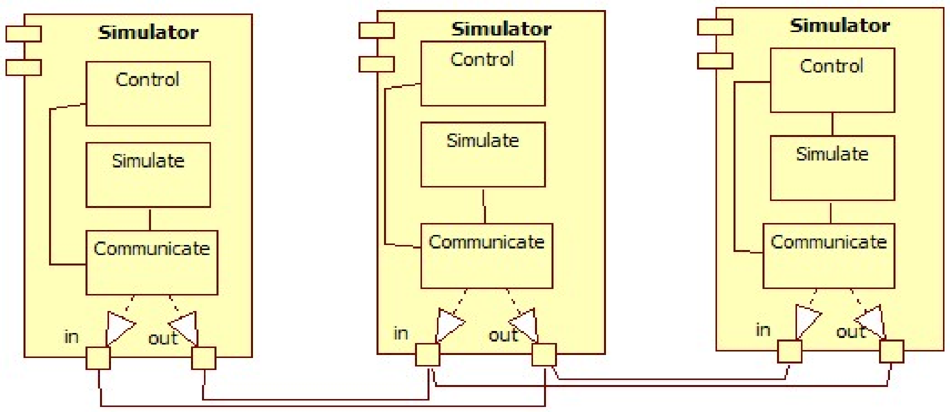


Figure 3. Simulators communication.

In this paper an asynchronous message passing model for implementing the distributed communication mechanism is proposed. This approach as been used both for implementing the communication at the simulator engine level but also at the controller level. The components will exchanges messages in an asynchronous manner.

In order to model task and threads the following stereotypes have been defined:

- <<PeriodicThread>>, which represent activities which are executed with a given time period;
- <SporadicThread>> , which represents activities which are executed each time an event occurs;

For communication between threads the following stereotypes have been defined:

- <<PriorityQueue>>, this is an unbounded queue in which elements are ordered based on an priority attribute associated with each element;
- <<Buffer>>, this is a simple buffer on which data are ordered based on the FIFO rule;

Based on the stereotypes defined before, the UML object communication diagram for the simulator system is presented in Figure 4.

The messages are passed between threads using unbounded buffers and queues. For send commands to the railway structure elements, and for receiving data about the state of the railway structure elements FIFOBuffers are used. The PriorityQyeye queue are used for exchanging messages between working threads (simulator SimulatorEngine and Controller) and communication threads (Sender and Receiver). Messages between distributed controllers are exchanged using CSender and CRecevier threads. When a new message must be sent, it must be added in the COutQueue queue or the corresponding SOutQueue, from where is get by the CSender or SSender and is sent to the destination. When a new message is received by SReceiver threads or the corresponding CReceiver, it is put in the CInQueue or SInQueue queue, from where is read by the Controller or Simulator Engine thread.

Time modeling

The method chosen to update time in the simulators is to use discrete time steps. This means that time is not updated continuously but in blocks of a certain interval. The time updates are executed by calling a tick method in the simulator engine. Each entity new state is calculated based on (e.g. a train updates its position, velocity and acceleration) the previous state and the time passed since last tick. The component responsible for calling the tick method is the Timer.

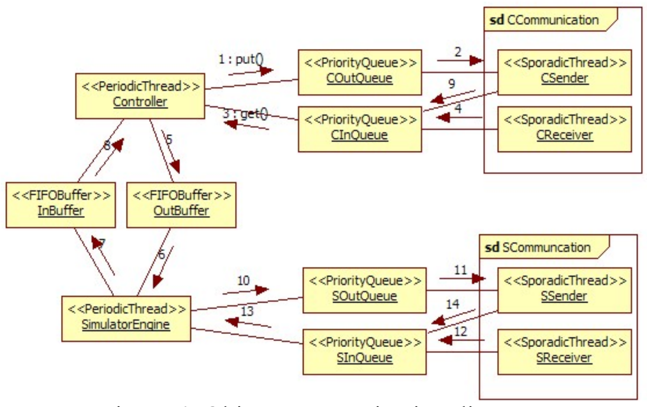

sd CCommunication
1 : put()
<<PriorityQueue>> COutQueue
2
<<SporadicThread>> CSender
<<PeriodicThread>> Controller
9
4
3 : get()
<<PriorityQueue>> CInQueue
<<SporadicThread>> CReceiver
8
5
<<FIFOBuffer>> InBuffer
<<FIFOBuffer>> OutBuffer
7
6
sd SCommunication
10
11
<<PriorityQueue>> SOutQueue
<<SporadicThread>> SSender
13
14
12
<<PeriodicThread>> SimulatorEngine
<<PriorityQueue>> SInQueue
<<SporadicThread>> SReceiver

Figure 4. Object communication diagram.

In order to synchronize time of tow or more simulator instance, a time server coordinate all the simulators timers and notify each simulator timer when the simulator must advance at the next step.

SIMULATOR IMPLEMENTATION

The simulator application has been implemented in Java language using JDK v1.6.0. The implementation has been tested on Windows XP and Linux RedHat platforms.

In Figure 5 a screenshot of the simulator graphical user interface is presented. The railway network managed by the simulator is presented into a window. We are using color codes to represent states of structure elements. For example a free segment is drawn in green while a segment on which is at least one train is drawn in red. A manual control interface is available to the operator from which he can change the state of structure elements, and also can control trains.
The screenshot in Figure 6 represents the structure view window. In this window the details of the simulated structure are presented to the operator. For each structure element a separate table is displayed containing the attributes values.

The simulator architecture has been designed with the goal to be used for testing various controls and routing algorithms. In order to provide this functionality the simulator implements tow interfaces through which other modules like controller, monitoring or information systems can have access to the simulation data or can interact with the simulated railway structures.

CONCLUSIONS

This paper highlights the benefits of applying distributed techniques to rail transportation system modeling.
The proposed simulator enables the handling of more complex problems than the existing technology can handle. In the case of railroads, this might include routing more trains over more tracks, whereas traditional movement planning systems are able to plan the movement of only one train at a time. Another advantage of this simulator is that it enables more rapid adaptation to alternative schedules because of changes in the environment (e.g., a track blocked by an accident) and does not require a total reassessment.
The proposed system architecture is distributed since several train stations are connected and on each of it a simulator railway is started.
The communication will be realize on each layer (control, monitoring, information, simulation) between two simulator railways and between different layers inside of a simulator railway.
Using a good scheduling algorithm the traffic can be increase and to test the result is better to use a proper distributed simulator.

REFERENCES

Bernaer, S.; E. Burke; P. De Causmaecker; G. Vanden Berghe; and T. Vermeulen, 2006. "A Multi Agent System to Control Complexity in Multi Modal Transport", *The IEEE Simulation Tran*.

Bonaventura, C.S.; J.W. Palese; and A.M. Zarembki, 2000. "Intelligent system for real-time prediction of railway vehicle response to the interaction with track geometry", *Railroad Conference, Proceedings of the 2000 ASME/IEEE* Joint Volume, Issue, 2000 Page(s):31 – 45.

Cai, G.; Z. Zhang; L. Jia; and Y.Ye, 2006. "A Multi-Agent Model of Railway Intelligent Safety Guarantee System, (PRC)", *From Proceeding (523) Computational Intelligence*.

Cuppari, A.; P.L. Guida; M. Martelli; V. Mascardi; and F. Zini. 1999. "An Agent Based Prototype for Freight Trains Traffic Management", *Proc. of FMERail Workshop*.

Faber J; and R. Meyer. 2006. "Model Cheking Data-Dependent Real-Time Properties of the European Train Control System", *Proceedings of the Formal Methods in Computer Aided Design.*

Ferschea, A. 2005. "Parrallel and distributed simulation of discrete events systems", *Handbook of Parallel and Distributed Computing.* McGraw-Hill.

Fung, Y.F.; T.K. Ho; and W.L. Cheung. 2000. "Real-time simulation for power systems based on parallel computing-an empirical study", *International Conference on Advances in Power System Control,* Operation and Management, Volume 2.

Gambardella, L.M.; A.E. Rizzoli. 2005. "Agent-based Planning and Simulation of Combined Rail/Road Transport", *Mathematical and Computer Modelling*.

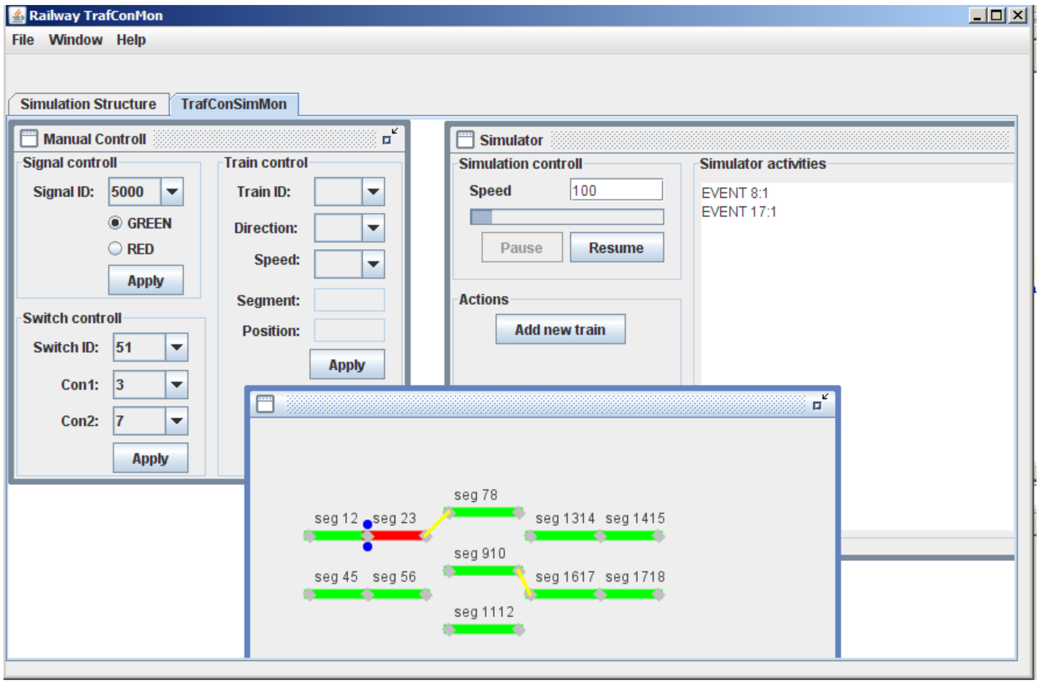
Ho, T.K.; L. Ferreira; and K.H. Law. 2004. "Agent applications in rail transportation", *Proc of International Conference on Intelligent Agents Web Technologies and Internet Commerce*, pp. 251-260, Vienna.

Kaakai, F.; S. Hayat; and A. El Moudni. 2007. "A hybrid Petri nets-based simulation model for evaluating the design of railway transit stations", *Simulation Modeling Practice and Theory*, Science Direct.

Rodriguez, J. 2005. "A constraint programming model for real-time train scheduling at junctions", *Transportation Research Part B 41* pp 231–245, Science Direct.

Schlenker, H. 2005. "Distributed Constraint Based Railway Simulation", Springer Verlag Berlin Heidelberg 2005.

Zimmermann A.; and G. Hommel. 2003. "A Train Control System Case Study in Model-Based Real Time System Design", *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03).*



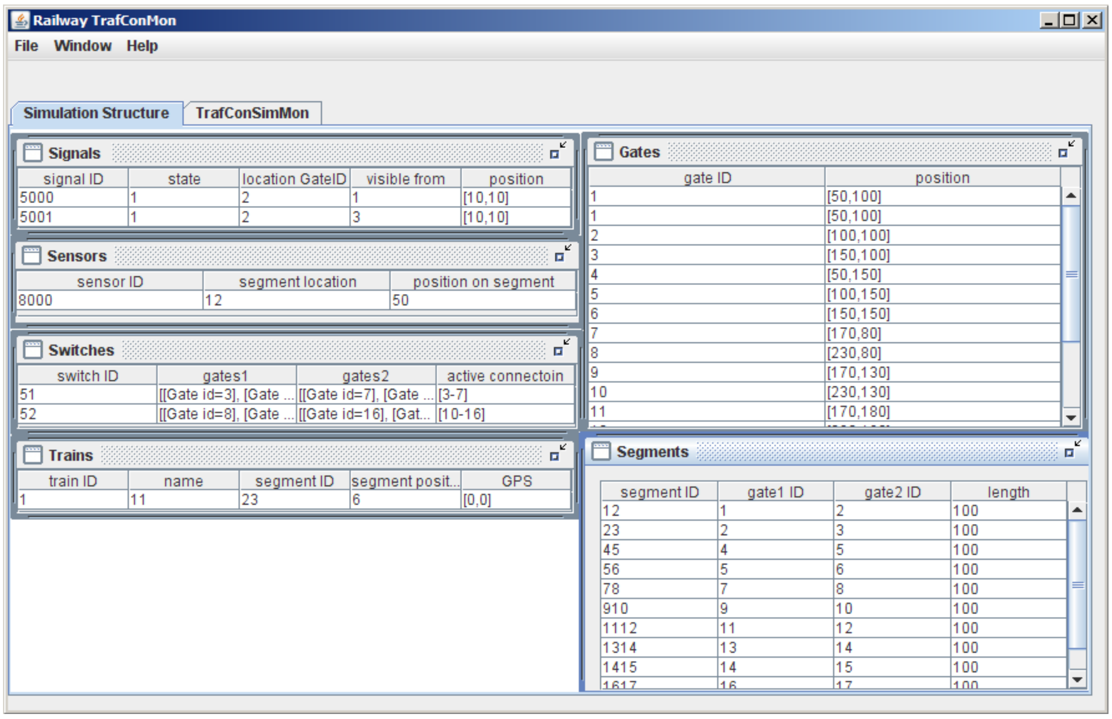
Signal ID: 5000 [✓ GREEN] RED

Switch ID: 51, Con1: 3, Con2: 7

Speed: 100

EVENT 8:1
EVENT 17:1

Figure 5. Simulator application main view.



Signals

signal ID	state	location GateID	visible from	position
5000	1	2	1	[10,10]
5001	1	2	3	[10,10]

Sensors

sensor ID	segment location	position on segment
8000	12	50

Switches

switch ID	gates1	gates2	active connectoin
51	[[Gate id=3], [Gate ...	[[Gate id=7], [Gate ...	[3-7]
52	[[Gate id=8], [Gate ...	[[Gate id=16], [Gat...	[10-16]

Trains

train ID	name	segment ID	segment posit...	GPS
1	11	23	6	[0,0]

Gates

gate ID	position
1	[50,100]
1	[50,100]
2	[100,100]
3	[150,100]
4	[50,150]
5	[100,150]
6	[150,150]
7	[170,80]
8	[230,80]
9	[170,130]
10	[230,130]
11	[170,180]

Segments

segment ID	gate1 ID	gate2 ID	length
12	1	2	100
23	2	3	100
45	4	5	100
56	5	6	100
78	7	8	100
910	9	10	100
1112	11	12	100
1314	13	14	100
1415	14	15	100
1617	16	17	100

Figure 6. Simulator structure view.

AUTHOR BIOGRAPHIES

Camelia Avram works in the field of designing and implementing of real time applications and discrete events systems applied in communications protocols.

Mihai Hulea works in the filed of object oriented programming and real time applications.

Architectural and Organizational Infrastructure for HPC Systems

PARALLEL CLUSTERING AND DIMENSIONAL SCALING ON MULTICORE SYSTEMS

Xiaohong Qiu
Research Computing UITS
Indiana University Bloomington
Email: xqiu@indiana.edu

Geoffrey C. Fox, Huapeng Yuan, Seung-Hee Bae
Community Grids Laboratory
Indiana University Bloomington
gcf@indiana.edu yuanh@indiana.edu sebae@indiana.edu

George Chrysanthakopoulos, Henrik Frystyk Nielsen
Microsoft Research Redmond WA
georgioc@microsoft.com henrikn@microsoft.com

INVITED PAPER

KEYWORDS
Multicore, Grids, Data mining, Parallel Programming

ABSTRACT

Technology advances suggest that the data deluge, network bandwidth and computers performance will continue their exponential increase. Computers will exhibit 64-128 cores in some 5 years. Consequences include a growing importance of data mining and data analysis capabilities that need to perform well on both parallel and distributed Grid systems. We discuss a class of such algorithms important in Chemoinformatics, bioinformatics and demographic studies. We present a unified formalism and initial performance results for clustering and dimension reduction algorithm using annealing to avoid local minima. This uses a runtime CCR/DSS that combine the features of both MPI, parallel threaded and service paradigms.

1. INTRODUCTION

There are many important trends influencing scientific computing. One is the growing adoption of the eScience paradigm which emphasizes the growing importance of distributed resources and collaboration. Another is the data deluge with new instruments, sensors, and the Internet driving an exponential increase of data [1]. On the other hand, multicore chips are challenging because they require concurrency to exploit Moore's law in contrast to the improved architectures and increasing clock speed of the last 15 years that has allowed dramatic performance increase within a well established fixed (sequential) programming paradigm [2-4]. Thus we suggest that it is important to look at data analysis and data mining and derive efficient multicore implementations. The data deluge, its management in a distributed environment and its analysis (mining) are relevant for both eScience and commodity applications. The former could involve data from high throughput instruments used in Life Sciences. The latter includes the analysis of environmental and surveillance monitors or the data fetched from the Internet that could characteristic a user's interests. The RMS (Recognition, Mining, Synthesis) analysis from Intel [5, 6] identified data mining and gaming as critical applications for multicore chips. Scientific data is likely to be so voluminous that we need any implementation to work well on clusters of multicore chips with preferably the same programming model for the inter-chip as well as the intra-chip parallelism. On the other hand commodity applications might well not need cluster implementations but probably would choose thread-based runtimes involving managed code – Java or C#. Data is often distributed so the Grid capabilities are essential; data mining can be extremely computationally intense so parallel implementations will sometimes be necessary.

The importance of Grids and multicore to both eScience (scientific computing) and commodity applications, motivates us to look at scientific data mining but in a programming model that is natural for related commodity applications. This motivates the SALSA (Service Aggregated Linked Sequential Activities) [7] research that we describe here. SALSA is implementing a set of data mining applications on multicore systems using managed code (C#) with parallel synchronization from a runtime CCR (Concurrency and Computation Runtime) developed at Microsoft Research [12, 13]. CCR supports both MPI style synchronization and the dynamic threading essential in many concurrent commodity applications. Further there is a service model DSS (Decentralized System Services) built on top of CCR [14]. CCR is a possible choice of runtime that could bridge between scientific and commodity applications as it supports the key concurrent primitives used in both of them. SALSA proposes that one builds applications as a suite of services [8] rather than traditional subroutine or class libraries. The service model allows one to support integration within grid, cluster and inter-chip environments. Thus SALSA is exploring a possible future application (data mining) on multicore chips using a programming model that could be used across a broad set of computer configurations

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

and could be the basis of an approach that integrates scientific computing with commodity applications. We note that we program in a low level style with user responsible for explicit synchronization in the fashion that is familiar from MPI. There certainly could be general or domain specific higher level environments such as variants of automatic compilation, OpenMP, PGAS or even the new languages from Darpa's HPCS program [6, 15]. Our work can still be relevant here as it uses a runtime that is a natural target for such advanced high-level environments.

Performance is a critical question for any system that spans multiple different domains; integration of multiple paradigms requires that the performance is good in each paradigm. In previous papers [9-11], we have discussed the core performance of CCR and DSS and here we focus on applications and discuss in more detail their structure and performance.

The SALSA work is currently performed on a variety of two CPU multicore systems with a total of 4 or 8 cores and running variants of Linux and Windows operating systems. The results reported in this paper use a single 8 core machine termed Intel8b. This is a Dell Precision PWS690, with 2 Intel Xeon CPUs x5355 at 2.66GHz, an L2 Cache 2X4M, 4GB Memory, and running Vista Ultimate 64bit.

In the following section we briefly discuss our programming model and refer the reader to other papers [9-11] for more details. In section 3, we discuss the data mining algorithms investigated and give some overall performance results. In section 4, we summarize our results and identify the key features of the application structure and the implications for the parallel run time. Conclusions are in section 5.

2. PARALLEL CCR RUNTIME

CCR provides a framework for building general collective communication where threads can write to a general set of ports and read one or more messages from one or more ports. The framework manages both ports and threads with optimized dispatchers that can efficiently iterate over multiple threads. All primitives result in a task construct being posted on one or more queues, associated with a dispatcher. The dispatcher uses OS threads to load balance tasks. The current applications and provided primitives support a dynamic threading model with capabilities that include:

1) *FromHandler*: Spawn threads without reading ports
2) *Receive*: Each handler reads one item from a single port
3) *MultipleItemReceive*: Each handler reads a prescribed number of items of a given type from a given port. Note items in a port can be general structures but all must have same type.
4) *MultiplePortReceive*: Each handler reads a one item of a given type from multiple ports.
5) *JoinedReceive*: Each handler reads one item from each of two ports. The items can be of different type.
6) *Choice*: Execute a choice of two or more port-handler pairings
7) *Interleave*: Consists of a set of arbiters (port -- handler pairs) of 3 types that are Concurrent, Exclusive or Teardown (called at end for clean up). Concurrent arbiters are run concurrently but exclusive handlers are not.

One can spawn handlers that consume messages as is natural in a dynamic search application where handlers correspond to links in a tree. However one can also have long running handlers where messages are sent and consumed at a rendezvous points (yield points in CCR) as used in traditional MPI applications. Note that "active messages" correspond to the spawning model of CCR and can be straightforwardly supported. Further CCR takes care of all the needed queuing and asynchronous operations that avoid race conditions in complex messaging. CCR is attractive as it supports such a wide variety of messaging from dynamic threading, services (via DSS described in [9]) and MPI style collective operations.

```
Main Routine for Exchange Pseudocode {
    Create CCR dispatchers to control threads
    Create a queue to hold tasks
    Set up start ports with MPI initialization data such as thread
      number
    Invoke handlers (MPI threads) on start ports
} End Main Routine

MPI logical thread Pseudocode (Arguments are start port
contents) {
    Calculate nearest neighbors for exchange collective
    Loop over stages  { Post information to 2 ports that will be
      read by left and right neighbors

      yield return on CCR MultipleItemReceive will wait till this
      thread's information is available in its ports and continue
      execution after reading 2 ports

      Do computation for this stage
    } End loop over stages

   Each thread sends information to ending port
   and thread 0 only does yield return on CCR
   MultipleItemReceive to collect information from all threads to
   complete run after reading from one port for each thread
   (this is a reduction operation).
} End MPI Thread
```

CODE SAMPLE 1: MPI EXCHANGE IN CCR

We have [11] already compared CCR with MPI and note that posting to a port in CCR corresponds to a MPISEND and the matching MPIRECV is achieved from arguments of handler invoked to process the port. MPI has a much richer set than CCR of defined methods that describe different synchronicity options, various utilities and collectives. These include the multi-cast (broadcast, gather-scatter) of messages with the calculation of associative and commutative functions on the fly. It is not clear what primitives and indeed what implementation will be most effective on multicore systems and so we have not performed an exhaustive study of MPI collective patterns in CCR. In fact it is possible that SALSA's results which suggest one can support in the same framework a set of execution models that is broader than today's MPI, could motivate a new look at messaging standards for parallel computing. CCR only has built-in primitives to support MPI shift and reduction operations but we exploited CCR's ability to construct customized collectives sketched in Code Sample I to implement the MPI Exchange pattern [11]. An important innovation of the CCR is to allow sequential, asynchronous computation without forcing the programmer to write callbacks, or continuations, and at the same time not blocking an OS thread. This allows the CCR to scale to tens of millions of pending I/O operations, but with code that reads like synchronous, blocking operations.

Note that all our work was for managed code in C# which is an important implementation language for commodity desktop applications although slower than C++. In this regard we note that there are plans for a C++ version of CCR which would be faster but prone to traditional un-managed code errors such as memory leaks, buffer overruns and memory corruption. The C++ version could be faster than the current CCR but eventually we expect that the C# CCR will be within 20% of the performance of the C++ version. CCR has been extensively applied to the dynamic threading characteristic of today's desktop application but its largest use is in the Robotics community. One interesting use is to add an efficient port-based implementation of "futures" to C#, since the CCR can easily express them with no modifications in the core runtime. CCR is very portable and runs on both CE (small devices) and desktop windows. DSS sits on top of CCR and provides a lightweight, service oriented application model that is particularly suited for creating Web/Grid-style applications as compositions of services running in a distributed environment. Its use in SALSA with performance results is described in [9, 10] and very briefly in Section 4 of this paper.

3. DATA MINING

In this paper we consider data mining algorithms that analyze a set of N data points $\underline{X}(x)$ labeled by x in a D dimensional space. These algorithms have a common formalism corresponding to iterative minimization of the function F given by equations (1) and (2).

$$F = -T\sum_{x=1}^{N} a(x)\ln Z(x) \text{ where} \qquad (1)$$

$$Z(x) = \sum\nolimits_{k=1}^{K} g(k)\exp[-0.5(\underline{X}(x)-\underline{Y}(k))^2/(Ts(k))] \quad (2)$$

There are four useful algorithms covered by the above: Clustering with Deterministic Annealing (CDA) [16-19]; Gaussian Mixture Models (GMM) [21]; Gaussian Mixture Models with DA (GMMDA) [22]; Generative Topographic Maps (GTM) [20]. We show how equations (1) and (2) cover each of these cases below. Note that for GMM and GTM, F is directly the cost function C (or negative of log likelihood) while for the annealing cases CDA and GMMDA, F is C-TS, the "free energy" where T is a temperature and S is the Shannon Entropy [18]. The sum over k corresponds to sum over clusters or mixture model components. A key characteristic of all these algorithms is "missing data" represented by the sum over clusters k in equation (2). We do not know a priori which value of k (e.g. which cluster) is associated with each data point $\underline{X}(x)$. This missing data characteristic also allows the applicability of the well known EM method [21] which is similar to steepest descent and can be shown to always decrease the objective function F in all four cases. Steepest descent methods are prone to find local minima so DA is attractive as it mitigates the effect of local minima.

In the annealing method one includes the entropy associated with these degree of freedom k and minimizes the Free Energy. The temperature is varied in an annealing schedule from high values (when F is dominated by entropy) to low values when the true cost C dominates. Unlike simulated annealing, DA involves no Monte Carlo but rather optimizes (1, 2) iteratively as temperature T is varied from high to low values. For clustering CDA improves on the well known K-means clustering algorithm [20]. In our cases the annealing can be interpreted as a multi-scale approach with $T^{1/D}$ as a distance scale. Now we define the four methods for which equations (1, 2) can be used and after that discuss their solution and our approach to parallelism.

For the first example, CDA clustering, the variables in (1) are given by:

$$a(x) = 1/N,\ g(k)=1,\ s(k) = 0.5 \qquad (3)$$

and T is temperature decreased to 1 by some schedule. DA finds K cluster centers $\underline{Y}(k)$ *where K is initially 1 and is incremented by algorithm as T decreases.*

We emphasize that unlike K-Means [19] or GMM, one need not specify the number of clusters K a priori in CDA. Rather K is determined by the annealing; as the distance scale decreases (see example in fig. 5 later), more clusters are determined. In the extreme limit T=0, all points x become clusters of size one and $K=N$.

For the second example, Gaussian Mixture Models GMM are defined by:

$$a(x)=1,\ g(k)= P_k/(2\pi\sigma(k)^2)^{D/2},\ s(k)= \sigma(k)^2 \qquad (4)$$

The component probability P_k, *the standard deviation* $\sigma(k)$ *and component center* $Y(k)$ *are varied with number of components K fixed a priori.*

Equation (3) specializes to a common case of spherical distributions. The general case has *s(k)* as a general symmetric *DxD* correlation matrix but this does not impact key ideas; it just makes the formalism more complex. Of course the model components in GMM are "just" clusters but GMM is more natural than clustering when the components have very different sizes. Although GMM makes P_k and $\sigma(k)$ as fitted variables, the formulae for their estimated values (using EM Method) are in fact identical to those from clustering. So CDA can find variable sized clusters although GMM could be a more precise approach.

One can easily extend GMM to add annealing [22] although there is currently little practical experience. This leads to GMMDA, which is given by:

$$a(x)=1,\ g(k)= \{P_k/(2\pi\sigma(k)^2)^{D/2}\}^{1/T},\ s(k)= \sigma(k)^2 \qquad (5)$$

and T is temperature decreased to 1 by some schedule. GMMDA finds K component probabilities P_k, standard deviations $\sigma(k)$ and component centers Y(k) where K is initially 1 and is incremented by algorithm as T decreases.

The final algorithm considered here has a very different goal to GMM and DAC; namely it addresses dimensional scaling or the derivation of a set of vectors $\underline{v}_i$ in a metric space where the distance between vectors i and j is given by a known discrepancy function δ_{ij}. Here δ_{ij} may come from the distance between points i and j in another vector space or be a discrepancy derived from an algorithm like BLAST comparing sequences in bioinformatics. In particular, we look at a powerful algorithm GTM (Generative Topographic Mapping) developed in [20] and often used to map from high dimensional spaces to two or three dimensions for visualization. This is illustrated in fig. 1 showing the 2D GTM projection of a set of three Gaussians in a 5D space. Note that one could use the simple Principal Component Analysis (PCA) approach [23]. This gives an optimal linear projection but does not perform well on complex problems whereas GTM has a nonlinear algorithm that is broadly effective [20]. PCA gave poor results on many Cheminformatics problems in high dimensions as the top two eigenvectors (used by PCA for 2D projection) do not capture much of the structure. Fig. 2 shows a successful GTM projection of two clusters in a D=155 dimensional Cheminformatics case. GTM is defined in the syntax of Equation (1) by:

$a(x) = 1$; $g(k) = (1/K)(\beta /2\pi)^{D/2}$; $s(k) = 1/\beta$; $T = 1$ and β and $\underline{W}_m$ are varied for fixed K, $\underline{L}(k)$, and M below. $\underline{L}(k)$, $\underline{\lambda}$ and $\underline{\mu}_m$ are vectors in the latent (2D) space.

$$\underline{Y}(k) = \sum_{m=1}^{M} \underline{W}_m \phi_m(\underline{L}(k)) \text{ with fixed} \qquad (6)$$

$$\phi_m(\underline{\lambda}) = \exp(-0.5(\underline{\lambda} - \underline{\mu}_m)^2 / \sigma^2)$$

GTM has excellent scaling properties and works well on large problems. We are currently applying it to the over 10 million compounds in PubChem. GTM can used for both clustering and dimensional reduction but we use DA for clustering and GTM just for the projection to 2D. The clusters found in GTM are viewed as a convenient averaging of the high dimensional space. For example in figs. 1 and 2 with 3 or 2 "real" clusters respectively, our GTM used K=225 averaging clusters. This large number of clusters used in averaging for projection leads to exceptional parallel performance for GTM given below. As discussed above, GTM and DA clustering are essentially the same algorithm with different optimizations; one for mapping and the other for robust clustering. Note GTM is closely related to SOM (Self Organizing Maps) but there are other important dimensional scaling methods whose parallel implementation we are also working on. The classic MDS (Multi Dimensional Scaling) approach using the SMACOF algorithm [23] has the advantage that it preserves distances δ_{ij} and not just the spatial topology mapped by GTM. Further it does not need vectors in the original space but just their discrepancies δ_{ij}. Other interesting methods with this property include random projection [24] and quadratic programming methods [25]. We will report on the multicore implementation of these other projection methods later. Here we just look at GTM which is a powerful efficient tool when the original vectors $\underline{X(x)}$ are known. We note that one could add annealing to GTM but we have not explored this yet.

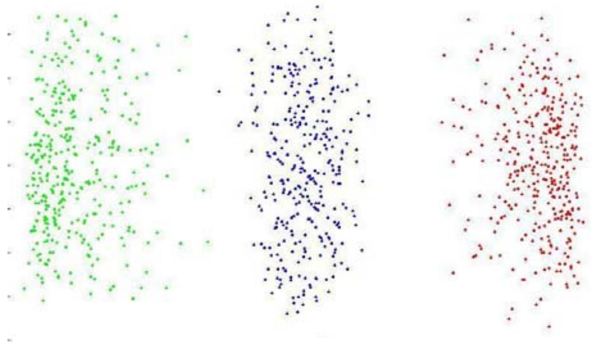


Figure 1: GTM Projection of a simple test problem with three Gaussians in a D=5 dimensional space

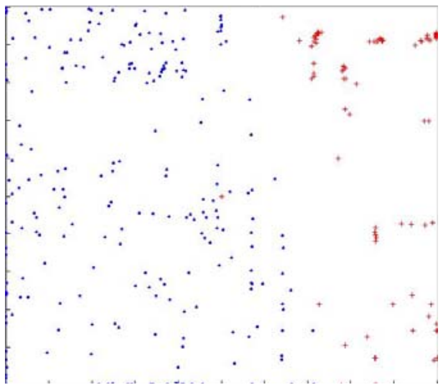


Figure 2. GTM Projection for 2 clusters from DA in space of 155 Chemical Properties labeled as a . or +

For the four algorithms defined in equations (3-6), solution of (1, 2) is implemented by a variation of the Expectation Maximization (EM) algorithm [21]:

$$\underline{Y}(k) = \sum_{x=1}^{N} \underline{X}(x)\Pr[\underline{X}(x) \in C(k)] / \sum_{x=1}^{N} \Pr[\underline{X}(x) \in C(k)] \quad (7)$$

$$\Pr[\underline{X}(x) \in C(k)] = \exp[-0.5(\underline{X}(x) - \underline{Y}(k))^2 / T] / Z(x) \quad (8)$$

written for the case of DA clustering where new values of cluster centers *Y(k)* are calculated iteratively from probabilities of *x* belonging to cluster *C(k)*. GTM, GMM and GMMDA have similar formulae with more quantities being calculatedly but always as averages with probabilities that a point $\underline{X}(x)$ belongs to a component/cluster *k*.

Realistic implementations must support both conventional real valued quantities in the above equations and also binary variables (for chemical fingerprints) and profiles in bioinformatics where the variables represent the frequencies with which features occur.

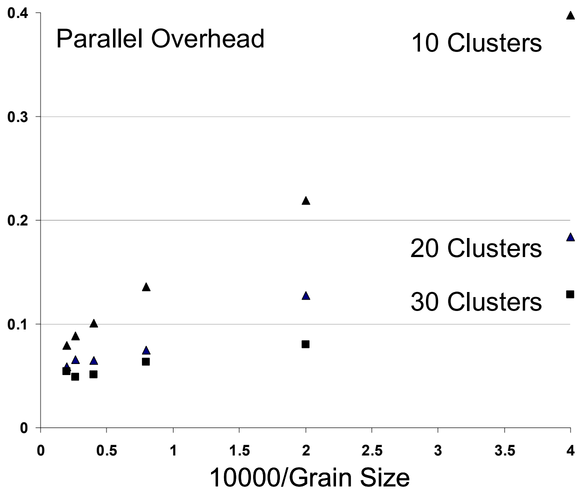


Figure 3. 8 core Parallel Overhead (approximately 1-efficiency) from Equation (9) for GIS 2D DA Clustering on Intel8b for three values (10, 20,30) of the number of clusters and plotted against 10000/N

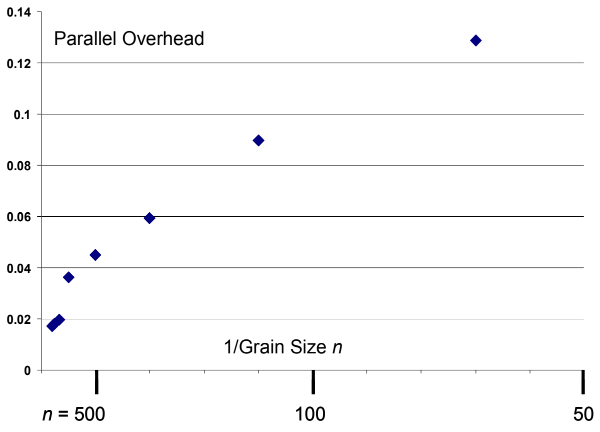


Figure 4. 8 core Parallel Overhead defined in Eq. (9) on Intel8b plotted against 8/N for GTM using M=256 and K=4096

Initial results on the parallel performance of DA clustering are shown in fig. 3 for runs on the 8 core Intel machine Intel8b used in all results presented in this paper. The figure shows that DA has a parallel overhead [26, 27] that decreases asymptotically like 1/grain size as the data set increases. Here grain size *n* is the dataset size N divided by the number of processors (cores) which is here 8. Putting *T(P)* as the execution time on *P* cores, we can define:

Overhead $f = (PT(P)-T(1))/T(1)$ (9)

Efficiency $\varepsilon = 1/(1+f)$ = Speed up $S(P)/P$ (10)

Thus the overhead of 0.05 seen for large *n* (small *1/n*) in fig. 3 corresponds to an excellent speedup of 7.6. The results for GTM in fig. 4 show even smaller overheads even at small grain size due to the substantial additional computation (matrix multiplication and equation solving) in this case. We emphasize that much of the critical overhead in multicore parallel code is not synchronization but rather due to interference between cores in the memory subsystem.

Tables 1, 2 and 3 study the parallel overhead for GTM as a function of the variables *N* (number of points), *K* (number of averaging clusters), *M* (number of mapping functions). The overhead lies between .01 (speedup of 7.9) and .05 (speedup of 7.6) except for the "small problem" *N*=1000 data points in table 1, where the overhead rises to 18% for the smallest problem. These results emphasize the excellent parallel efficiency of these algorithms and that large problems run well!

Table 1: Parallel Overhead for GTM as function of *M*

M=	**128**	**256**	**512**	**768**	**1024**	**1280**
K=16384 N=20000	-	0.014	0.013	0.014	0.016	0.022
K=900 N=1000	0.18	0.17	0.16	0.09	-	-

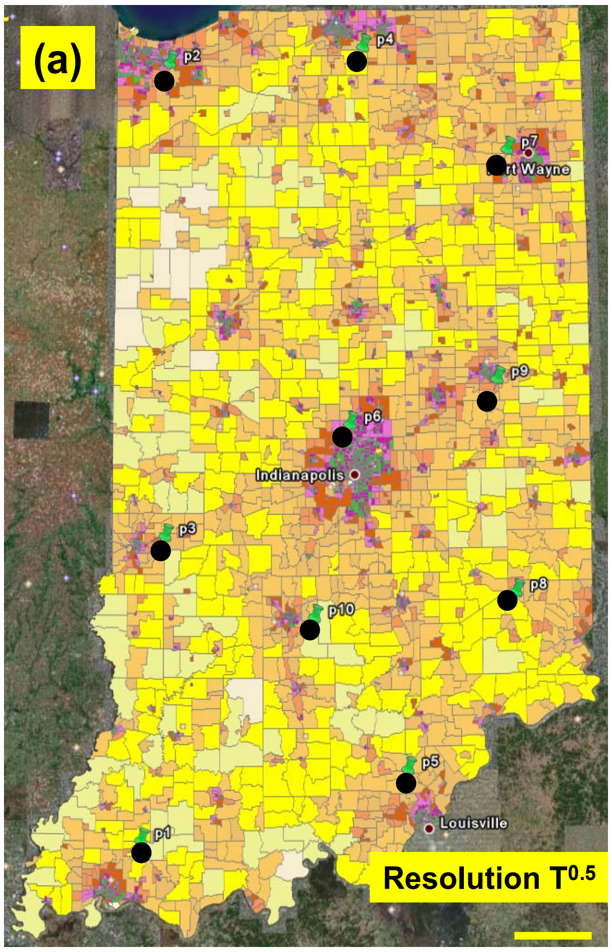
Table 2: Parallel Overhead for GTM as function of *K*

K=	**1024**	**2304**	**4096**	**6400**	**9216**	**12544**
N=20000 M=256	0.026	0.023	0.018	0.017	0.028	0.015

Table 3: Parallel Overhead for GTM as function of *N*

N=	**4000**	**8000**	**12000**	**16000**	**20000**
K=4096 M=256	0.045	0.036	0.020	0.018	0.017

In fig. 5, we illustrate the multi scale aspect of DA clustering with results from the clustering of the State of Indiana Census data for two temperatures in the annealing schedule. The value of the temperature is represented by a distance $T^{0.5}$ on figures 5(a) and 5(b). At the higher temperature in fig. 5(a), one finds 10 clusters. The larger cities in Indiana (such as the capital Indianapolis) are identified but some other municipalities are averaged together and not found individually. As we lower resolution in fig. 5(b), there are 30 clusters and most of the major towns in Indiana are identified. This figure shows possible studies one can perform as it looks at total population as well as three different groupings: Hispanics, Asians and renters with somewhat different clustering results.



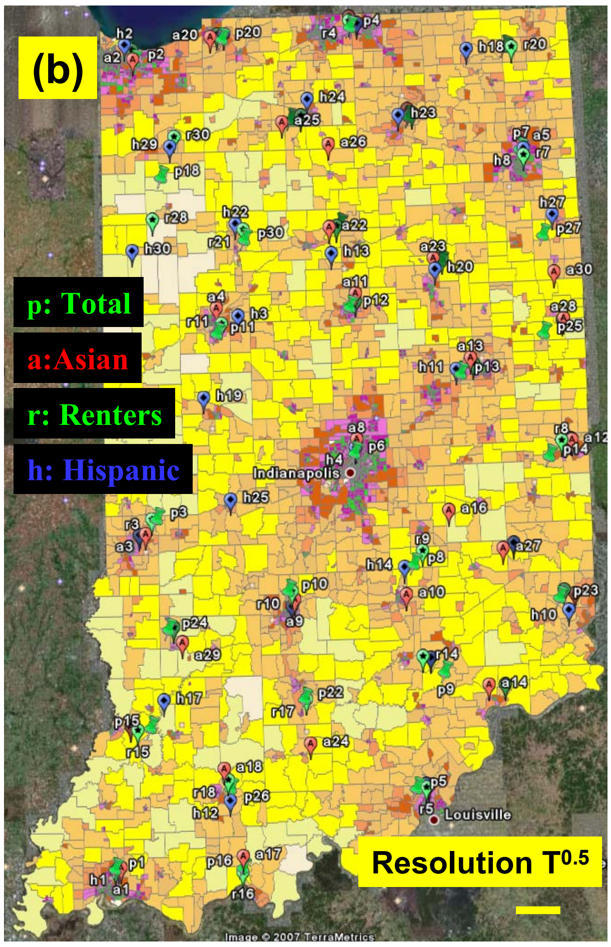


Figure 5. DA Clustering of Census data for state of Indiana showing *K*=10 clusters in (a) and 30 in (b). The resolutions $T^{0.5}$ are shown in bottom right. (a) shows just total population while (b) also shows Asian, Hispanic and Renter communities

4. PARALLEL PROGRAMMING

The algorithms illustrated in equations (1-8) have a structure familiar from many scientific computing areas [6, 26, 27]. There is an iteration – in this case over the annealing schedule for T and the steps needed for the EM method to converge. Synchronization is needed at the end of each iteration. Further looking into more detail, we find that the iteration consists of sums like Equations (7) and (8) calculating vector and matrix elements combined with linear algebra. The latter is identification of principal directions for CDA and GMMDA. There is no significant linear algebra for GMM while GTM needs matrix multiplication and linear equation solution. The sums themselves are first calculated in the memory of the threads and then after synchronization, accumulated into "global" variables.

This strategy assures good use of cache with negligible interference that occurs when two cores write to different memory locations that share the L1 cache [11]. Thus we see that all these algorithms have a "loosely synchronous" structure where the parallel algorithm consists of "compute-synchronize" stages where synchronization typically implies all cores reach a barrier [6, 27]. CCR supports the loosely synchronous paradigm with modest overheads analyzed in detail in earlier papers. Although CCR supports messaging like MPI, we only need CCR for synchronization in the applications considered in section 3. Data communication is achieved by accessing it in the memory shared by the threads. This is attractive as read access to shared information does not incur cache interference penalties. One does not need to put in thread memory the "edges" of regions as in Halos or ghost cells familiar in MPI. Rather one needs cache-sized blocks of data copied into thread memory; that is performed by the thread itself and not by communicating between threads. The critical source of overhead on a multicore chip is the memory subsystem.

Comparing our multicore implementations with traditional parallel programming, we see that we are using essentially the same programming model with domain decomposition breaking up the application into parallel components that execute in a loosely synchronous fashion. We use threads not processes in each core which allows us to optimize data connection between the decomposed components. Note we do need to link our thread based model inside a multicore system with a traditional distributed memory model if our algorithm needs parallelism across a cluster. The model for parallelism is identical inside and outside the multicore system but the data connection is different.

The fine grain parallelism is handled by CCR but this is not a complete software engineering model as it does not provide the desired modularity. Here we are using services as the building block. Services are attractive as they allow linkage to the distributed (Grid) programming model. We have successfully used DSS in in our early work. This is a Grid compatible service model which runs with high performance inside a chip. Further DSS is built on top of CCR which we use for synchronization inside the multicore and will use for linking to MPI for cluster operations. DSS has latencies of around 35 μs which corresponds to between 0.25 and 0.5 (floating point) million operations on an 8 core system achieving 1-2 Gflops per core. This implies that for example linear algebra on 100x100 matrices can be packaged as services without significant overhead. We have used DSS to encapsulate data reading, manipulation and visualization and will extend to break up the data mining itself in later work.

5. CONCLUSIONS

SALSA aims to develop scalable parallel data mining algorithms with good multicore and cluster performance and understand needed software runtime and parallelization method. We use managed code (C#) and package algorithms as services to encourage broad use assuming experts parallelize core algorithms. We have shown that Microsoft CCR supports well MPI, dynamic threading and via DSS a service model of computing. Detailed performance measurements give speedups of 7.5 or above on 8-core systems for "large problems" with deterministic annealed algorithms for clustering, Gaussian Mixtures, GTM and MDS (dimensional reduction). Future work includes further algorithms and applications as well as extensions to distributed and cluster implementations.

REFERENCES

[1] Tony Hey and Anne Trefethen, *The data deluge: an e-Science perspective* in "Grid Computing: Making the Global Infrastructure a Reality" edited by Fran Berman, Geoffrey Fox and Tony Hey, John Wiley & Sons, Chicester, England, ISBN 0-470-85319-0, February 2003

[2] Jack Dongarra Editor *The Promise and Perils of the Coming Multicore Revolution and Its Impact*, CTWatch Quarterly Vol 3 No. 1 February 07, http://www.ctwatch.org/quarterly/archives/february-2007

[3] David Patterson *The Landscape of Parallel Computing Research: A View from Berkeley 2.0* Presentation at Manycore Computing 2007 Seattle June 20 2007 http://science.officeisp.net/ManycoreComputingWorkshop07/Presentations/David%20Patterson.pdf

[4] Annotated list of multicore Internet sites http://www.connotea.org/user/crmc/

[5] Pradeep Dubey *Teraflops for the Masses: Killer Apps of Tomorrow* Workshop on Edge Computing Using New Commodity Architectures, UNC 23 May 2006 http://gamma.cs.unc.edu/EDGE/SLIDES/dubey.pdf

[6] Geoffrey Fox tutorial at Microsoft Research *Parallel Computing 2007: Lessons for a Multicore Future from the Past* February 26 to March 1 2007.

[7] Home Page for SALSA Project at Indiana University http://www.infomall.org/salsa and http://www.infomall.org/ has links to publications and presentations of SALSA group.

[8] Dennis Gannon and Geoffrey Fox, *Workflow in Grid Systems* Concurrency and Computation: Practice & Experience 18 (10), 1009-19 (Aug 2006), Editorial of special issue prepared from GGF10 Berlin

[9] Xiaohong Qiu, Geoffrey Fox, and Alex Ho Analysis of Concurrency and Coordination Runtime CCR and DSS, Technical Report January 21 2007

[10] Xiaohong Qiu, Geoffrey Fox, H. Yuan, Seung-Hee Bae, George Chrysanthakopoulos, Henrik Frystyk Nielsen *High Performance Multi-Paradigm Messaging Runtime Integrating Grids and Multicore Systems*, published in proceedings of eScience 2007 Conference Bangalore India December 10-13 2007

[11] Xiaohong Qiu, Geoffrey C. Fox, Huapeng Yuan, Seung-Hee Bae, George Chrysanthakopoulos, Henrik Frystyk Nielsen *Performance of Multicore Systems on Parallel Data Clustering with Deterministic Annealing* Proceedings of ICCS Krakow Poland June 23-25 2008.

[12] Microsoft Robotics Studio is a Windows-based environment that includes end-to-end Robotics Development Platform, lightweight service-oriented runtime, and a scalable and extensible platform. For details, see http://msdn.microsoft.com/robotics/

[13] Georgio Chrysanthakopoulos and Satnam Singh "An Asynchronous Messaging Library for C#", Synchronization and Concurrency in Object-Oriented Languages (SCOOL) at OOPSLA October 2005 Workshop, San Diego, CA. http://urresearch.rochester.edu/handle/1802/2105

[14] Henrik Frystyk Nielsen, George Chrysanthakopoulos, "Decentralized Software Services Protocol – DSSP" http://msdn.microsoft.com/robotics/media/DSSP.pdf

[15] Internet Resource for HPCS Languages http://crd.lbl.gov/~parry/hpcs_resources.html

[16] Geoff M. Downs, John M. Barnard *Clustering Methods and Their Uses in Computational Chemistry*, Reviews in Computational Chemistry, Volume 18, 1-40 2003

[17] Kenneth Rose, Eitan Gurewitz, and Geoffrey C. Fox *Statistical mechanics and phase transitions in clustering* Phys. Rev. Lett. 65, 945 - 948 (1990) http://dx.doi.org/10.1103/PhysRevLett.65.945

[18] Rose, K. *Deterministic annealing for clustering, compression, classification, regression, and related optimization problems*, Proceedings of the IEEE Vol. 86, pages 2210-2239, Nov 1998

[19] *K-means algorithm* at Wikipedia http://en.wikipedia.org/wiki/K-means_algorithm

[20] Bishop, C. M., Svensen, M., Williams, C. K. I. *GTM: The generative topographic mapping*. Neural Comput. 1998, 10, 215-234.

[21] Dempster, A.P., Laird, N.M., & Rubin, D.B. (1977). *Maximum-likelihood from incomplete data via the EM algorithm*. J. R. Statist. Soc. Ser. B (methodological), 39, 1–38.

[22] Naonori Ueda and Ryohei Nakano *Deterministic annealing EM algorithm* Neural Networks Volume 11, Issue 2, 31 March 1998, Pages 271-282 http://dx.doi.org/10.1016/S0893-6080(97)00133-0

[23] Ingwer Borg, Patrick J. F. Groenen *Modern*

Multidimensional Scaling: Theory and Applications Springer August 2005 ISBN-10: 0387251502

[24] Golan Yona *Methods for Global Organization of the Protein Sequence Space* PhD Thesis Hebrew University 1999 http://www.cs.cornell.edu/golan/Thesis/thesis.ps.gz

[25] E. Halperin, J. Buhler, R. Karp, R. Krauthgamer and B. Westover *Detecting protein sequence conservation via metric embeddings* Bioinformatics Vol. 19 Suppl. 1 2003 Pages i122-i129

[26] "The Sourcebook of Parallel Computing" edited by Jack Dongarra, Ian Foster, Geoffrey Fox, William Gropp, Ken Kennedy, Linda Torczon, and Andy White, Morgan Kaufmann, November 2002.

[27] Fox, G. C., Messina, P., Williams, R., "Parallel Computing Works!", Morgan Kaufmann, San Mateo Ca, 1994.

PRACTICAL PRECISE EVALUATION OF CACHE EFFECTS ON LOW LEVEL EMBEDDED VLIW COMPUTING

Samir Ammenouche, Sid-Ahmed-Ali Touati, William Jalby
University of Versailles St-Quentin en Yvelines, France
Email: saam@prism.uvsq.fr, Sid.Touati@uvsq.fr, William.Jalby@uvsq.fr

KEYWORDS

non-blocking cache, load-use distance, ILP.

ABSTRACT

The introduction of caches inside high performance processors provides technical ways to reduce the memory gap by tolerating long memory access delays. While such intermediate fast caches accelerate program execution in general, they have a negative impact on the predictability of program performances. This lack of performance stability is a non-desirable characteristic for embedded computing. We will present the progress of our experimental study about the influence of cache effects on embedded VLIW processors (*ST2xx* processors). We are trying to understand qualitatively and quantitatively the interactions between cache effects (Data cache) and instruction level parallelism at different granularities: applications and functions (coarse grain), program regions (medium grain) and instructions (fine grain). Our aim is to come up with experimental arguments helping to decide whether non-blocking caches would be a *reasonable* architectural design choice for embedded VLIW processors. By reasonable, we mean bringing opportunities at two levels: 1) program execution acceleration with tolerable performance predictability, and 2) active interactions with compiler optimization techniques. Our study is based on many months of full-time simulations on tens of workstations producing many terabytes of data to analyse.

Introduction

Cache effects permit to hide the existing gap between memories and processors performances. However, caches have a negative impact on loads latency predictability, depending on dynamic data location in the memory hierarchy. Our goal is to understand the execution behaviour by taking into account the cache effects; we measure the impact of different cache architectures and also the impact of the compiler. We follow a practical approach with common benchmarks (*mediabench*) and less common applications(*ffmpeg*) This is a typical embedded multimedia application used by STmicroelectronics to design their chips. It a video compression basing on h263 standard which are precisely simulated. The used industrial simulator (implemented by STmicroelectronics) models an embedded processor which is the *ST231* of STMicroelectronics. We are targeting reducing memory cache penalty. Reducing processors stalls due to memory access latencies is an old goal for the community. Some software techniques are available, like tiling (2), loop permutation, loop fusion, loop unrolling, loops jam, software prefetching (3). Hardware techniques are also available such as hardware prefetching and non blocking caches (4).

The authors in (1) explain the interest of non-blocking cache architecture for the Out-Of-Order processor. We try to measure the benefit of such cache architecture for an In-Order processor. To achieve that, we used an embedded VLIW processor, the *ST231*. We have collected the simulation results of the benchmarks on the *ST231* using two cache architectures: a blocking cache architecture and a non blocking one. We make comparison of obtained results, and we proposed a compilation method to better exploit the non-blocking cache feature.

Current commercial and academic backend compilers schedule all loads using the cache hit latency. That is, cache misses latencies are not used during instruction scheduling. Assuming statically that all data reside in low level caches involves a great difference between real and expected execution times. Current compilers do not schedule the loads operations using a miss latency because: 1) The benefit of the cache would disappear. 2) The register pressure would increase. 3) It is not always possible to schedule operations to hide such long load latencies (ILP extraction is limited in some applications). So, when a Dcache miss occurs, the VLIW processor stalls completely if the cache is blocking. If the cache is non-blocking, the VLIW processor continues to execute other pending operations but stalls quickly because the compiler schedule loads with short latencies (the distance between the issued load and the first reader is equal to a cache hit latency). We consider this gain too slender. We propose to adjust the load latencies in order to improve the gain of the non-blocking caches.

This paper is organized as follows. Sect. 1 first presents some related work. Our target embedded processor is described in Sect. 2. Sect. 3 presents the collected simulation results of the blocking cache architecture. Sect. 4 presents the collected results on the non-blocking cache architecture. Before concluding, we also present our proposal for a compile-time pre-loading technique to generate better embedded VLIW codes in presence of a non-blocking cache.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

1 Related Work

Several research on cache effects at fine grain level have been carried out. Touati included the impact of the compulsory misses in an optimal acyclic scheduling problem (6) in a single basic block. He models the exact scheduling problem by including the constraint of data dependences, functional units, registers and compulsory misses. Our current work is different because we try to cover all the kinds of misses (compulsory, capacity and conflict). Also, we do not focus in a single DAG (basic block) only, we are interested in optimising of whole application. Tien *et al.* studied the effects of non-blocking loads and the prefetch in MIPS3000 processor (7), and tried some compiler optimisation adapting loads to have more gain with the non-blocking loads. Whereas in our work, we study the cache effects for a VLIW (multiple issue) processor. We also use two phases compilation to adapt latencies to loads operations (as we will explain later). Oner *et al.* made a study of kernel scheduling over a MIPS processor (8). They increased the load-use dependency distance in loop kernel using loop pipelining. Ding *et al.* (9) made a static analysis of code to determinate which is a cache hit and which is a cache miss load instruction, this technique is called selective schedule. Abraham *et al.* (10) made a profiling of the load instructions, then the step of selecting loads which misses the cache. The final state is the prefetching of these delinquent loads. Our study is also based into profiling and analysis of trace, but we change the load-use distance rather than adding prefetch instructions. Furthermore, we aim to propose a pre-loading technique in conjunction with global scheduler (handling a whole function), and such scheduler does not necessarily target regular codes such as loop nests.

As far as we know, this is the first study demonstrating the practical effectiveness (or not) of a non-blocking cache inside an embedded VLIW processor. The next section presents our processor model.

2 ST231 Processor Description

The *ST231* (5) is the latest processor of the *ST2xx* in the market of embedded VLIW computing. It is a integer 32bits VLIW processor, 3 stages pipelined, which contains 4 integers units, 2 multiplications units and 1 load/store unit. It has a 64KB L1 cache. The latency of the L1 cache is 3 cycles. The cache is blocking, *i.e.* in the case of load cache-miss, the pipeline stalls until the commit of the pending load. The cache is separated Data/ Instruction. The Data cache is 4 way associative. It operate with write back no allocate policy. A 128 bytes write buffer is associated with the Dcache.

The next formula describes the execution time of a VLIW code on an *ST231* in function of different stalls sources resulted from dynamic hardware mechanisms:

$$T = Calc + DC + IC + InterS + Br$$

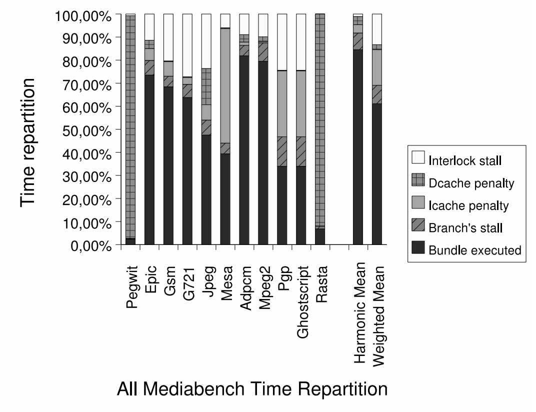


Figure 1: Mediabench Execution Time Distribution

where: T: is the total execution time in processor clock cycles, $Calc$: is the effective computation time in cycles, DC is the number of stall cycles due to Dcache misses, IC is the number of stall cycles due to instruction cache misses, $InterS$ is the number of stall cycles due to the interlock mechanism and finally Br is the number of taken branch (for each branch, there is one penalty cycle).

STmicroelectronics provided us a precise pipeline accurate simulator of the *ST231*. Our approach does not focus on benchmark's kernel only, we want to study and improve performance of application benchmarks using full precise, but long, simulation. The next section presents our performance analysis study of *ST231* using a regular blocking cache.

3 Blocking Cache Architecture Results

For a coarse grain profiling we use a simulator named ST200run with the simulator option -a statistics. It prints precise and detailed execution statistics. We collect simulation results of the *mediabench* and *ffmpeg* execution.

We can observe in Fig. 1 that a mean of 3,5% of time is lost in stalls due to Dcache misses. We focus on *pegwit* and *jpeg* benchmarks while Dcache miss represent 96.91% and 15.66% resp.

Fig. 2 shows that 33.34% of execution time is wasted in Dcache stalls. We calculate another parameter to quantify the Dcache misses, this parameter is the distance between a *load* operation and the first load's costumer operation. We call this distance as the *load-use distance*. There are two kinds of *load-use distances*. The first one is the load-use distance in cycles (Dynamic), which expresses the effective dynamic distance including all the stalls. The second one is the static distance set by compiler in the generated code (it is resulted from the static instruction scheduler).

Through several experiences, we observed a great difference between measured static distances and the dy-

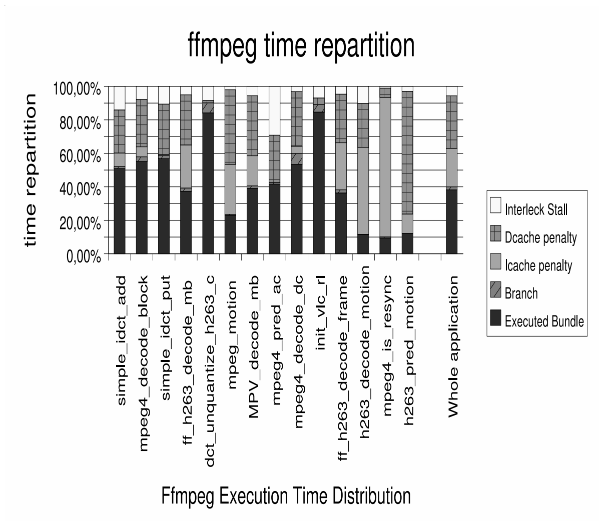


Figure 2: Ffmpeg Execution Time Distribution

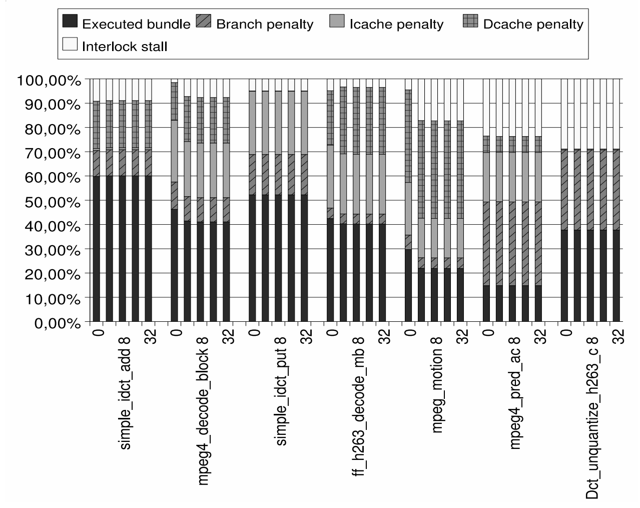


Figure 3: Execution Time Distribution of ffmpeg Different Sizes of Pending Load Queues (0, 1, 8, 16, 32)

namic ones. The mean of static distance values is around three bundles. This demonstrated that the compiler makes an optimistic load latency, it assumes that all data reside in the L1 cache (since the cache latency is 3 cycles). The dynamic load-use distances measured at the simulation time. Its mean value is around thirty cycles, which is far from the three optimistic cycles. The gap is due to Dcache misses. In the next section, we will see the contribution of a hardware mechanism (lock-up-free caches) generally designed for reducing stalls due to cache misses.

4 Non-blocking Cache Simulation Results

Kroft (11) defined the lock-up-free (non-blocking) caches. The interesting aspect of this architecture is the ability to overlap the execution and the memory data loading. When a cache miss occurs, the processor continues its execution of independent operations. This produces an overlap between bringing up the data from memory and the execution of independent instructions. In (1), the authors show that a non blocking cache can significantly improve the performances of an out-of-order (OoO) processor. So, many high performance OoO currently adopted this cache architecture. Embedded processors do not have non-blocking caches yet because: its cost is not negligible (energy consumption and price), and its benefit in cache of in-order processors is not demonstrated.

In order to make a full exploitation of non blocking, the memory architecture should also be improved. Indeed, memory must now become fully pipelined and multi-ported (These architectural enhancements are not an obligation in case of blocking cache). This improvement allows memory to serve multiple pending cache misses in a pipelined way. These cache misses are stored inside a queue (called *pending load queue*). The size of this queue, that we note *SPQ*, is a micro-architectural parameter which defines the number of concurrent loads waiting for memory service. Intuitively, when SPQ is large, more pending cache misses can be concurrent resulting in better load overlap. This section makes a precise performance evaluation resulting from adding a non-blocking cache inside an embedded VLIW processor (*ST231*). We also study the influence of SPQ, and the influence of the instruction schedules generated by the compiler.

In the first step, we collect the same execution statistics of the blocking cache experiences i.e. the number of cycles of effective calculation, the stall cycles due to Dcache misses, the stall cycles due to instruction cache misses, the cycle lost in branch and the interlock stalls. The used binary codes are the same used in Sect. 3.

Fig. 3 shows execution distribution time of the *ffmpeg* benchmark. We made distinct simulations, changing each time the size of the pending load queue size (SPQ) from 0 to 32 entries. A pending load queue equal to zero means that the architecture implements a blocking cache. A pending load queue with n entries means that at most n cache misses can be issued concurrently by the non-blocking cache.

For a pending load size equal to zero, Fig. 3 shows that the results are similar to the simulation results obtained with the blocking cache simulator (in Sect. 3, Fig. 2). The surprise is that there is a negligible performance improvement whatever the SPQ size. The performance improvement is 1.62% for the whole *ffmpeg* application with a SPQ equal to one! The result is similar in case of *mediabench* applications. Contrary to OoO processors, introducing a non-blocking cache in a VLIW in-order processor does not provide a performance gain, unless the codes are recompiled with some special instruction scheduling techniques (shown later).

Another result is shown in Fig. 3: when the SPQ is changed from 1 to 8, 16 and 32, we obtain the same performance gain 1.66% for the whole application. Contrary to OoO processors, increasing the SPQ size has little impact (unless we re-optimise the VLIW code as we will show later).

The *mediabench* benchmark simulation gives similar results, *i.e.* a weak performance improvement. this is shown in Fig. 4.

All the observed small speed-ups are due only to Dcache stall reduction. When considering exactly the same binary codes as in Sect. 3, executing them on the same VLIW processor but with changing the blocking cache non a non blocking one seems to do not alter other dynamic performance metrics: Icache stalls, branch penalties ans interlock stalls remain the same except Dcache stalls. This would improve the predictability of the execution time.

The experimental results of this section can be summarized as follows:

1. A disappointing cache stall reduction when changing cache configuration from blocking to non-blocking ones. The maximum obtained performance gain is 2.62% in the pegwit application and the worst one is less than 0.1% in MPEG2.

2. All the performance gains are calculated in the whole applications, not just in functions which make numerous Dcache misses. The performance improvement is of course better when the amount of Dcache misses is important.

3. The codes were not changed or tuned for the new cache architecture, the same binaries were executed over the two cache platforms.

4. We do not observe any speed-down due to the non-blocking cache.

5. The negligible speed-up is observed as maximal with a pending load queue size of 8 entries only.

All the negligible speed-ups obtained with a non-blocking cache architecture are disappointing but can be explained: when a Dcache miss occurs, the processor does not stall, it still execute the next bundle (VLIW) thanks to the non-blocking cache opportunity. However, the consumer of the loaded data is too close (three bundles later). Thus, the processor stalls too early and the benefit of the non-blocking cache is limited. We believe that the in-order architecture can better exploit the non-blocking cache architecture as well as the out of order does. However, the binary codes must be adapted to take in consideration the cache model. To avoid the poor performance improvement of the non-blocking cache architecture, we propose to reschedule the instructions by increasing the static load-use distance. We change the load instruction latency, and adapt its to each code. We must calculate for each load instruction the most suitable latency whatever it hits or misses the Dcache. For the loads that hit the cache, we do not need to change their latency.

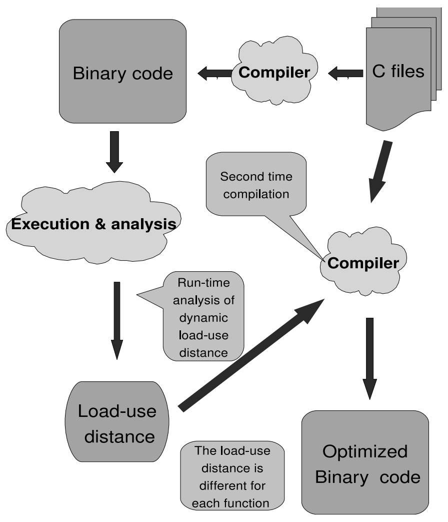


Figure 5: the used methodology

For the other loads, the static latency must be changed. For instance, the latency of the delinquents loads must be adapted.

In this section, we consider load-use intervals, where: load-use interval = [cycle of load , cycle of the user]. Thanks to the non-blocking cache, load-use intervals may overlap.

In order to compute a new metric each load capturing load-use intervals overlap, we propose the next formula:

$$NormalizedDistance = \left\lceil \frac{C_2 - C_1}{L} \right\rceil \tag{1}$$

where C_1 and C_2 are cycles when load or consumer load instruction occurs and L is the number of overlapped loads.

This *normalized distance* is our new metric that we use as a parameter to a new static compiler optimisation option. It sets a new static load latency in whole function scope: the normalized distance represents the number of additional static cycles to a cache hit latency. That is, a load that was initially scheduled with a cache hit latency by the compiler, becomes scheduled with a new latency equal to the cache hit plus the normalized distance. The overlap parameter L in Equ. 1 shows that when L is high then the normalized distance tends to zero. So, the static distance tends to a cache hit latency. This means that the compiler does not change its initial latency because a good machine usage (sufficient overlapped memory requests). If the value of L is low, the static load distance is increased to allow more pending loads to be executed in parallel during cache miss.

We have calculated this new normalized distance metric for all the *mediabench* and also for *ffmpeg* functions. We also experimented some regular usual codes such as, matrix-vector multiplication and matrix-matrix multiplication.

To decrease the Dcache stall penalty by adapting the static loads latencies. We use an on-the-fly trace analyser

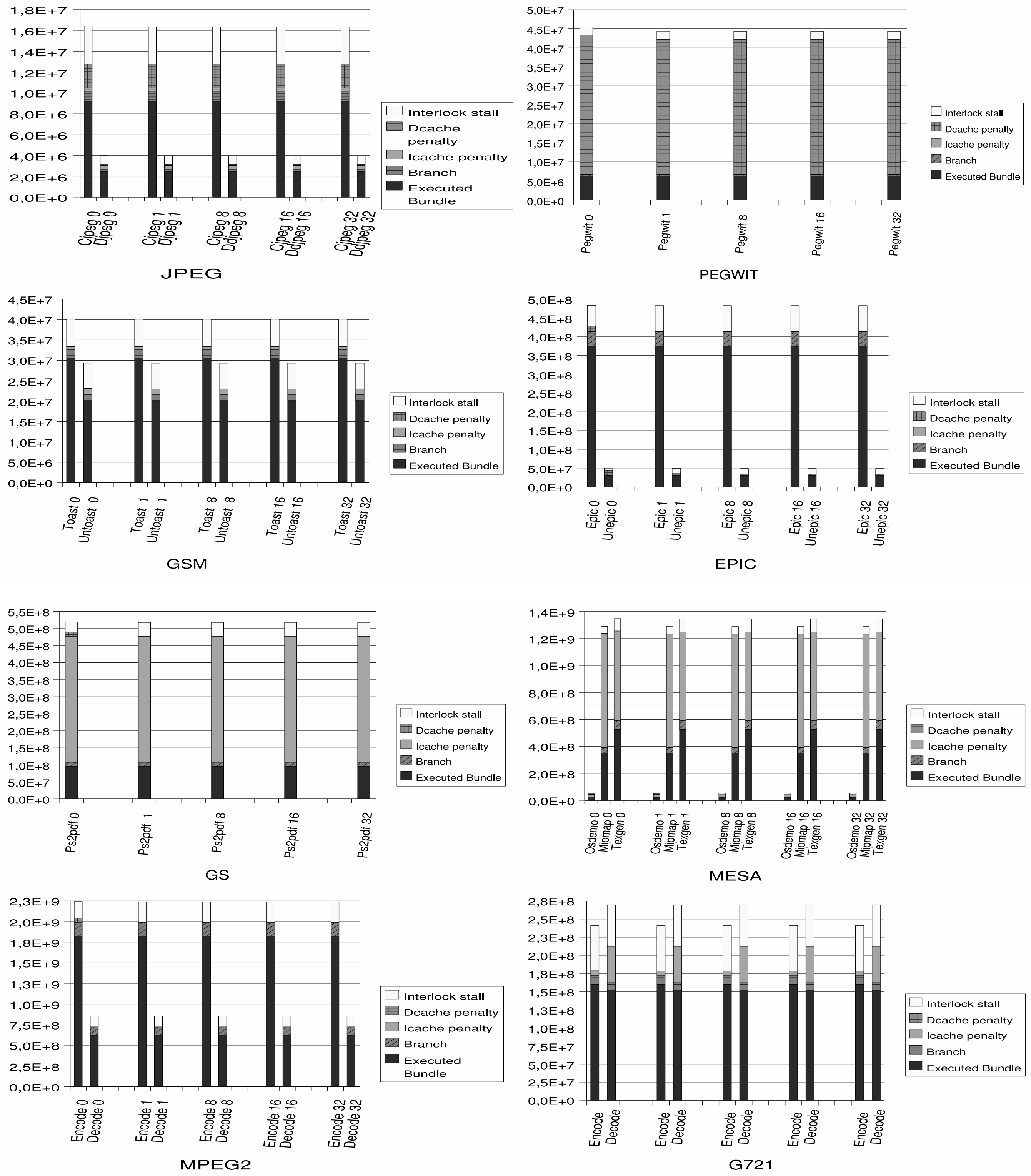


Figure 4: Execution Time Distribution of Mediabench with Distinct Pending Load Queue Sizes (0, 1, 8, 16, 32)

during the simulation to calculate the dynamic load-use intervals, and the number overlapped loads at each processor clock cycle. With Equ. 1, we manage to compute the new static load distance that we should apply in a re-compilation process. For this purpose, the regular optimising compiler of STmicroelectronics has been modified by the vendor to allow us such low level code optimisation. A new compiler version has been designed for our study. We can now adjust load latency by a special compiler option. Simply increasing the static load latencies without careful attention may produce many impacts in the final generated code:

Fig. 5 shows all steps of our methodology to decrease the data cache stall penalty by adapting the static loads latencies. We use an on-the-fly trace analyser during the simulation to calculate the dynamic load-use distances each processor clock cycle. With Equ. 1, we manage to compute the adapted load distance that should be applied in a re-compilation process. For this purpose, the regular optimising compiler of STmicroelectronics has been modified by the vendor to allow us such low level code optimisation. A new compiler version has been designed for our study. We can now adjust load latency by a special compiler option.

1. When instruction rescheduling, the code size may increase and consequently may have negative effects on instruction cache misses. So for some short loop, we force compiler to unroll it rather than pipeline it. In case of pipelined loop, increasing load latency can increase the II, however, in some case greater II can gives better performances than smaller one (due to cache effects which are not considered at scheduling time).

2. For the non-loop code, if the new latencies are too long, the compiler may not find enough ILP. To avoid that, several methods can be applied as tail duplication, region scheduling, Super-block instruction scheduling, trace scheduling, scheduling non-loop code with prologue/epilogue of loop blocks.

3. Increasing load latency increases the register pressure; the compiler can introduce spill code to reduce simultaneously alive variables. So, when scheduling, we must take care about register pressure.

Our normalized distance as proposed in Equ. 1 aims to reduce the negative impact described above. Furthermore, we should be aware that modifying a load latency may considerably modify the cache effects of a code: Since load operations are reschedule, some initial cache misses may become hits and *vice-versa*, because of the instruction rescheduling that modifies the spacial/temporal locality of the code. In order to guarantee that the cache effects stay the same before and after static load modification, we impose to the compiler (via a special pragma) to keep the same order for the loads before and after latency modification. Consequently, applying a new latency to loads does not modify the relative order between the loads, keeping the same cache effects (and thus, our normalised distances computed via an initial program simulation remains valid).

Also, we can see that the distance decreases while the size of the pending load queue increases until a limit where increasing the size of the pending load has no effect on the normalised. Not all functions are candidate for our code optimisation methodology. We consider the function that has two properties : 1) a considerable fraction of Dcache stall in the execution time of the function, and 2) the normalised distance should be larger that the cache hit latency (3 cycles). When we look for these two parameters (Dcache fraction plus considerable normalised calculated distance), we find that few functions in *ffmpeg* and *mediabench* are candidate to our optimisation. For ffmpeg application, we obtained 28.28% whole application speed-up using adapted loads latencies.

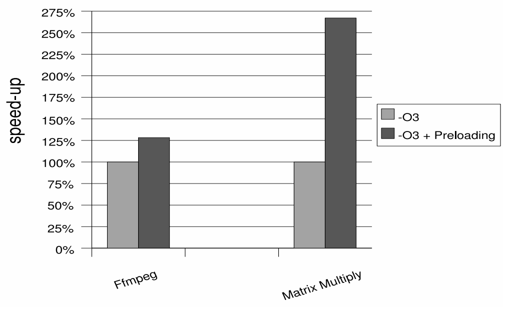


Figure 6: Original vs Optimized ffmpeg & Matrix-Matrix Multiplication Benchmarks Results

We can now apply our optimisation method in case of more well known benchmark such as square Matrix-Matrix (512*512) multiplication: we use a non-naive implementation, produced by ATLAS ("best" loop tiling, each tile contains 64 * 64 elements = 16KB which are kept by the Dcache). The obtained results are promising and conclusive. In Fig. 6, we can observe the positive effects of using the normalized distances. The main advantage of this optimisation is that it can be applied to any control flow graph, not necessary to loops. Finally the speed-up obtained thanks to Preloading and non blocking cache hides the cost of the added hardware (non-blocking cache)

5 Conclusion

Our study was based on precise full simulation of whole embedded applications (*mediabench* and *ffmpeg*). Our experimental study consumed many months of full simulation on tens of workstations producing tera-bytes of data to collect and analyse. We precisely measured the impact of a non-blocking cache inside a VLIW embedded processor (*ST231*) compared to a blocking cache architecture. As shown in our experimental results, if the binary codes are not modified, the performance improvement is poor (program acceleration less than 3% in the best case!). This situation has a concrete explanation. Many current compilers schedule load instruction too close to their consumers: this is a common heuristics to decrease the register pressure. Such scheduling heuristics assume that data reside in L1 cache, and consequently loads are scheduled as cache hits. Such scheduling heuristics reduces the benefit of a non-blocking cache in case of in-order and VLIW embedded processors. This situation is not altered when increasing the size of the hardware pending queue associated to the non-blocking cache.

Our experimental results are in opposition with the case of high performance out-of-order processors, where non-blocking caches provide positive effects in execution performance without changing program binaries. High performance out-of-order processors contain much more hardware mechanisms (resulting in higher costs) that allow program acceleration without instruction rescheduling at compile time. In the case of an embedded VLIW processor, we showed that if the code is not re-optimised in order to take into account the non-blocking cache, the benefit is negligible.

Our code optimisation methodology is based on data pre-loading. Our method performs in two steps. The first step computes normalised load-use distances using execution trace analysis. Then, this distance is used in the second step to reschedule the code at instruction level, while keeping the same loads order as the one analysed in the first step. Keeping the same loads order in the second step guarantees that the cache effects analysed in the first step are not altered in the second step. Our results on matrix-matrix multiply and ffmpeg show respectively a speed-up of 267.61% and 28.28% for the whole program execution. This provides us promising demonstration of the effectiveness of our ideas. In the future, we will combine pre-loading with data prefectching in order to optimise memory requests for both regular and irregular embedded VLIW codes.

REFERENCES

[1] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*, Morgan Kaufman, CA, 1996.

[2] Michael E. Wolf and Monica S. Lam. *A Data Locality Optimizing Algorithm* . PLDI'91, pages 30 – 44, 1991.

[3] Randy Allen and Ken Kennedy.*Optimizing Compilers for Modern Architectures*. Morgan and Kaufman, 2002.

[4] James Edward Sicolo. *A Multiported Nonblocking Cache for a Superscalar Uniprocessor* B.S. State University of New York at Buffalo, 1989.

[5] STMicroelectronics ADCS 7645929F ST231 Core and Instruction Set Architecture Manual, 2005.

[6] S.-A.-A. Touati. *Optimal Acyclic FineGrain Scheduling with Cache Effects for Embedded and Real Time Systems*. ACM Proceedings of the Ninth International Symposium on Hardware/Software Codesign. Copenhagen, Denmark, April 25-27, 2001, IEEE.

[7] Tien-Fu Chen and Jean-Loup Baer. *Reducing Memory Latency via Non-blocking and Prefetching Caches*. Proceedings of the fifth international conference on Architectural support for programming languages and operating systems. Boston, Massachusetts, United States. 1992.

[8] Koray Öner and Michel Dubois.*Effects of Memory Latencies on Non-Blocking Processor Cache Architecture*. Proceedings of the 7th international conference on Supercomputing ICS. 1993.

[9] Chen Ding and Steve Carr and Phil Sweany.*Modulo Scheduling with Cache Reuse Information*. European Conference on Parallel Processing. 1997.

[10] Abraham Sugumar, Windheiser, Rau, Gupta. *Predictability of Load/Store Instruction Latencies*. Proceedings of the 26th annual international symposium on Microarchitecture. Austin Texas. 1993.

[11] David Kroft. *Lockup-free Instruction Fetch/Prefetch Cache Organization* . Proc. 8th International Symposium on Computer Architecture, Minneapolis, MN, May 1981, p. 81-85.

Acknowledgements

This research result has been supported by the ANR MOPUCE project (ANR number 05-JCJC-0039) and the French Ministry of Industry. We would like to greatly thank Francesco PAPARIELLO and Giuseppe DESOLI from STMicroelectronics-Milano for their valuable effort in implementing the non-blocking cache simulator. We would also like to greatly thank Benoit DUPONT-DE-DINECHIN and Christophe GUILLON from STMicroelectronics-Grenoble for their valuable effort in implementing our pre-loading technique in the ST VLIW compiler.

Non-linear Seek distance for optimal accuracy of zoned disks seek time in Multi-RAID storage systems

Soraya Zertal[1] and Peter Harrison[2]
[1] PRiSM, Université de Versailles, 45 Av. des Etats-Unis, 78000 Versailles, France
Zertal@prism.uvsq.fr
[2] Imperial College London, South Kensington Campus, London SW7 2AZ, UK
pgh@doc.ic.ac.uk

ABSTRACT

Models of multi-RAID storage systems, implemented on modern zoned disks, are simplified by using an approximation for the seek distance that assumes that the number of sectors per track varies linearly with the cylinder number. Results obtained in this way have matched well against simulation, but the relationship of the number of sectors per track against cylinder number has turned out to be rearkably linear for the specific RAID systems modelled so far. This will not always be the case, necessarily, and in this paper, we go a step further, calculating exactly the seek distance moments, then those of the seek time on zoned disks for specific manufacturer's specifications. This is to ensure the highest accuracy possible for our model, especially for the non-sequential request streams. The results show good accuracy, even in highly non-linear systems, and indicates a threshold for the model parameters at which the linear approximation becomes unacceptable.

KEYWORDS

Multi-RAID, Zoned disks, Seek distance distribution, M/G/1 queues, I/O modeling and Simulation.

1 Introduction

A continual, heavy and increasing pressure persists on storage systems, necessitating accurate models of their operation, capable of analysing and predicting the performance and quality of service (QoS) these systems can deliver. To fulfill these requirements, models should provide detailed abstractions of a wide range of possible real system architectures. Many storage system models do exist and we can split them into three broad categories: the first focuses on calculating the service time for a given RAID configuration and a specific type of workload [10]; the second – widely investigated – concerns the analysis of the performance of a given RAID configuration in a specific working mode [2, 3, 16, 17, 18, 11, 1, 13]; and the third category studies the effect of caching and controller optimizations on a disk array's performance [15, 14]. Regardless of the main goal of a study among these categories, the objective is always one and only one RAID configuration per disk array. We proposed previously, in [28], a Multi-RAID model which is – as far as we know – the only one that models a RAID storage system with multiple, coexisting RAID organisations, using modern zoned disks. In that model, the probability distribution of the seek distance is considered to be a continuous random variable, assuming that the number of sectors per track varies linearly with the cylinder number. In this paper, we calculate exactly the seek distance moments to provide the highest accuracy for our model, which would become applicable to arbitrary zoned disks, possibly with a highly non-linear relationship between number of sectors and cylinder sequence number.

In the rest of the paper, section 2 considers the technology issues in the design of RAIDs of zoned disks as well as the details of our analytical model. Section 3 describes the calculation of the moments of the seek distance and their impact on our zoned Multi-RAID model. Results are presented and discussed in section 4, and the paper concludes in section 5 with a summary and some future related research objectives.

2 Technological and modeling context

A RAID storage system consists of a disk system manager and a collection (array) of independent disks. The disk system manager is a software component of the RAID controller. It is responsible for the logical to physical mapping of requests according to the prevailing RAID organisation scheme [4, 5].

We proposed a dynamic Multi-RAID architec-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

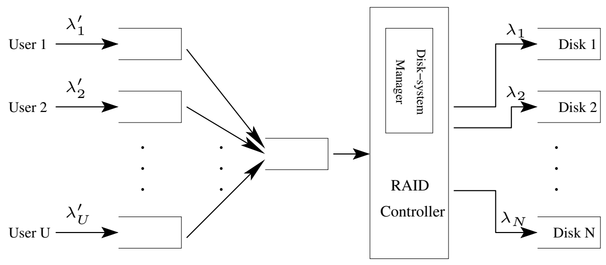


Figure 1: Requests flow in a RAID storage system

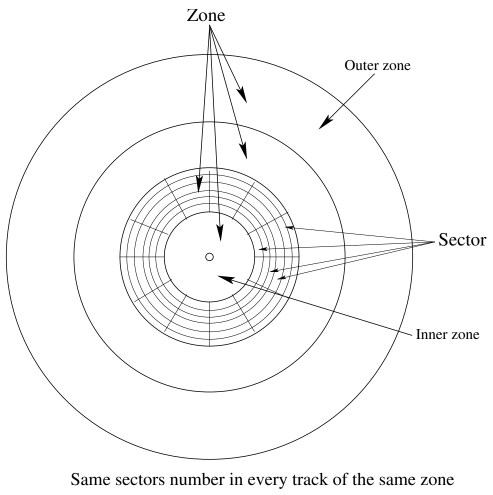


Figure 2: Zoned disk technology

ture, on which various RAID organisation schemes coexist, with a dynamic selection of a redundancy pattern during the data lifetime, in order to provide enhanced space use and access time, [26] . Related requests' independent executions on such asynchronous disks lead to Fork-Join-type modeling problems. We modelled this architecture, using a collection of M/G/1 queues with various extensions to account for the parallel disk (physical) accesses corresponding to a logical request [27, 8]. The response time of each physical request, to an individual disk, is composed of four components: the queueing time (Q), the seek time (S), the rotational latency (R) and the transfer time, which itself is divided into two components, t and T_{bus}, corresponding to the transfer time between the disk's cylinder and its buffer, and between this buffer and the controller via the bus, respectively. We first considered uniform disks[1] to validate our model [9] and extended our initial model to modern zoned disks using more accurate access time functions [28].

On zoned disks (see figure 2), the number of sectors per cylinder is variable. Consecutive cylinders are collected into groups, called *zones*, such that within each zone, the track capacity (number of sectors) and the transfer rate are fixed. However, these two parameters decrease from the outer to the inner zones. These disks have become very popular due to their greater storage capacitiy and transfer rate. Their average rotational latency is constant but the variable seek and transfer times necessitate more complex calculations in terms of the assumed statistical workload and disk operation principles in a storage model [20, 21].

The model we presented in [28] deals with an assumed linear relationship between the number of sectors per track and the cylinder number. This gives a good approximation to the seek distance distribution on such modern zoned disks in the context of a Multi-RAID system. As far as we know, there is no analytical model for a non-linear seek distance distribution on zoned disks. We approximated it using a linear function in [28], rather than by using interpolation, as in [24], or using parameters based solely on preliminary simulations, as in [19]. An approximation for seek time in terms of seek distance is proposed in [7]: proportional to the square root of seek distance, when below some threshold value, otherwise linear in the seek distance. A Chernoff bound on the transfer time of a 'sweep' of N requests is found, assuming equidistant seek positions, giving a constant total seek time for the sweep. Further, this model assumes that all zones have the same number of tracks and that the track capacity increases linearly, which is much more restrictive than our approximation in [28]. Here, we calculate the moments of seek distance exactly, for uniform address accesses, and hence of seek time, allowing highly non-linear regimes to be investigated. A glossary of the notation used is provided by Table 1.

3 The non-linear seek distance

Seek time is one of the main components of a disk's access time. The morphology of a zoned disk improves its performance but makes its modelling much more complicated because of having to deal with a non-constant (here, not even linear) seek distance between the read/write head's current position and its target one in any disk access. We assume that the incoming logical requests' addresses are independent random variables, uniformly distributed over the disk-address space. This does not mean that the accesses are uniformly distributed over the disk's physical space, of course – the access distribution over the physical space follows its density, i.e. higher on the outer zone and decreasing towards the

[1]Sectors are uniformly distributed across the tracks and all the tracks have the same number of sectors.

Param.	Description
N	Number of disks in the storage system.
C	Number of cylinders on a disk.
SEC	Number of sectors on the disk.
N_z	Number of zones in the disk.
SEC_c	Number of sectors on cylinder c.
spb	Number of sectors per block.
B	Logical request size (transfer block).
K_i	The number of blocks generated by a logical request at disk i
D_i	Seek distance on a disk i.
S_i	Seek time on a disk i. buffer and its cylinder.
λ	Logical request arrival rate to the storage system.
p_i	Probability that disk i is used.
λ_i	Physical request arrival rate to disk i.
λ_{Rj}	Physical request arrival rate to the RAIDj area.
λ_{iRj}	Physical request arrival rate to a RAIDj area on disk i.
P_{raid_j}	$RAIDj$ area's proportion in the whole storage system space.
p_w	Probability that a request is a write.
p_r	Probability that a request is a read.
p_s	Probability of a sequential access.
z_i	The probability to access a sector in zone i.
C_i	The first cylinder's number of zone i.
d_i	The number of cylinders in zone i.

Table 1: Notation for the RAID model's parameters

inner zone. Since the number of cylinders C is large, the seek distance D can be well approximated by a continuous random variable. On Uniform disks, the seek distance density function is [9]:

$$f_D(x) = p_s\delta(x) + (1-p_s)\frac{2(C-x)}{(C-1)^2}$$

for $0 \leq x \leq C-1$, where p_s is the probability that consecutive accesses are sequential, i.e. on the same track, requiring no seek for the second access.

The term $\frac{2(C-x)}{(C-1)^2}$ is the probability density function of the difference between two uniform random variables on $[0, C-1]$, and $\delta(x)$ is the Dirac delta-function (unit impulse). Turning now to zoned disks, assuming that the number of sectors (and hence blocks) per track varies linearly with the cylinder number, the density function of D can be shown to be [28]:

$$f_D(x) = A + Gx + Ex^3 \quad (0 \leq x \leq C-1)$$

Thus, the nth moment of the seek distance D can be calculated as:

$$M_n = (C-1)^{n+1}[\frac{A}{n+1} + \frac{G(C-1)}{n+2} + \frac{E(C-1)^3}{n+4}]$$

where

$$\begin{aligned}
A &= \frac{V(C-1)}{3\gamma^2} \\
G &= -\frac{V+\beta^2(C-1)^2}{3\gamma^2} \\
E &= \frac{\beta^2}{3\gamma^2} \\
V &= 6\alpha^2 + 6\alpha\beta(C-1) + 2\beta^2(C-1)^2 \\
\gamma &= \alpha(C-1) + \beta(C-1)^2/2 \\
\alpha &= SEC_{C-1}/spb, \\
\beta &= (SEC_0 - SEC_{C-1})/(spb(C-1))
\end{aligned}$$

We implemented this linear approximation and the obtained results showed good agreements when applied to a real disk device : Fujitsu-MAN3397 disk [6], as we can see in figure 3. This figure plots the exact number of sectors per track (as specified in the disk's technical data) against cylinder sequence number (with the higher numbered, inner cylinders on the left) over the whole disk. This graph is compared with the model's linearised version in the same figure, showing close agreement. However, we cannot rely on such an approximate linear relationship holding good on all zoned disks (see section 4). The reason is that on such disks, the streaming bandwith varies by over 50% from one part of the disk to another [22], showing clear non-linearity.

Very few applications writers exploit this non linearity, however, specifying explicitly the data placement so that the behaviour of an application can be predicted [23, 25]. Most programmers completely ignore the data placement that their applications will use. Models of such applications that employ (approximately) linearized seek time distributions do not, therefore, represent real system behaviour faithfully, and so predict performance poorly. This motivated our exact seek distance calculation, using real zoned disks' published hardware characteristics. This results in seek time moments being estimated by the formulae given in the proposition below:

Proposition 1 *Consider a disk with N_z zones, numbered $0, 1, \ldots, N_z - 1$ and C cylinders, counting from the outside of the disk. Let the first cylinder in zone i be numbered C_i, so that the number of cylinders in zone i is $d_i = C_{i+1} - C_i$. Then the n^{th} moment M_n of seek distance, assuming uniform access to sectors over the whole disk, is*

$$M_n = \frac{2}{(n+1)(n+2)} \sum_{0 \leq j < i}^{N_z - 1} \frac{z_i z_j}{(d_i - 1)(d_j - 1)} \times$$

$$[(C_i - C_{j+1} + 1)^{n+2} + (C_{i+1} - C_j - 1)^{n+2} - (C_i - C_j)^{n+2} - (C_{i+1} - C_{j+1})^{n+2}] + \frac{2}{(n+1)(n+2)} \sum_{i=0}^{N_z-1} z_i^2 (d_i - 1)^n$$

where $z_i = \frac{d_i \times SEC_i}{SEC}$ is the probability of a single request accessing a sector in zone i.

Proof Let the random variable D denote the seek distance for an access on the disk, assuming that all accesses are uniformly distributed over the sectors. Then, the probability that a given access is to a sector in zone i is z_i, as defined. Hence we have:

$$\begin{aligned}
M_n &= E[D^n] \\
&= E[E[D^n \mid \text{seek between cylinders } i \text{ and } j, i \geq j]] \\
&= \sum_{j<i} \frac{z_i z_j}{(d_i-1)(d_j-1)} \times \\
&\quad \int_{x=0}^{d_i-1} \int_{y=0}^{d_j-1} (C_i - C_j + x - y)^n \mathrm{d}x\mathrm{d}y \\
&\quad + \sum_{i=0}^{N_z-1} \frac{z_i^2}{(d_i-1)^2} \int_{x=0}^{d_i-1} \int_{y=0}^{d_i-1} |x-y|^n \mathrm{d}x\mathrm{d}y \\
&= \frac{2}{(n+1)(n+2)} \sum_{0 \leq j<i}^{N_z-1} \frac{z_i z_j}{(d_i-1)(d_j-1)} \times \\
&\quad [(C_i - C_{j+1} + 1)^{n+2} + (C_{i+1} - C_j - 1)^{n+2} \\
&\quad - (C_i - C_j)^{n+2} - (C_{i+1} - C_{j+1})^{n+2}] \\
&\quad + 2 \sum_{i=0}^{N_z-1} \frac{z_i^2}{(d_i-1)^2} \int_{x=0}^{d_i-1} \int_{y=0}^{x} (x-y)^n \mathrm{d}x\mathrm{d}y
\end{aligned}$$

The result follows on evaluating the integral. $\diamondsuit$

The seek time is calculated according to the widely accepted formula of Lee [12] :

$$S_i(D) = \begin{cases} 0 & \text{if } D_i = 0 \\ a\sqrt{D_i} + b(D_i - 1) + c & \text{otherwise} \end{cases}$$

where a, b, c are hardware-related constants :

$$\begin{aligned}
a &= (-10 \times MinSeek + 15 \times AvgSeek \\
&\quad -5 \times MaxSeek)/(3 \times \sqrt{C}) \\
b &= (7 \times MinSeek - 15 \times AvgSeek \\
&\quad +8 \times MaxSeek)/(3 \times C) \\
c &= MinSeek
\end{aligned}$$

and the three first moments required for the response time moment calculation on the Multi-RAID system are, as in [28]:

$$\begin{aligned}
\overline{S} &= (c-b) + aM_{1/2} + bM_1 \\
\overline{\overline{S}} &= (c-b)^2 + 2a(c-b)M_{1/2} \\
&\quad +[a^2 + 2b(c-b)]M_1 + 2abM_{3/2} + b^2M_2 \\
\overline{\overline{\overline{S}}} &= (c-b)^3 + a(c-b)^2M_{1/2} \\
&\quad +3(c-b)[a^2 + b(c-b)]M_1 \\
&\quad +[a^3 + 6ab(c-b)]M_{3/2} \\
&\quad +3[a^2b + b^2(c-b)]M_2 + 3ab^2M_{5/2} + b^3M_3
\end{aligned}$$

4 Results and discussion

We calculated and compared the first three moments of the seek time using the linear seek distance approximation [28], the non-linear seek distance calculation of proposition 1 and real system simulation using our Multi-RAID system simulator. The system modelled was composed of 16 disks with 200,000 simulated small logical requests arriving at different rates from 10req/s to 1000req/s. For our experiments, two kinds of zoned disks were used: a fujitsu-MAN3367 and a fictitious one with a highly non-linear relationship between sectors per track and cylinder number – see the characteristics in table 2.

It is not unusual to use a fictituous component to validate an extreme case. For example, [22] used a hypothetical fast disk, called Uberdisk, the parameters of which were scaled from contemporary disks, to approximate the performance of a MEMstore. Here, we use a fictitious disk as well, parameterized in terms of capacity (total number of Gigabytes and number of sectors), performance (rotation time and minimum/average/maximum seek times) and morphology (numbers of data heads and cylinders). These were scaled from a real disk (Fujitsu-MAN3367), apart from a reduced number of zones, to yield a significantly non-linear distribution of seek distance. This is to show the accuracy of the approximation as well as the value of the exact calculation for disks with a non-linear morphology. Figures 3 and 4 show that the number of zones changes from 18, on the Fujitsu disk, to 4, on the fictitious disk, but that the distribution of storage density changes drastically from linear to clearly non-linear, as intended.

The three first seek moments using the Fujitsu-MAN3367 disk in table 3 highlight the accuracy of the linear approximation for this kind of device. In fact, on such devices with a certain linearity of the recorded density distribution, the distribution of sectors across the zones shows consequently a certain linearity which makes the linear approximation and the non linear 'exact' calculation close to each other, matching very well with the simulation results. However, we notice the small advantage of the latter, nonlinear calculation.

Param.	value	value
	Fuj-MAN3367	Fictif
capacity	36,74 GB	36,74 GB
Sectors	$18,37 \times 10^7$	$18,37 \times 10^7$
Rotation	10000 rpm	10000 rpm
Cylinders	29950	29950
Min Seek	0,4 - 0,6 ms	0,4 - 0,6 ms
Avg Seek	4,5 - 5 ms	4,5 - 5 ms
Max Seek	11 - 12 ms	11 - 12 ms
Data Heads	4	4
Zones nbr	18	4

Table 2: Fujitsu-MAN3367 Vs Fictitious disks characteristics

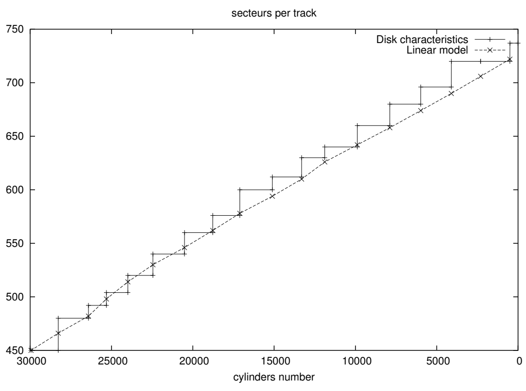


Figure 3: Sectors per track (FujitsuMAN3367)

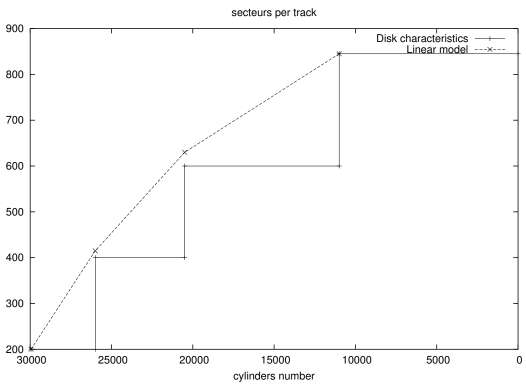


Figure 4: Sectors per track (Fictitious disk)

This linear distribution of sectors across the zones is not always respected, however. Even when it is, the seek time does not depend solely on the seek distance but also on the start position (or end position) of a seek. This is more important when these two positions belong to different zones, which is generally the case for non-sequential accesses. For such an access profile, using an approximation for the seek time calculation introduces a delay, accumulating along the request stream and so generating a larger discrepancy against the real system. Even if we can consider the seek time approximation accurate enough for sequential access, we certainly cannot consider it thus for non-sequential access, due to the cumulative delays introduced in a sequence of accesses. The more these non-sequential accesses appear, the more significant is the re-positioning delay and the less accurate is the seek time approximation.

	Model approx. linear Di	Model calc. Non linear Di	Simul.
$\overline{S}$	4.7	4.72	4.73
$\overline{\overline{S}}$	28.54	28.56	28.61
$\overline{\overline{\overline{S}}}$	200.35	199.98	199.21

Table 3: Seek time moments comparison (Fujitsu-MAN3367

Hence, the linear approximation cannot always give accurate results regardless of the device's morphology. Consequently, we used the fictitious disk, with the same space storage capacity and performance characteristics, but with a non linear distribution of sectors on zones to calculate the superiority of the non-linear seek distance calculation against its linear counterpart, with the Multi-RAID system simulation as a reference. The results obtained in table 4 confirm this superiority.

	Model approx. (linear Di)	Model calc. (Non linear Di)	Simul.
$\overline{S}$	4.47	4.41	4.36
$\overline{\overline{S}}$	26.00	24.90	24.56
$\overline{\overline{\overline{S}}}$	176.26	163.85	161.35

Table 4: Seek time moments comparison (Fictitious disk)

5 Conclusion

In this paper, we have compared our previously published model for the moments of seek distance in RAID models, which approximated the number of sectors on a track as a linear function of the track's sequence number, against an exact, explicit calculation. We showed that the proposed approximation is

still accurate for devices showing a certain degree of linearity in the density of storage over zones. However, the results obtained show the clear superiority of the non-linear moments calculation, regardless of the disk morphology. Using this result, we can guarantee almost perfect accuracy in the seek time calculation for uniform disk block accesses, which is a major component of the I/O response time, especially for non-sequential request streams.
We thereby constructed an accurate disk array model and in the near future, we expect to extend it to account for the operation of the bus that connects disks to the controller, which is subject to congestion and a consequent bus-queue, introducing further delay at high request arrival rates.

REFERENCES

[1] E. Bachmat and J. Schindler. Analysis od methods for scheduling low priority disk drive tasks. In *ACM Sigmetrics*, 2002.

[2] S. Chen and D. Towsley. The design and evaluation of raid5 and parity striping disk array architectures. *Parallel and distributed computing*, 17, 1993.

[3] S. Chen and D. Towsley. A performance evaluation of raid architectures. *IEEE Transactions on Computers*, 45(10), October 1996.

[4] G. Gibson D. A. Patterson and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of SIGMOD Conference*, June 1988.

[5] G. Gibson D. A. Patterson, P. M. Chen and R. H. Katz. Introduction to redundant arrays of inexpensive disks (RAID). In *IEEE COMPCON*, 1989.

[6] Fujitsu. *Disk Drives. Products/Maintenance Manual*. Fujitsu, 2001.

[7] Peter Muth Guido Nerjes and Gerhard Weikum. Stochastic service guarantees for continuous data on multi-zone disks. In *Symposium on Principles On Database Systems*, 1997.

[8] P.G. Harrison and S. Zertal. Queueing models with maxima of service times. In *Proceedings of TOOLS Conference*, 2003.

[9] P.G. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation*, 2007.

[10] M. Y. Kim and A.N. Tantawi. Asynchronous disk interleaving: approximating access delays. *IEEE Transactions on Computers*, 40(7), 1991.

[11] A. Kuratti and W. H. Sanders. Performance analysis of the raid5 disk array. In *Proc. IEEE Int'l Computer Performance and dependability Symp.*, 1995.

[12] E.K. Lee. *Perfomance modelling and analysis of disk arrays*. ph.D. thesis, University of California, Berkeley, USA, 1993.

[13] E.K. Lee and R.H. Katz. An analytic performance model of disk arrays. In *Proc. ACM SIGMETRICS*, May 1993.

[14] G. A. Alvarez M. Uysal and A. Merchant. A Modular, Analytical Throughput Model for Modern Disk Arrays. In *Proceedings of the International Symposium onModelling, Analysis and Simulationof Computer and Telecommunications Systems (MASCOTS)*, August 2001.

[15] J. Menon. Performance of raid 5 disk arrays with read and write caching. *Distributed ND Parallel Databases*, 2(3), July 1994.

[16] A. Merchant and P.S. Yu. An analytical model or reconstruction time in mirrored disks. *Performance evaluation*, 20, May 1994.

[17] A. Merchant and P.S. Yu. Analytic modeling and comparisons of striping strategies for replicated disk arrays. *IEEE Transactions on Computers*, 44(3), March 1995.

[18] A. Merchant and P.S. Yu. Analytic modeling of clustered raid with mapping based on nearly random permutation. *IEEE Transactions on Computers*, 45(3), March 1996.

[19] R. Nelson and A. N. Tantawi. Approximate analysis of fork-join synchronisation in parallel queues. In *IEEE Trans. Computers*, volume 37, June 1998.

[20] S. Christodoulakis P. Triantafillou and C. A. Georgiadis. A Comprehensive Analytical Performance Model for Disk Devices Under Random Workloads. *IEEE Transactions on Knowledge and data Engineering*, 14(1), 2002.

[21] Sangsoo Park and Heonshik Shin. *Rigourous Modeling of Disk Performance for Real-Time Applications*, volume 2986. Springer Berlin, 2004.

[22] Steven W. Scholosser and Gregory R. Ganger. Mems-based storage devices and standard disk interfaces: A square eg in a round hole. Technical Report CMU-PDL-03-102, Carnegie Mellon University, December 2003.

[23] seon ho kim Shahram Ghandeharizadeh and cyrus Shahabi. Continuous display of video objects using multi-zone disks. Technical Report 94-592 USC, University of South California, April 2003.

[24] S. Varma and A. M. Makowski. Interpolation approximations for symmetric fork/join queues. In *Performance Evaluation Journal*, volume 20, 1994.

[25] Jun Wang and Yiming Hu. Profs - performance-oriented data organization for log-structured file system on multi-zone disks. In *9th Internantional Symposium on Modeling, Analysis and Simulation on Computer and Telecommunication Systems*, 2001.

[26] S. Zertal. *Dynamic redundancy mechanisms for storage customisation on multi disks storage systems*. ph.D. thesis, University of Versailles, France, January 2000.

[27] S. Zertal and P.G. Harrison. Multi-level raid storage system modelling. In *Proceedings of 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (Spects)*, 2003.

[28] S. Zertal and P.G. Harrison. Multi-raid queueing model with zoned disks. In *High Performance Computing and Simulation*, 2007. Best paper Award.

VIRCONEL: A NEW EMULATION ENVIRONMENT FOR EXPERIMENTS WITH NETWORKED IT SYSTEMS

Yacine Benchaïb and Artur Hecker
Department of Computer Science and Networking
TELECOM ParisTech (ENST)
37-39, rue Dareau, 75014 Paris, France
E-mail: {benchaib, hecker}@enst.fr

KEYWORDS
Modeling, Simulation and Evaluation Techniques ; Grid and Cluster Computing.

ABSTRACT

In this paper we present VIRCONEL, a new, open-source emulation environment for experiments with and evaluation of networked IT systems. Based on previous open-source projects, VIRCONEL proposes a graphical modeling interface with node template support, entity cloning, IP configuration auto-completion and an easy scenario definition with label-based multiple role assignment and local script execution on virtual machines. Moreover, VIRCONEL has a graphical interface for the control of the deployed virtual network, allowing in particular one click logins, monitoring and value recording, as well as link and node fault injection. Most importantly, VIRCONEL easily installs on typical PC hardware and features explicit support for multiple physical hosts, thus providing a better scaling. Multiple physical hosts are seamlessly supported both in the virtual network design and operation phases.

INTRODUCTION

The evaluation of large, networked IT systems often raises questions with regard to the best evaluation environment. This is a known issue in the evaluation of performance, robustness and assurance properties of distributed systems and applications, new distributed maintenance algorithms, middleware architectures, P2P proposals, etc. (Jiang and Xu 2003). The problem is that formal approaches are either very difficult to apply or need certain assumptions that are difficult to verify in practice. The alternative is the experimental evaluation. Real testbeds are attractive because they are often more representative than other experimental evaluations. However, they inflict a high administrative burden (deployment, maintenance, operational effort). In practice, this results in prohibitive limitations of evaluable system sizes. On the other hand, classic simulation techniques can easily deal with thousands of nodes. Yet, they impose a controversial tradeoff between precision, complexity and control (Bavier et al., 2006). When left at the consideration of the author alone, this deserves doubts with respect to the trustworthiness of the results (Pawlikowski, 2002).

Emulation using virtual networks and virtual machines represents an interesting alternative to the experimental evaluation techniques (Ruth et al. 2005). Note that in this paper, we do not distinguish between different virtualization technologies and use the terms virtualization and emulation in their broad sense. See (Nanda and Chiueh, 2005) for background details.

Essentially using the same software as real testbeds but in virtual execution environments, emulation is very close to real, at least regarding local node behavior. Differences are in the performance and capacities of virtual nodes, especially when several virtual machines share one physical host. More importantly, there can be substantial differences in the link behavior. This is essentially comparable to simulation issues: e.g. accepted models are necessary to simulate a wireless link. Still, emulation can represent an attractive alternative to real testbeds and simulations. First, emulation by virtualization features binary compatibility with the real testbed and therefore, unlike simulations, does not need an additional model programming. This also permits for closed-code execution as emulated instances, thus allowing evaluation of commercial software whose behavior might be not completely known. Besides, since model programming is not necessary, and the software to be evaluated can be used directly, this avoids a potential error or imprecision source, thus yielding results closer to real than the simulation. Second, emulation can be used to evaluate networked IT environments composed of several hundreds of nodes with a relatively low deployment and operations effort in comparison to a full testbed, and this in a fully controlled environment. However, to do this, we need to supply the evaluator with a toolbox permitting to easily set up and control different virtual environments spanning over several real hosts.

In this paper, we present the design and implementation of Virtual Computer Network Lab (VIRCONEL), an open-source, easy-to-use, multi-host, networked emulation environment for the evaluation of large, networked IT systems with the help of several off-the-shelf PCs. VIRCONEL features a very easy installation method and explicitly supports several physical hosts for better scalability. VIRCONEL features graphical interfaces for the design of the system, emulated service deployment, scenario definition and emulation control. In operation, VIRCONEL can record typical parameters and the data as requested from the emulated network.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

The rest of the paper is organized as follows. In the next section, we present our rationale and the resulting requirements. We then present previous work in this area and explain the motivation for the design and development of VIRCONEL. Then, we explain its architecture and justify some of our decisions by providing insights on the related development effort. Next, we present the possibilities already provided for emulated system design, control and measurement. After that, we demonstrate the resource usage of VIRCONEL when running typical scenarii on our hardware and try to estimate its limits. Finally, we give an outlook to our future work.

RATIONALE AND REQUIREMENTS

The main drive for this work comes from the need for the setup and evaluation of large, distributed IT systems within the scope of the ICT FP6 DESEREC project. The DESEREC testbed needs to be capable of hosting different types of networked enterprise IT systems, often running complex, commercial closed-source software. Typical services within such systems include VoIP sessions between different locations, Web-based access to SQL/LDAP databases etc., provided over several LANs connected by routers over VPNs and the Internet and completed by obligatory security, reliability and management subsystems.

In practice, this translates to very close to real testbed of potentially several hundreds of nodes and a strong accent on the local applications and node behavior, i.e. being capable of running common open and commercial software under close-to-real constraints (Unix-like OS, TCP/IP plus NAT+DHCP, firewalls, NAS and AAA, etc.). Besides, to evaluate system behavior and resilience faced with node and communication breakdowns, we needed a possibility for a scenario generation, including failure scenarios. Finally, to allow collaborative partner work, the ease of installation of the emulation environment per se and the support for the interconnection of several local environments are considered important.

In summary, we distilled our wish list for an emulation package for large IT system evaluation to the following concrete MUST requirements:

-- *Open-source emulation software:* the emulation toolbox itself should not involve complicated licensing issues and be based on open-source software;

-- *Relative ease of installation:* complex installations on real hosts would be unattractive since the goal is to install the testbed and not to maintain the physical host;

-- *Support of several physical hosts:* one physical host usually cannot run more than several dozens of virtual machines. To support scaling to several hundreds of emulated nodes, we therefore need a possibility to easily support multiple, networked physical hosts. This support also needs to be integrated with the modeling of the network, i.e. it should be possible to assign virtual nodes to physical hosts;

-- *Ease of scenario definition:* the modeling of the emulated system and services upon it should be easy, preferably supported by an easy-to-use graphical tool. It should be possible to start the defined emulated network upon the available physical hosts;

-- *Binary compatibility with the existing software:* it should be possible to evaluate the existing software without understanding how it works (Ruth et al., 2005). As explained before, this has a double advantage of permitting direct usage of the existing software, e.g. closed source. What is more, it significantly reduces the modeling time, since one does not need to model local node behavior. This removes a potential error source.

-- *Running emulation monitoring and control:* it should be possible to influence a running emulation by provoking node and communication breakdowns, starting and stopping software on virtual machines, reconfiguring interfaces, etc. On the other hand, it should be possible to see what is happening within the emulation and, in particular, capture and record values of different interesting variables, limiting the perturbation of the emulated virtual reality.

In principle, our list corresponds to the requirements stated in (Bavier et al., 2006), which mainly underlines realism (real software, realistic conditions, real traffic) and controllability.

RELATED WORK

Network experiments are conducted today mainly through simulations with NS-2, OMNET++, Glomosim or commercial tools like Opnet. As discussed above, without a substantial additional effort, the simulation tools do not allow direct execution of closed-source, (e.g. commercial) software.

PL-VINI (Bavier, 2006) running over PlanetLab has been proposed for similar purposes, but access to PlanetLab is not always suitable.

Different “local” virtualization environments with networking support have been proposed, including commercial environments like VMWare and Parallels, and open-source projects like Qemu (Bellard, 2005), openVZ (http://openvz.org/), Xen (Barham, 2003) and User-Mode Linux (UML) (Dike, 2006) . Besides the previously cited survey (Nanda and Chiueh, 2005), an up-to-date comparison of these can be found in Wikipedia under “Comparison_of_virtual_machines”.

A substantial work has been done on server virtualization and containment (see e.g. Padala, 2007). However, the aim of this work is on the one hand on the performance and high availability of any single server, and on the hand, on the management of such servers. We would like to place as many virtual or paravirtual instances on any single physical host so as to make our network emulation scale. We also need the control, but focus on scenario control and monitoring, rather than on the service management (patches, security updates, user/account management, etc.), typical for real servers. Besides, we need a tool to graphically define virtual topologies.

Several open-source projects add virtual networking support to virtual machines. For instance, Netkit (http://www.netkit.org/) is a collection of shell scripts

for instantiating a virtual network of UML-based virtual machines. VN-UML (Galan, 2004) and MLN (http://mln.sourceforge.net/) support structured descriptions of the network to be set up on one machine, with MLN also supporting Xen and UML combinations. Graphical editors emerged for such structured descriptions, like NetGUI (Nemesio, 2006), vnumlgui (http://pagesperso.erasme.org/michel/vnumlgui/) producing VN-UML's XML. Marionnet (Loddo, Saiu 2007), principally accentuating on didactics and dedicated to teaching, adds dynamic network reconfiguration support.
Having studied the related work, we concluded that very interesting building blocks exist in the open-source community. On the other hand, no proposal permitted to fulfill all of our requirements. Especially the graphical editors building upon VN-UML generally come close to our requirements. However, all of them are limited to one physical host, both in the modeling and in the emulation execution phases. Second important point (Bavier et al., 2006): they generally do not integrate monitoring and control utilities. Marionnet features support for topology changes in operation. However, it targets education purposes and is rather committed to realism when working with small networks, while we would like to simplify modeling work.

DESIGN AND IMPLEMENTATION OF VIRCONEL

Profiting from previous experiences and trying to reduce precious development time, VIRCONEL relies upon and extends VN-UML (Galan, 2004).
VN-UML uses UML as virtual machines. UML is in principle a Linux kernel started in the user-space, as any other process. It can thus be stopped and interrupted at any time. For networking support, the VMs make use of the Linux kernel's ability to provide virtual network interfaces (*tun* and *tap* devices). The interface of the UML VM (*eth0*) connects to such a virtual interface of the host Linux. By defining proper IP forwarding and/or bridging rules, any VM can get customized network access. To simplify the necessary configuration, VN-UML introduces a virtual switch and a structured definition language, which defines the interconnects of VMs with each other and the physical host. Isolation is possible through the use of VLANs (see Figure 1, PC1).

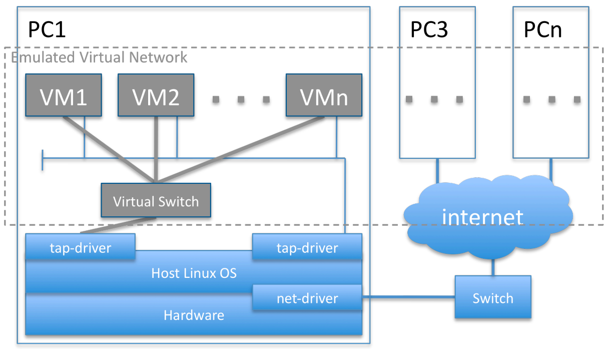


Figure 1: System architecture of VIRCONEL

In VIRCONEL, the VN-UML virtual switch is currently bridged to the real network device of the host Linux in a VLAN. If available physical hosts are interconnected by a real network (e.g. switch, router, VPN gateway, the Internet), the virtual network can span over these hosts, as shown in Figure 1. Currently this is done with limitations on topology but in principle, different isolated virtual networks can be set up with known real network separation measures (e.g. VLAN or VPN). However, this is not an urgent requirement for us.
Besides, any VM can be configured with several interfaces. An interesting point in VN-UML is the explicit presence of an additional interface, used for direct communications with the hosting physical machine. We call this interface "management interface". In VIRCONEL, we use it for operational control, scenario deployment, measurement traffic, etc.
The emulated machines are Linux kernels working over a specific file on a host PC as a shared partition. By installing software in this "partition" and preparing different partition files, we provide VM templates. The software installed within the latter can also be started and controlled over the management interface.
In principle, to run an emulation, four phases are necessary in VIRCONEL after it has been installed:
-- virtual network modeling,
-- virtual scenario definition and setup,
-- deployment of virtual entities to physical hosts, and
-- virtual network operations.
For modeling, we use a popular open-source graphical editor called *Dia* that produces XML output (http://www.gnome.org/projects/dia/). We integrated Dia by adding a VIRCONEL-specific workbench, permitting the choice of different typical entities (switches, routers, hosts). VIRCONEL's parser processes Dia's XML output and translates it into a VN-UML XML input file, producing one XML file per specified physical host as mentioned in the graphical model. Scenario setup is described in the next section.
In the deployment phase, these files are then distributed to and executed on the available physical hosts, as specified in the model file. We use SSH for that.
In the operational phase, VIRCONEL starts a control panel that displays the emulated network within one graphical interface (developed in Tcl/Tk), permitting scenario start, local login to every virtual machine, activating/deacting links, etc.
This design and the reuse of the previous work have permitted us to accomplish a first working version of VIRCONEL in about five men-months of integration work. In the following, we describe the usage and features of VIRCONEL.

USAGE AND FEATURES OF VIRCONEL

Local Installation

Similarly to VN-UML, we use the Live-DVD concept permitting a very easy local deployment on a spare physical host. This allows concentrating on the essential, emulation-related things.

However, since UML, unlike other virtualization technologies, does not necessarily require host machine changes, VIRCONEL can also be installed and executed on an available Linux host used for other tasks.

Modeling: Virtual Topology Definition

Modeling is done within the integrated graphical editor (Dia, slightly modified). Currently, VIRCONEL comes with switch, router and host templates. The existing templates, available as icons in Dia, can be positioned on the screen and interconnected by links as necessary. Graphical links represent emulated network links.
The provided host template comes with a variety of typical applications, including Web server, client, SIP instances, etc. It is possible to change the existing/to add new VM templates at any time (and to integrate them into the graphical tool).
To further simplify things, we explicitly support starting and stopping processes on any operational VM from the modeling phase on. This is used for assigning roles for the scenario definition (see below), but also renders any usage of the existing templates more flexible, since the same template can be used to instantiate semantically different VMs (e.g. a server and a client).

Setup: Scenario Definition

To assign such roles and/or configuration parameters, we use a simple *labeling* technique. Designer can attach a number of text labels to any existing basic entity. When all labels are assigned, the designer simply groups all labels with the original entity using Dia's grouping function. This attaches the labels to this specific entity. Therefore, designer can define IP addresses, specify which processes should be started, etc. We also provide some support for rapid modeling, namely *IP-configuration auto-completion* and simple entity *cloning*. The auto-completion function can automatically find the responsible router from the XML topology file. Hence, attaching an IPv4 address in CIDR to each host is sufficient. Cloning is very useful to produce high numbers of identical hosts. Currently, it is possible to attach a <clone=N> label to a well-defined host in order to clone the latter N times (thus resulting in N+1 identical entities with the same behavior). The IP addresses of the emulated interfaces are automatically renumbered within the subnetwork space.
The scenario per se is defined through labels, which identify scripts to be executed on each concerned modeled entity. More precisely, the <*-Client> and <*-Server> labels are interpreted as parameters to a launcher script. The latter searches and executes the script with the same name within the targeted VM. Thus, the scenario definition needs a machine pre-provisioning with all required executables (template, or copying by hand in the operational machine), various script placements on the machine (with names corresponding to the labels) and the definitions of resource consumption models (e.g. period, number of bytes to send, etc.) within the scripts.
Such scripts can be developed by the designer and are very flexible (basically, shell script, perl, python etc.). They permit to define any typical scenario, for instance a number of Web clients accessing a multi-tiered Web server with certain distributions, etc. (Padala, 2007). Since any script and binary execution is supported on virtual machines, this approach does not constrain the possibilities. A list of currently supported labels with their semantical meaning is given in Table 1 but is being constantly worked on.

Table 1 Currently supported VIRCONEL labels

Label	
*-Client	Used to start a client script on the VM
*-Server	Used to start a server script on the VM
Clone	Clone a specified virtual machine.
IP address	Define the VM's IP address (IP/mask)

Deployment

We support multiple physical hosts in modeling, setup and execution phases. The designer needs to assign virtual machines to physical servers. In VIRCONEL, this can be done by enclosing a required number of subnets/hosts and a router into a graphical rectangle in Dia. The designer then specifies the IP address of the physical host as shown in Figure 2. Once these phases are accomplished, the emulation can be started by parsing the produced output file. This locally starts the operational GUI, which exactly represents the whole modeled topology hiding the physical hosts as can be seen in Figure 3. It permits to start/stop both the virtualization and the defined scenario and has some other features to be described in the next phase.
Driven over this GUI, which uses SSH from the designer host to physical hosts, VIRCONEL first distributes the designed virtual topology within the specified testbed and then initiates the virtual network entities necessary to combine the subnetworks hosted on different physical machines. The testbed is composed of PCs, each of which is running VIRCONEL. The virtual machines are started on the physical hosts of the emulation platform as identified by their IP addresses.

Operation

Furthermore, the same GUI also permits to control the operation of the emulation. It is possible to launch and stop the defined scenario. Commands are sent over SSH from the designer host to each VM as specified.
Second, the operation GUI features a *one-click-login* to any virtual machine, which opens an SSH session from the designer host to the virtual machine's management interface. This is very practical for manual error introduction or for tests/measures and slight changes within the operational virtual environment.

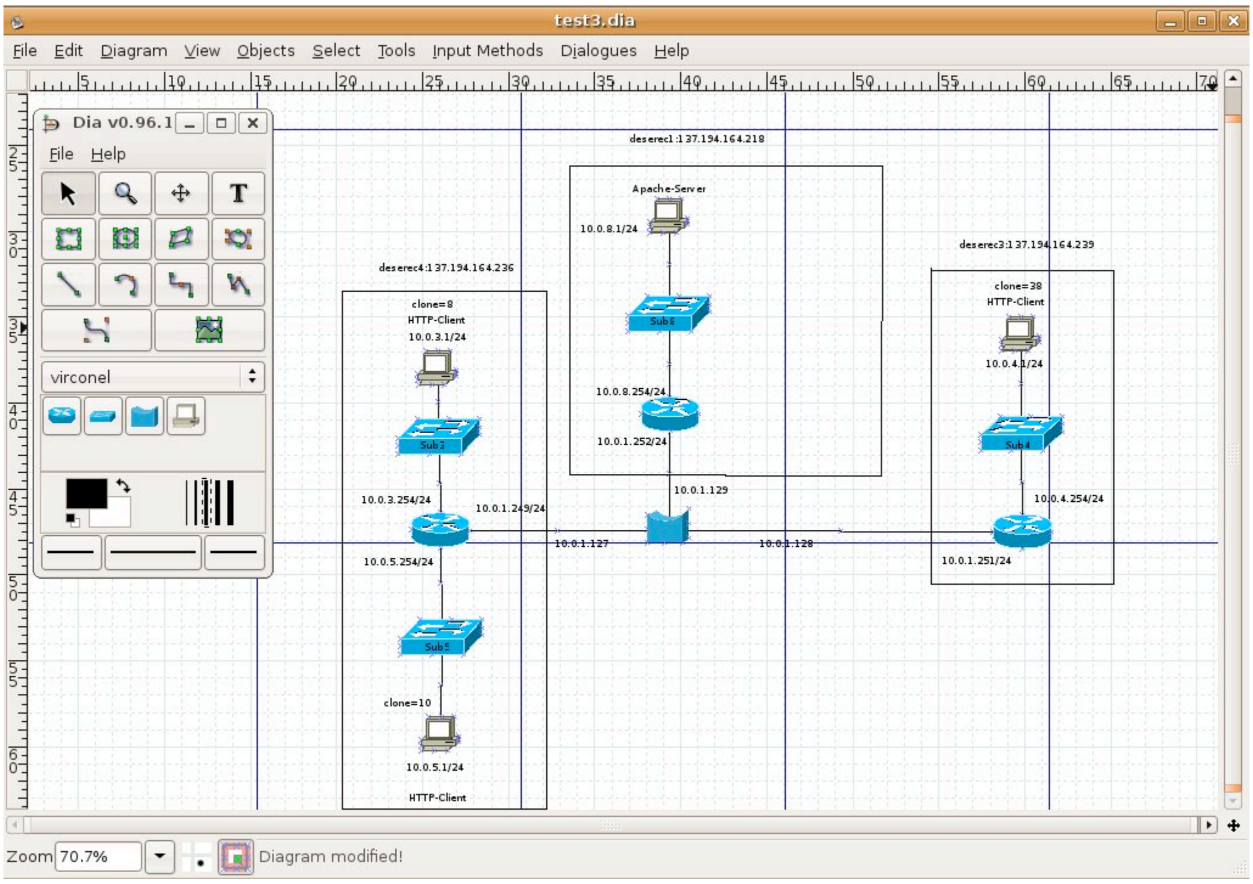


Figure 2: VIRCONEL modeling interface

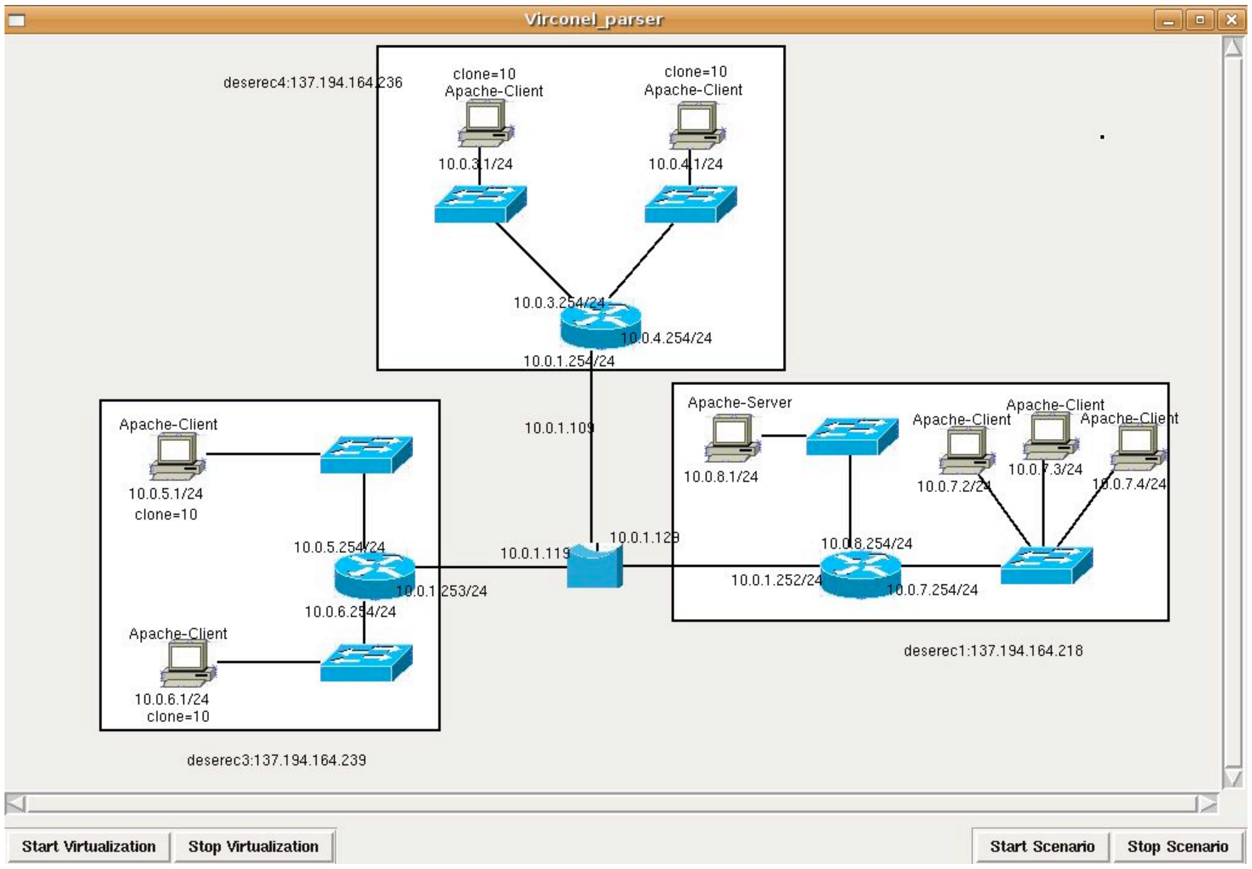


Figure 3: Control of the emulated network

The management interface permits to collect various measurements without perturbing the emulated network traffic. Per default, VIRCONEL assesses typical data, like overall CPU consumption and overall network traffic on every emulated interface, and represent these with *gnuplot*. Yet, more complicated measurements can be defined in the setup phase. In principle, whatever can be measured in the real network can be measured in VIRCONEL. The assessed data is either sent over the management interface to the operational GUI, or it is stored in the virtual or physical host partition. Resource usage measurements are also possible from the host PC.

EVALUATION RESULTS

Evaluation Testbed

The testbed on which we install VIRCONEL and run our evalution is composed of three servers (PC), each equipped with 4GB of RAM and a 2.6GHz QuadCore Intel CPU. The servers are running Ubuntu Linux 7.04 Feisty Fawn, kernel version 2.6.20 patched with the SKAS3 patch for better UML performance.

Evaluated Scenarios

We use three scenarios to evaluate VIRCONEL. The first scenario evaluates the computational penalty experienced by a process within the virtual machine. We want to find answers as to how much performance we lose per VM when running several VMs on the same host. This gives an estimate on how many concurrent VMs we can put on one Linux host.

We use *openssl* to symmetrically encrypt a 20MB binary file. We first sample the host Linux performance and then repeat the exact same command within the VM with concurrent 6, 11 and 16 VMs on the same host Linux. On our hardware, the host Linux performs this task (measured with *time*) in an average time of 1.048s with a standard deviation of 0.114 s. The VMs take an average of 2.5s (0.447) for 6, 3.265s (0.432) for 11 and 6.702s (1.381) for 16 concurrent VMs on one physical host respectively (standard deviation in brackets). The results of these measurements as percentage, normalized to fixed host performance, are shown in Figure 4.

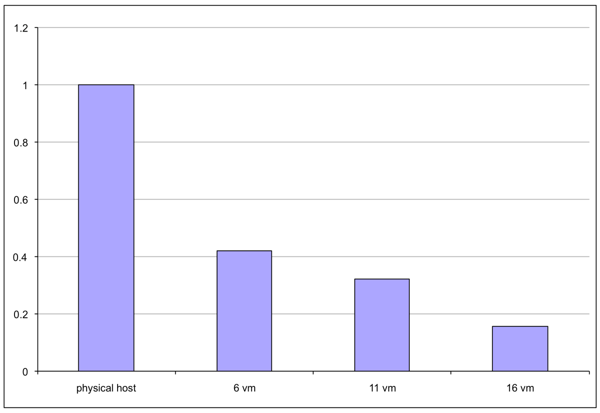


Figure 4: Computation performance penalty per VM on one host PC with the increasing number of VMs

We can see that at least up to 11 VMs can be used for this computationally intense task with a reasonable and stable penalty. For 16 VMs the results start varying too much because of complex interactions of concurrent processes with the task scheduling. Note that while the performance per VM decreases, the overall performance for at least up to 16 VMs is better than for the single process at the host Linux: while the host Linux takes 1.0457s per file encryption, 6 VMs take only 0.42s per file, 11 VMs take 0.296s per file and 16 VMs take 0.42s per file. We can see that 11 concurrent VMs have the best performance in that scenario.

In the second scenario, we use three physical hosts connected by a real switch (100BaseT). We emulate HTTP traffic from virtual clients (*wget*) to one virtual webserver (Apache). This roughly represents a mixed resource usage typical for a modern distributed application. Using Unix *at*, HTTP client starts on all client VMs simultaneously, sends an HTTP request to the webserver for a hosted file of 50kB and exits immediately. We use the topology as illustrated in Figure 2: we use 3 host Linux PCs, with the webserver and router being the only VMs on the host deserec2. There are 11 concurrent HTTP clients on deserec1 (plus router VM) and 16 HTTP clients on deserec4 (idem). We measure the delay for a succcesfull transaction from within the VM, i.e. the time from the start to the exit of *wget*. The averaged results for physically identitical hosts deserec1 (11 clients) and deserec4 (16 clients) in 20 experiments are illustrated in Figure 5.

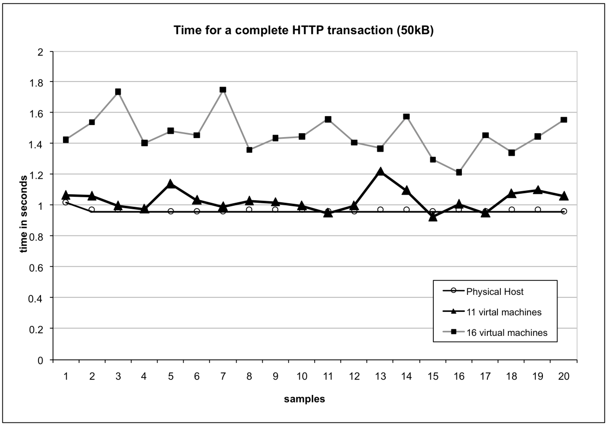


Figure 5: Time for a complete execution of an emulated HTTP transaction with 11 and 16 concurrent VMs

Third scenario is similar to the second one. We use HTTP traffic from virtual clients to one virtual server in the same topology (Figure 2). However, we start the clients periodically, with the inter-process invocation time from a uniform distribution in the interval [1s..2s], independent for every VM. The client starts, sends an HTTP request to the Web server for a hosted file of 50kB and exits. The concurrent process start/end produces a considerable I/O activity.

Under these conditions, we vary the overall number of client VMs on deserec1 and deserec4 host PCs and measure the CPU consumption on the physical host. The limit is reached for 27 VMs due to the frequent process starts and stops on the concurrent VMs. In Figure 6, we show CPU utilization on the host Linux under the number of concurrent VMs. For scenario 3 the increase is linear. Hence, in similar scenarios, it is possible to maintain a ratio of about 20 VMs per physical host.

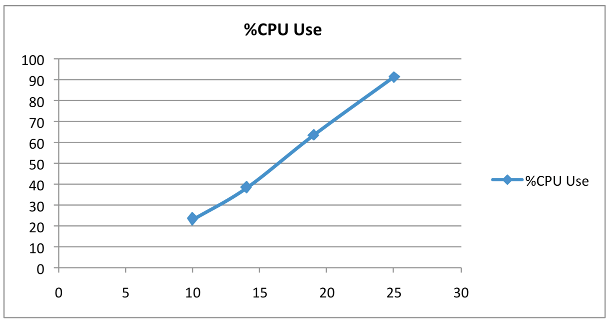


Figure 6: CPU usage on one physical host (Y axis) with different number of UML virtual machines (X axis)

Current VIRCONEL Limits

The usage of VN-UML and of UML technology imply several limits. First, VIRCONEL is a Linux-only environment, both for physical and virtual machines. Second, if with VIRCONEL it is possible to use several physical hosts, in practice this will reach management limits. Also, the assignment of VMs to physical hosts is done manually. However, the main limitation is within the topological constraints: it is currently necessary to specify a virtual router per physical host. For our work in DESEREC, this is not a serious problem. But we may consider this point in our further work.

CONCLUSION

VIRCONEL is a very easy to install and rather simple to use emulation environment for experiments with IT systems. Compared to the existing work, our main contributions are the intrinsic support for multiple physical hosts and integrated monitoring and control. With VIRCONEL, it is interesting to combine real and emulated entities. In that manner, resource-demanding entities can be treated separately, while numerous small entities can be easily cloned in the emulation.
In future we plan to improve the functionalities of VIRCONEL, namely its monitoring capabilities, tying these to the modeling. VIRCONEL can be freely (GPL) downloaded from http://www.infres.enst.fr/~deserec.

ACKNOWLEDGMENTS

This work is supported in part by the EC ICT FP6 DESEREC (CN 026600, www.deserec.eu) project. The authors are thankful to the anonymous reviewers whose valuable remarks helped to improve this work.

REFERENCES

Barham P.; Dragovic, B.; Fraser, K.;, Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield A. “Xen and the Art of Virtualization”, in proc. ACM HotNets-I, ACM Press, 2003, pp. 59-64.

Bavier, A; Feamster, N.; Huang, M.; Peterson, L.; and Rexford, J. “In VINI Veritas: Realistic and Controlled Network Experimentation”, in proc. ACM SIGCOMM 2006, Pisa, Italy.

Bellard, F. “Qemu, a Fast and Portable Dynamic Translator”, in proc. FREENIX Track of USENIX 2005 Annual Technical Conference, pp. 41-46.

Dike, J. “User Mode Linux”, Prentice-Hall, April 2006.

Galan, F.; Fernandez, D.; Ruiz, J.; Walid, O.; de Miguel, T. “Use of Virtualization Tools in Computer Network Laboratories”, in proc. 5th IEEE ITHET, June 2004.

Jiang, X.; Xu, D.. “vBET: a VM-Based Emulation Testbed”, in proc. ACM SIGCOMM 2003, Karlsruhe, Germany.

Loddo, J.-V.; Saiu, L. “Status Report: Marionnet. How to implement a Virtual Network Laboratory in Six Months and Be Happy”, in proc. ACM ML’07, Freiburg, Germany, October, 2007.

Nanda, S.; Chiueh, T.. “A Survey on Virtualization Technologies,” the Research Proficiency Report, Stony Brook, ECSL-TR-179, February 2005.

Nemesio, S. C.; de las Heras Quiros, P.; Barbero, E. M. C.; Gonzalez, J. C. “Early Experiences with NetGUI Laboratories”, SIIE’06, Leon, Spain, October 2006.

Padala, P.; Zhu, X.; Wang, Z.; Singhal S.; Shin, K. G. “Performance Evaluation of Virtualization Technologies for Server Consolidation”, HP Laboratories Technical Report, April 2007 (available online).

Pawlikowski K.; Jeong, H.-D. J. and Lee, J-S. R. “On Credibility of Simulation Studies Of Telecommunication Networks”, IEEE Communications Magazine, January 2002, pp. 132-139.

Ruth, Paul; Jiang, Xuxian; Xu, Dongyan; Goasguen, Sebastien. “Virtual Distributed Environments in a Shared Infrastructure”, IEEE Computer, May 2005, pp. 63-69.

AUTHOR BIOGRAPHIES

YACINE BENCHAÏB holds a Master in Computer Science from the Université d’Amiens, France. In 2007, he joined the networking and computer science department of TELECOM ParisTech (ENST), where he currently works as research engineer. His research work includes virtualization and network security. Contact him under `benchaib@enst.fr`.

ARTUR HECKER holds a diploma in Computer Science (Dipl.inform.) from the University of Karlsruhe (TH), Germany and a PhD degree in Computer Science and Networking from the ENST, France. He worked as CTO of Wavestorm SAS, which he co-founded in 2003. Since 2006, he is Associate Professor at the INFRES department at the ENST. His present research interests are wireless access security, autonomous networking and security assurance of complex systems. Dr. Hecker is actively involved in several IST FP6 and EUREKA CELTIC research activities. Contact him under `hecker@enst.fr`.

Resource Allocation, Sharing and Management in HPC Systems

Resource Sharing Usage Aware Resource Selection Policies for Backfilling Strategies

F. Guim [1], J. Corbalan [2] and J. Labarta [3]

Barcelona Supercompuning Center
Jordi Girona 13, Barcelona, Spain
[1]francesc.guim@bsc.es
[2]julita.corbalan@bsc.es [3]jesus.labarta@bsc.es

KEYWORDS

Resource Selection Policies, Resource Sharing Consideration, Backfilling, Job Scheduling

1 ABSTRACT

Job scheduling policies for HPC centers have been extensively studied in the last few years, specially backfilling based policies. Almost all of these studies have been done using simulation tools. All the existent simulators use the runtime (either estimated or real) provided in the workload as a basis of their simulations. In our previous work we analyzed the impact on system performance of considering the resource sharing of running jobs including a new resource model in the Alvio simulator.

In this paper we present two new Resource Selection Policies that have been designed using the conclusions reached in our preliminary work. First, the *Find Less Consume Distribution* that attempts to minimize the job runtime penalty that an allocated job will experience. Based on the utilization status of the shared resources in current scheduling outcome and job resource requirements, the LessConsume policy allocates each job process to the free allocations in which the job is expected to experience the lowest penalties. Second, we have also described the *Find Less Consume Threshold Distribution* selection policy which finds an allocation for the job that satisfies the condition that the estimated job runtime penalty factor is lower than a given value *Threshold*.

2 INTRODUCTION

Several works focused on analyzing job scheduling policies have been presented in the last decades. The goal was to evaluate the performance of these policies with specific workloads in HPC centers. A special effort has been devoted to evaluating backfilling-based (Chiang et al. (2002)Tsafrir et al. (2005)) policies because they have demonstrated an ability to reach the best performance results (i.e: Feitelson et al. (2004) or Talby and Feitelson (1999)). Almost all of these studies have been done using simulation tools. To the best of our knowledge, all the existent simulators use the runtime (either estimated or real) provided in the workload as a basis of their simulations. However, the runtime of a job depends on runtime issues such as the specific resource selection policy used or the resource jobs requirements.

In Guim et al. (2007) we evaluated the impact of considering the penalty introduced in the job runtime due to resource sharing (such as the memory bandwidth) in system performance metrics, such as the average bounded slowdown or the average wait time, in the backfilling policies in cluster architectures. To achieve this, we developed a job scheduler simulator (Alvio simulator) that, in addition to traditional features, implements a job runtime model and resource model that try to estimate the penalty introduced in the job runtime when sharing resources. In this work we have only considered in the model the penalty introduced when sharing the memory bandwidth of a computational node.

Results showed a clear impact of system performance metrics such as the average bounded slowdown or the average wait time. Furthermore, other interesting collateral effects such as a significant increment in the number of killed jobs appeared. Moreover the impact on these performance metrics was not only quantitative.

In this paper we describe two new resource selection policies that are designed to minimize the saturation of shared resources. The first one, the *Find Less Consume Allocation* (henceforth referred to as *LessConsume*) attempts to minimize the job runtime penalty that an allocated job will experience. It is based on the utilization status of shared resources in the current scheduling outcome and the job resource requirements. The second once, the *Find Less Consume Threshold Distribution* (henceforth referred to as *LessConsumeThreshold*) , finds an allocation for the job that satisfies the condition that the estimated job runtime penalty factor is lower than a given value *Threshold*. This resource selection policy has been designed to provide a more sophisticated interface between the local resource manager and the local scheduler in order to find the most appropriate allocation for a given job.

The rest of the paper is organized as follows: section 3 briefly introduce the resource and runtime models that we proposed; next, the two resource selection policies we propose are described; in section 6 we present their evaluation; and finally in section 7 we present the conclusions of this work.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

3 MODELING RUNTIME PENALTY IN ALVIO

In this section we provide a brief characterization for the runtime model that we designed for evaluate the resource sharing in the Alvio simulator. In Guim et al. (2007) we present a detailed description of the model and its evaluation.

3.1 The Job Scheduling Policy

The job scheduling policy uses as input a set of job queues and the information provided by the Local Resource Manager (LRM) that implements a Resource Selection Policy (RSP). It is responsible to decide which of the jobs that are actually witting to be executed have to be allocated to the free resources. To do this, considering the amount of free resources it selects the jobs that can run and it requires to the LRM to allocate the job processes.

3.2 The Resource Selection Policy

The Resource Selection Policy, given a set of free processors and a job α with a set of requirements, decides to which processors the job will be allocated. To carry out this selection, the RSP uses the Reservation Table (RT). The RT represents the status of the system at a given moment and is linked to the architecture. The reservation table is a bi dimensional table where the X axes represent the time and the Y axes represent the different processors and nodes of the architecture. It has the running jobs allocated to the different processors during the time. One allocation is composed of a set of buckets[1] that indicate that a given job α is using the processors $\{p_0,..,p_k\}$ from *start time* until *end time*.

An allocation is defined by: $allocation\{\alpha\} = \{[t_0,t_1], P = \{p_{\{g,n_h\}},..p_{\{s,n_t\}}\}\}$ and indicates that the job α is allocated to the processors P from the time t_0 until t_1. The allocations of the same processors must satisfy that they are not overlapped during the time.

3.3 Modeling the conflicts

The model that we have presented in the previous section has some properties that allows us to simulate the behavior of a computational center with more details. Different resource selection policies can be modeled. Thanks to the Reservation Table, it knows at each moment which processors are used and which are free.

Using the resource requirements for all the allocated jobs, the resource usage for the different resources available on the system is computed. Thus, using the Reservation Table, we are able to compute, at any point of time, the amount of resources that are being requested in each node.

[1]The $b_{(i,t_{i_0},t_{i_1})}$ bucket is defined as the interval of time $[t_x,t_y]$ associated to the processor p_i

In this extended model, when a job α is allocated during the interval of time $[t_x,t_y]$ to the reservation table to the processors $p_1,..,p_k$ that belong to the nodes $n_1,..,n_j$, we check if any of the resources that belong to each node is overloaded during any part of the interval. In the affirmative case a runtime penalty will be added to the jobs that belong to the overloaded subintervals. To model this properties we defined the Shared Shadows and the penalty function associated to it.

The Shared Windows

A *Shared Window* is an interval of time $[t_x,t_y]$ associated to the node n where all the processors of the node satisfy the condition that: either no process is allocated to the processor, or the given interval is fully included in a process that is running in the processor.

The penalty function

This function is used to compute the penalty that is associated with all the jobs included to a given Shared Window due to resources overload. The input parameters for the function are:

- The interval associated to the Shared Window $[t_x,t_y]$.
- The jobs associated to the Shared Window$\{\alpha_0,..,\alpha_n\}$
- The node n associated to the Shared Window with its physical resources capacity.

The function used in this model is defined as [2]:

$$\forall res \in resources(n) \rightarrow demand_{res} = \sum_{\alpha}^{\{\alpha_0,..,\alpha_n\}} r_{\alpha,res} \quad (1)$$

$$Penalty = \sum_{resources(n)}^{res} \left(\frac{\max(demand_{res}, capacity_{res})}{capacity_{res}} - 1\right) \quad (2)$$

$$PenlizedTime = (t_y - t_x) * Penalty \quad (3)$$

First for each resource in the node the resource usage for all the jobs is computed. Second, the penalty for each resource consumption is computed. This is a linear function that depends on the overload of the used resource. Thus if the amount of required resource is lower than the capacity the penalty will be zero, otherwise the penalty added is proportional to the fraction of demand and availability. Finally, the penalized time is computed by multiplying the length of the Shared Window and the penalty. This penalized time is the amount of time that will be added the node penalized time to all the jobs that belong to the Window. This model has been designed for the

[2]Note that all the penalty, resources, resource demands and capacities shown in the formula refer to the node n and the interval of time $[t_x,t_y]$. Thereby, they are not specified in the formula

memory bandwidth shared resource and can be applicable to shared resources that behave similar. However, for other typology of shared resources, such as the network bandwidth, this model is not applicable. Future work will be focused on modelizing the penalty model for the rest of shared resources of the HPC local scenarios that can impact in the performance of the system.

For compute the penalized time that is finally associated to all the jobs that are running: first, the shared windows for all the nodes and the penalized times associated with each of them are computed; second the penalties of each job associated with each node are computed adding the penalties associated with all the windows where the job runtime is included.

4 THE LESSCONSUME RESOURCE SELECTION POLICY

The core algorithm of this selection policy is similar to the the First Fit resource selection policy. This last one selects the first $\alpha_{\{CPUS,p\}}$ where the job can be allocated. However, in contrast to this previous algorithm, the LessConsume policy, once the base allocation is found, the algorithm computes the penalties associated with the different processes that would be allocated in the reservation. Thereafter it attempts to improve the allocation by replacing the selected buckets (used for create this initial allocation) that would have higher penalties with buckets that can be also selected, but that have not been evaluated. The LessConsume algorithm will iterate until the rest of the buckets have been evaluated or the penalty factor associated to the job is 1 (no penalty). [3]

The LessConsume policy, given a required start time t_{req} and given the job α with a requested runtime of $\alpha_{\{RunTime,rt\}}$ and number of requested processors $\alpha_{\{CPUS,p\}}$, finds in the reservation table the $\alpha_{\{CPUS,p\}}$ processor allocation that tries to minimize the job runtime penalties due to resource sharing saturation closest to the t_{req}. To do this, the selection policy follows these steps:

1. For each processor p_i in the reservation table, the resource selection policy selects the interval of time $[t_{x_i}, t_{y_i}]$ that is closest to the t_{req} that satisfies it: no process is allocated to the processor during the interval and its length is equal or greater to $\alpha_{\{RunTime,rt\}}$. All the buckets associated to the selected intervals of time are added to the set *Buckets* where they are strictly ordered by the interval start time.

2. Given all the different buckets $\left\{b_{(1,t_{1_0},t_{1_1})},..,b_{(N,t_{N_0},t_{N_1})}\right\}$ associated with the reservation table that are included in the set *Buckets*, the resource selection policy will select the first $\alpha_{\{CPUS,p\}}$ buckets that satisfy the condition that their interval of time shares at least the runtime required by the job.

3. In the case that in step *2* the number of buckets that satisfied the required conditions was lower than the required processors, this implies that there were not enough buckets which shared the required amount of time. In these situations, the first bucket $b_{(i,t_{i_0},t_{i_1})}$ with start time greater than to t_{req} is selected, the t_{req} is updated as $t_{req} = t_{i_0}$ and the steps *1*, *2* and *3* are iterated again.
   At this point, from the buckets obtained in the first step we will have three different subsets:

   - The buckets $\Pi_{disc} = \left\{b_{(k,t_{k_0},t_{k_1})},..,b_{(l,t_{l_0},t_{l_1})}\right\}$ that have already been selected since they cannot form part of a valid allocation for the specified requirements.
   - The buckets $\Pi_{sel} = \left\{b_{(m,t_{m_0},t_{m_1})},..,b_{(n,t_{n_0},t_{n_1})}\right\}$ that have been selected for the base allocation and that conform to a valid allocation which satisfy the requirements for the job. In the case that the penalties of this allocation cannot be improved by a valid allocation, this set of buckets will be used.
   - The buckets $\Pi_{toCh} = \left\{b_{(o,t_{o_0},t_{o_1})},..,b_{(p,t_{p_0},t_{p_1})}\right\}$ that have not already been evaluated. These buckets will be used to try to improve the base allocation. Thus, the LessConsume policy will try to find out if any of these buckets could replace any of the already selected buckets reducing the estimated penalty factor of the job.

4. For each of the buckets in the set of the selected buckets Π_{sel}, the algorithm checks the estimated penalty that a process of the given job would achieve if it were allocated to the given bucket. Each of the selected buckets has an associated penalty factor. If all the buckets have an associated penalty of *1* the allocation definition based on these buckets is defined and returned. Otherwise, the *last valid allocation* is initialized based on this set of buckets and the selection process goes ahead.

5. For each bucket $b_{(r,t_{r_0},t_{r_1})}$ in the set of buckets Π_{toCh}:

   (a) If the number of buckets in the set Π_{sel} is lower than the number of requested processors, the bucket is added to the set and the next iteration continues.

   (b) Similar to the previous step, the algorithm evaluates the penalty $penalty_r$ that a process of the job would have if this bucket were used to allocate it.

   (c) The bucket $bMax_{(p,t_{p_0},t_{p_1})}$ of the set Π_{sel} with the highest penalty $penalty_p$ is selected.

[3]The penalty factor is computed: $PenaltyFactor_{\alpha} = \frac{\alpha_{\{RunTime,rt\}}+\alpha_{\{PenalizedRunTime,prt\}}}{\alpha_{\{RunTime,rt\}}}$

(d) In case that the penalty that $penalty_r$ is lower than the penalty $penalty_p$, on one hand the bucket $b_{(r,t_{r_0},t_{r_1})}$ is added to the set Π_{sel} and removed from the set Π_{toCh}. On the other hand, the bucket $bMax_{(p,t_{p_0},t_{p_1})}$ is removed from the Π_{sel} and added to the set Π_{disc}. Note that at this point the inserted bucket may not share the interval required to allocate the job to the rest of the buckets of the set Π_{sel}.

i. The buckets of the set Π_{sel} that do not share the required time are removed from this set and added to the Π_{disc}.

ii. If the number of buckets of the Π_{sel} it is the number of requested processors *the last valid allocation* is built based on this set. If the current penalty of all the buckets is 1, the current set of buckets is returned as the allocation. Otherwise the algorithm goes ahead with the next iteration to evaluate the next bucket of the set Π_{toCh}.

6. The *last valid allocation* is returned.

As an example, suppose that the current scheduling outcome is the one presented in the figure 1 and that an allocation has to be found for a job with requested processors $\alpha_{\{CPUS,3\}}$ and runtime $\alpha_{\{RunTime,20secs\}}$.

1. Firtsly, in the step *1* the LessConsume algorithm would define the set $\Pi_{toCh} = \{b1-3, b7, b17, b4-6, b11-12, b14-16, b13, b18\}$.

2. In the step *2* the three buckets *1*, *2* and *3* would be selected. Note that all of them satisfy the condition that they all share an interval of time greater than the required runtime.

3. In the step *3* a valid allocation is created with the buckets selected in the previous step. Thus, the set of buckets Π_{sel} is composed of $\Pi_{sel} = \{b1, b2, b3\}$. Note that these buckets are removed from the set Π_{toCh}.

4. In the step *4* the penalty associated with each of the buckets is computed. In the example, due to the resource saturation, the three buckets have an associated penalty of $penalty_1$,$penalty_2$ and $penalty_3$ greater than one (marked with a gray area in the figure 2). With these buckets the basic allocation is created because they share the required amount of time.

5. In step *5* the algorithm has to evaluate whether the current allocation can be improved by replacing any of the buckets of $\{b1, b2, b3\}$ by any of the buckets that have not been evaluated Π_{toCh}.

(a) As the current number of buckets in Π_{sel} is equal to the requested processors the algorithm continues the iteration.

(b) The bucket with the maximum penalty is selected *b1* (> 1).

(c) In the first iteration the bucket *b7* is evaluated. The penalty factor that processing the job α would experience if it were allocated using this bucket would be $penalty_7 = 1$.

(d) As the penalty $penalty_7$ is lower than the $penalty_1$, on one hand the bucket *b1* is removed form the Π_{sel} and inserted to Π_{desc}. On the other hand the bucket *b7* is removed from the Π_{toCh} and inserted in Π_{sel}. The reservation table at this point is shown in figure 3. Since not all the penalties of the selected buckets are 1 the algorithm goes ahead with the next bucket.

(e) As in the first iteration, since the current number of buckets in Π_{sel} is equal to the requested processors the algorithm continues the iteration.

(f) The bucket with the maximum penalty is selected *b2*.

(g) In the second iteration, bucket *b17* is evaluated. The penalty factor that a process of the job α would experience if it were allocated using this bucket would be $penalty_{17} = 1$.

(h) As the penalty $penalty_{17}$ is lower than the $penalty_2$, on one hand the bucket *b2* is removed form the Π_{sel} and inserted to Π_{desc}. On the other hand the bucket *b17* is removed from the Π_{toCh} and inserted in Π_{sel}. The reservation table at this point is shown in figure 4. Since all the penalties of the selected buckets is 1, the algorithm returns the allocation based on the currently selected buckets, because they provide the minimum penalty.

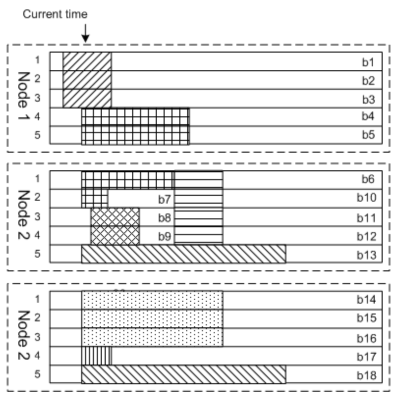


Figure 1: Less Consume Example

In the previous example, the start time for the allocation computed using the LessConsume resource allocation policy is the same as would have been obtained by using a First Fit resource selection policy. Thus, in the

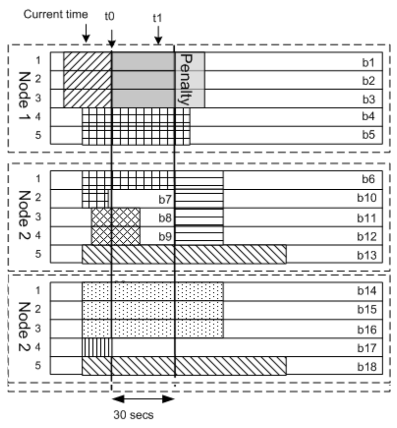


Figure 2: Less Consume Example - General Step 1

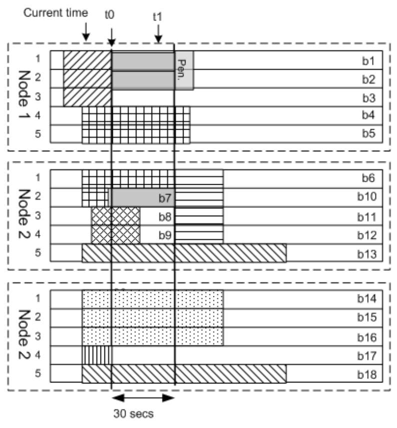


Figure 3: Less Consume Example - General Step 2

previous example, the start time remained equal to the First Fit RSP, and the penalty associated with the job has been reduced. However, in some situations, the Less-Consume policy may provide allocations with later start times than those using First Fit. For instance, if in this example the process allocated to the bucket b_{17} had experimented a penalty of *1.5* the LessConsume algorithm would have iterated again looking for an allocation with less penalty factor but later start time.

5 THE LESSCONSUME THRESHOLD RESOURCE SELECTION POLICY

As we have shown in the example, in some situations this policy not only minimizes the penalized factor of the allocated jobs, but it also provides the same start times as the first fit allocation policy, which in practice provides the earliest possible allocation start time. However, in many situations the allocation policy of the lower penalty factor provides a start time that is substantially later than that achieved by a FirstFit allocation. To avoid circumstances where the minimization of the penalty factor results in delays in the start time of scheduled jobs, we have designed the LessConsumeThreshold RSP. This is a

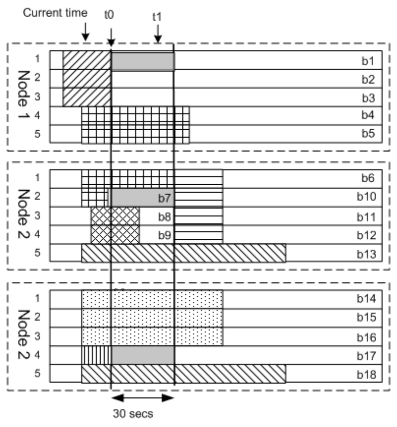


Figure 4: Less Consume Example - General Step 3

parametrized selection policy which determines the maximum allowed penalty factor allocated to any given job.

This resource selection policy has been mainly designed to be deployed in two different scenarios. In the first case, the administrator of the local scenario specifies in the configuration files the penalty factor of a given allocated job. This factor could be empirically determined by an analytical study of the performance of the system. In the second, more plausible, scenario, the local scheduling policy is aware of how this parametrized RSP behaves and how it can be used by different factors. In this second case the scheduling policy can take advantage of this parameter to decide whether a job should start in the current time or whether it could achieve performance benefits by delaying its start time. In the following subsection we describe a backfilling based resource selection policy that uses the LessConsumeThreshold RSP to decide which jobs should be backfilled and how to allocate the jobs.

The main differences between the two policies is that in the steps *4)* and *5.d.2)* if the number of buckets of the set Π_{sel} is the required processors for the job, and they have an associated penalty factor lower or equal to the specified *Threshold*, then the allocation will be defined and returned based on the current set. Note, that in the LessConsume policy the algorithm would iterate evaluating the rest of the free buckets. On the other hand, note that policy enforces that the job allocation penalty must be lower than the provided threshold. Thus if in the step *6)* of the algorithm an allocation is found but has a higher penalty the *treq* will be updated as in the step *4)* and the process would be iterated again. Note that the LessConsume would be stopped at this point due to it has a valid allocation with the lowest penalty that could achieve by optimizing the First First allocation.

EXPERIMENTS

In this paper we evaluate the effect of considering the memory bandwidth usage when simulating the Shortest Job Backfilled Firts policy under several workloads and both LessConsume resource selection policies (the **Less-**

Center	M.	FF	LC	1	1,15	1,25	1,5
CTC	H	8,8	8	7,6	7,8	7,9	8,1
	M	4,8	3,8	3	3,5	4,0	3,9
	L	0,9	0,7	0,5	0,7	0,6	0,8
SDSC	H	12	11	8,3	10	11	12
	M	6,7	6,1	4,7	4,8	5,6	5,9
	L	1,4	1,1	0,7	0,8	0,9	1,1

Table 1: Percentage of Penalized Runtime - 95_{th} Percentile

Center	M.	FF	LC	1	1,15	1,25	1,5
CTC	H	4,2	5,9	7,92	6,1	5,3	5,2
	M	2,8	3,5	4,22	3,8	3,6	3,5
	L	2,2	3,12	3,62	3,8	3,4	3,5
SDSC	H	99	110	128	115	109	106
	M	55	68	74	72	71	68
	L	37	45	57	52	42	42

Table 2: Bounded-Slowdown - 95_{th} Percentile

ConsumeThreshold with the thresholds *1*, *1,15*, *1,25* and *1,5*). Two different workloads from the Feitelson workload archive have been used. For each of them we have generated three different scenarios: with high (HIGH), medium (MED), and low (LOW) percentage of jobs with high memory demand.

5.1 Workloads

For the experiments we used the cleaned Tsafrir and Feitelson (2003) versions of the workloads SDSC Blue Horizon (SDSC-BLUE) and Cornell Theory Center (CTC) SP2. For the evaluation experiments explained in the following section, we used the first 10000 jobs of each workload. Based on these workload trace files, we generated three variations for each one with different memory bandwidth pressure:

- HIGH: 80% of jobs have high memory bandwidth demand, 10% with medium demand and 10% of low demand.
- MED: 50% of jobs have high memory bandwidth demand, 10% with medium demand and 40% of low demand.
- LOW: 10% of jobs have high memory bandwidth demand, 10% with medium demand and 80% of low demand.

5.2 Architecture

For each of the workloads used in the experiments we defined an architecture with nodes of four processors, 6000 MB/Second of memory bandwidth, 256 MB/Second of Network bandwidth and 16 GB of memory. In addition to the SWF Chapin et al. (1999) traces with the job definitions we extended the standard workload format to specify the resource requirements for each of the jobs. Currently, for each job we can specify the average memory bandwidth required (other attributes can be specified but are not considered in this work). Based on our experience and the architecture configuration described above, we defined that a *low memory bandwidth demand* consumes 500 MB/Second per process; a *medium* memory bandwidth demand consumes 1000 MB/Second per process; and that a *high memory bandwidth demand* consumes 2000 MB/Second per process.

6 EVALUATION

Table 2 present the 95_{th} percentile of the bounded slowdown for the CTC and SDSC centers for each of the three workloads for the FirstFit, LessConsume and LessConsumeThreshold resource selection policy. The last one was evaluated with four different factors: *1*, *1,15*, *1,25* and *1,5*. In both centers the LessConsume policy performed better than the LessConsumeThreshold with a factor of *1*. One could expected that the LessConsume should be equivalent to use the LessConsumeThreshold with a threshold of *1*. However, note that this affirmation would be incorrect. This is caused due to the LessConsume policy at the step *6)* of the presented algorithm will always stop. The goal of this policy is to optimize the First Fit allocation but without carry out a deeper search of other possibilities. However, the LessConsumeThreshold may look further in the future in the case that the penalty is higher than the provided threshold. Thereby, this last one is expected to provide higher wait time values. On the other hand, as we had expected, the bounded slowdown decreases while increasing the factor of the LessConsumeThreshold policy. In general, the ratio of increment of using a factor of *1* and a factor of *1,5* is around a 20% in all the centers and workloads.

The performance of these two resource policies, compared to the performance of the First Fit policy, shows that LessConsume policies give an small increment in the bounded slowdown. For instance, in the CTC high memory pressure workload the 95_{th} percentile of the bounded slowdown has increased from 4,2 in the First Fit to 5,94 in the LessConsume policy, or to 7,92 and 5,23 in the LessConsumeThreshold with thresholds of *1* and *1,5* respectively.

The 95^{th} percentage of penalized runtime is presented in the table 1. The penalized runtime clearly increases by incrementing the threshold. For instance, the 95^{th} Percentile of the percentage increases from 8,31 in the SDSC and the high memory pressure workload with a factor of *1* until 11,64 with a factor of *1,5*. The LessConsume, different from to the two previously described variables, shows similar values to the LessConsumeThreshold with a factor of *1,5*. This percentage of penalized runtime was reduced with respect to the First Fit when using all the

Center	M.	FF	LC	1	1,15	1,25	1,5
CTC	H	428	120	57	70	87	97
	M	247	101	76	77	102	99
	L	64	45	36	38	58	52
SDSC	H	475	105	87	130	127	130
	M	255	89	76	79	103	145
	L	51	34	22	27	33	41

Table 3: Number of Killed Jobs 95th Percentile

different factors in both centers.

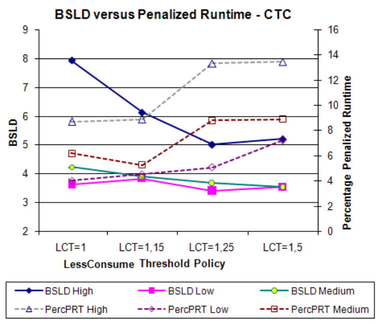


Figure 5: BSLD versus Percentage of Penalized Runtime - CTC Center

The number of killed jobs is the performance variable that showed most improvement in all the memory pressure workloads. The number of killed jobs is qualitatively reduced with the LessConsumeThreshold with a factor of *1*: for example with the high memory pressure workload and the CTC center, the number of killed jobs was reduced from 428 with the First Fit to 70. The other threshold factors also showed clear improvements; the number was halved. As to the LessConsume policy, the number of killed jobs was reduced by a factor of 4 compared to the First Fit and the high and medium memory pressure workloads of both centers.

The LessConsume policy shows how the percentage of penalized runtime and number of killed jobs can be reduced in comparison to the First Fit and First Continuous Fit, by using this policy with EASY backfilling. Furthermore, the LessConsume threshold shows how, with different thresholds, performance results can also be improved. Relaxing the penalty factor results in better performance of the system, and in an increase in the number of killed jobs and the percentage of penalized runtime. The LessConsume policy shows similar performance results as the LessConsumeThreshold with factors of *1,25* and *1,5*.

Figures 6 and 7 present the BSLD of the LessConsume policies against the percentage of penalized runtime of the jobs and the number of killed jobs. The goal of these figures is to show the chance that the LessConsumeThreshold and LessConsume policies have to improve the performance of the system while achieving an acceptable level of performance. As can be observed in figures 6 and 5 a good balance is achieved in the CTC center using the threshold of *1,15* where both the number of Killed Jobs and the percentage of penalized runtime converge are in acceptable values. In the case of the SDSC center, this point of convergence is not as evident as the CTC center. Considering the tendency of the bounded slowdown, it seems that the LessConsumeThreshold with a factor of *1* is an appropriate configuration for this center, due to the fact that the penalized runtime and the number of killed jobs presents the lowest values, and the bounded slowdown shows values that are very close to the factors of *1,15* and *1,25*.

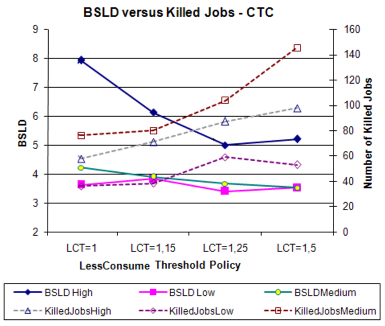


Figure 6: BSLD versus Killed Jobs - CTC Center

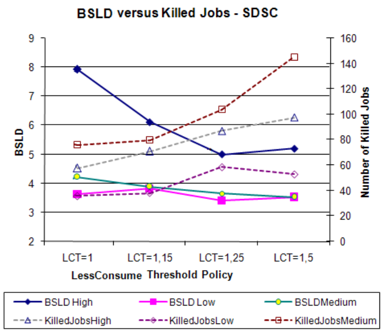


Figure 7: BSLD versus Killed Jobs - SDSC Center

7 Conclusions

In this paper we have shown how the performance of the system can be improved by considering resource sharing usage and job resource requirements by using the two LessConsume resource selection policies that consider the resource sharing when the jobs are allocated.

We have described the *Find Less Consume Distribution* that attempts to minimize the job runtime penalty that an allocated job will experience. Based on the utilization status of the shared resources in current scheduling outcome and job resource requirements, the LessConsume policy allocates each job process to the free allocations in which the job is expected to experience the lowest penalties. We have also described the *Find Less Consume Threshold Distribution* selection policy which finds an allocation for the job that satisfies the condition that the estimated job runtime penalty factor is lower than a given value *Threshold*. This resource selection policy has been designed to provide a more sophisticated interface between the local resource manager and the local scheduler in order to find the most appropriate allocation for a given job. Thus, this RSP can be used by the scheduler to find an allocation for a given job in an iterative process until the most appropriate allocation is found.

We have evaluated the impact of using the LessConsume and LessConsumeThreshold (Thresholds *1*, *1.15*, *1.25* and *1.5*) with the Shortest Job Backfilled first. In this evaluation, we have used the workloads described in the previous chapter where we evaluated the impact of memory bandwidth sharing on the performance of the system. Both resource allocation policies show how the performance of the system can be improved by considering where the jobs are finally allocated. The bounded slowdown of both policies show slightly higher values than those achieved by a First Fit resource allocation policy. However, they show a very important improvement in the percentage of penalized runtimes of jobs, and more importantly, in the number of killed jobs, showing a very good balance in the increment of the BSLD. Both have reduced by four or even six times the number of killed jobs in all the evaluated workloads. Note that each of the indicated thresholds depends on the center. In the SDSC the a threshold of *1* or *1.15* shows a good balance of performance (BSDL) and number of killed jobs and percentage of penalized runtime, and in the CTC center the appropriate threshold is *1.5*.

ACKNOWLEDGEMENTS

This paper has been supported by the Spanish Ministry of Science and Education under contract TIN200760625C0201, the under grant BES-2005-7919 and the IBM/BSC MareIncognito project.

REFERENCES

Chapin, S. J., Cirne, W., Feitelson, D. G., Jones, J. P., Leutenegger, S. T., Schwiegelshohn, U., Smith, W., and Talby, D. (1999). Benchmarks and standards for the evaluation of parallel job schedulers. *Job Scheduling Strategies for Parallel Processing*, vol 1659:pp. 66–89.

Chiang, S.-H., Arpaci-Dusseau, A. C., and Vernon, M. K. (2002). The impact of more accurate requested runtimes on production job scheduling performance. *8th International Workshop on Job Scheduling Strategies for Parallel Processing*, Vol. 2537:103 – 127.

Feitelson, D. G., Rudolph, L., and Schwiegelshohn, U. (2004). Parallel job scheduling - a status report. *Job Scheduling Strategies for Parallel Processing: 10th International Workshop, JSSPP 2004*, 3277 / 2005:9.

Guim, F., Corbalan, J., and Labarta, J. (2007). Modeling the impact of resource sharing in backfilling policies using the alvio simulator. *MASCOTS*.

Talby, D. and Feitelson, D. (1999). Supporting priorities and improving utilization of the ibm sp scheduler using slack-based backfilling. *Parallel Processing Symposium*, pages pp. 513–517.

Tsafrir, D., Etsion, Y., and Feitelson, D. G. (2005). Backfilling using runtime predictions rather than user estimates. *Technical Report 2005-5, School of Computer Science and Engineering, The Hebrew University of Jerusalem.*

Tsafrir, D. and Feitelson, D. G. (2003). Workload flurries. Technical report, School of Computer Science and Engineering and The Hebrew University of Jerusalem.

ENERGY EFFICIENT REAL TIME SCHEDULING OF DEPENDENT TASKS SHARING RESOURCES

Abdullah M. Elewi, Medhat H. A. Awadalla, and Mohamed I. Eladawy
Computer Engineering Department
Helwan University
11421,Cairo, Egypt
E-mail:{ abdullahelewi@gmail.com, awadalla_medhat@yahoo.co.uk }

KEYWORDS
Energy efficient scheduling, dynamic voltage scaling, real time, embedded systems, dependent tasks.

ABSTRACT

In the design of embedded systems, it is very important to reduce energy consumption and thus prolong battery life in everywhere existed battery-powered embedded systems. Dynamic voltage scaling (DVS) processors, which support many operating voltages and speeds, can efficiently reduce energy consumption by making appropriate decisions on the processor speed/voltage during the scheduling of real time tasks. This paper addresses the problem of energy efficient real time task scheduling where the tasks are dependent due to exclusive access shared resources. Furthermore, the paper proposes enhancements over the existing dual speed switching algorithm (DSA) where the proposed algorithm achieves more energy saving and has the capability to function with both SRP and DPCP protocols.

INTRODUCTION

Recently, embedded systems are seen in everywhere especially the portable ones such as cell phones, pocket PCs, multimedia devices, PDAs (personal digital assistants), wireless sensors, and medical implants, which all are battery powered. As the applications on these devices are being complicated, the energy consumption is also effectively increasing. So, minimizing energy consumption is a critical issue in the design of embedded systems, and techniques that reduce energy consumption have been studied at different levels in details (Chen and Kuo 2007).

Dynamic Voltage Scaling (DVS) is a new technology used to reduce power consumption in real time embedded systems, where the power consumption has two essential components: dynamic and static power. The dynamic power consumption, which is the main component (Gruian 2002), has a quadratic dependency on supply voltage and can be represented as:

$$P_d = C_{ef} \,.\, V^2_{dd} \,.\, F \qquad (1)$$

Where C_{ef} is the switched capacitance, V_{dd} is the supply voltage, and F is the processor clock frequency (sometimes referred as speed S) which can be expressed in terms of supply voltage V_{dd} and threshold voltage V_t as following:

$$F = k \,.\, (V_{dd} - V_t)^2 / V_{dd} \qquad (2)$$

The static power consumption is primarily occurred due to leakage currents (I_{leak}) and the static (leakage) power (P_{leak}) can be expressed as:

$$P_{leak} = I_{leak} \;.\; V_{dd} \qquad (3)$$

When the processor is idle, a major portion of the power consumption comes from the leakage. Currently the leakage power significantly increases with the new generations of processors, and much work has been done to address this problem (De and Borkar 1999, Butts and Sohi 2000).

So, lowering supply voltage is one of the most effective ways to reduce both dynamic and leakage power consumption. As a result, it reduces energy consumption where the energy consumption is the power dissipated over time:

$$Energy = \int Power\ dt \qquad (4)$$

However, DVS aims at reducing energy consumption by reducing the supply-voltage/speed of the processor provided that timing constraints are guaranteed. In other words, DVS makes use of the fact that there is no benefit of finishing a real time job earlier than its deadline.

A good review has been introduced in (Chen and Kuo 2007) about types and current trends of energy efficient real time scheduling techniques on dynamic voltage scaling (DVS) platforms.

DVS processors have two types: *ideal* and *non-ideal*. An ideal processor can operate at any speed in the range between its minimum available speed and maximum available speed. A non-ideal processor has only discrete speeds with negligible or non-negligible speed transition overheads.

Well-known examples of DVS processors are Intel StrongARM SA1100 processor which supports 12 voltage/speed levels, the Intel XScale which supports 4 voltage/speed levels, the Transmeta TM5400 processor which supports 6 voltage/speed levels, and the 1.6 GHz

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

Intel Pentium M processor which supports 6 voltage/speed levels.

RELATED WORK

During the last decade much work has been done in the field of energy efficient real time scheduling, but the authors of (Weiser et al. 1994) are considered the pioneers in this field, where they expected the amazing DVS technique, then Yao et al. (Yao et al. 1995) have proposed an optimal static (offline) scheduling algorithm by considering a set of aperiodic jobs on an ideal processor. Furthermore, many dynamic and static scheduling algorithms (Hu and Quan 2007, Pillai and Shin 2007, Shin and Kim 2004) have been proposed and applied on uniprocessor systems. Also multiprocessor and distributed systems have been considered (Andrei et al. 2007, Mishra et al. 2003). However, the problem of DVS with dependent tasks because of shared resources has been first addressed in (Zhang and Chanson 2002, Jejurikar and Gupta 2002a).
Jejurikar and Gupta (Jejurikar and Gupta 2002a) have proposed two algorithms for scheduling fixed priority (RM scheduler) tasks using priority ceiling protocol (PCP) described in (Sha et al. 1990) as resource access protocol. They have computed static slowdown factors which guarantee that all tasks will meet their deadlines taking into account the blocking time caused by the task synchronization to exclusive access shared resources. In their first algorithm, critical section maximum speed (CSMS), they have let the critical sections (sections deal with shared resources) to be executed at maximum processor speed and they have computed slowdown factors for executing non critical sections. The second algorithm, constant static slowdown (CSS), computes a uniform slowdown factor for all tasks and for all sections (critical and non-critical) saving speed switches occurred in the first algorithm (CSMS).
Then the same authors (Jejurikar and Gupta 2002b) have extended their previous algorithms (CSMS and CSS) to handle dynamic priority (EDF scheduler) tasks using dynamic priority ceiling protocol (DPCP) shown in (Chen and Lin 1990). The dynamic priority ceiling protocol is an extension of original priority ceiling protocol to deal with dynamic priority tasks (EDF scheduling).
Jejurikar and Gupta (Jejurikar and Gupta 2006) have also proposed a generic algorithm that works with both EDF and RM schedulers, and they have introduced the concept of frequency inheritance in their algorithm.
Zhang and Chanson (Zhang and Chanson 2002) have worked on the same problem, and proposed three algorithms for energy efficient scheduling of dependent tasks with shared resources, but they made use of stack resource policy (SRP) proposed by Baker (Baker 1991) as resource access protocol. The SRP can handle static and dynamic priority tasks (EDF and RM schedulers), reduces context switches over PCPs, and is easy implemented.

The first algorithm is the same as CSS for EDF scheduler proposed in (Jejurikar and Gupta 2002b) because they all have derived the static slowdown factor directly from the EDF schedulability test with blocking time:

$$\overset{\forall i}{i = 1,\ldots,n} \quad \frac{B_i}{D_i} + \sum_{k=1}^{i} \frac{C_k}{D_k} \leq 1 \qquad (5)$$

Where C is the computation time (worst case execution time WCET), D is the task relative deadline, and B is the blocking time that can be defined as the maximum time within which a high priority task can be blocked by a low priority task due to mutual exclusion shared resource.
The second algorithm, which is the interest of this paper, is the dual speed switching algorithm. The main concept of this algorithm is using two speeds (H, L) and switching between them. Initially the algorithm operates at the low speed L and switches to the high speed H as soon as a blocking occurs.
The last algorithm is the dual speed dynamic (online) reclaiming algorithm which dynamically collects the residue time from early completed jobs and redistributes it to the other pending jobs for further reducing of the processor speed and achieving more energy saving.

SYSTEM MODEL

Task Model

In this paper, real-time periodic tasks are considered. A periodic task is a sequence of jobs released at constant intervals (called the period). Each task τ is characterized by the following parameters (Cottet et al. 2002):

- The release time (r): the time when the task first released.
- The period (T): the constant interval between jobs.
- The relative deadline (D): the maximum acceptable delay for task processing.
- The computation time (C): the worst case execution time (WCET) of any job.
- The blocking time (B): the maximum time a task can be blocked by another lower priority task.

In this paper we consider well formed tasks that satisfy the condition: $0 \leq C \leq D \leq T$.
A 3-tuple τ ={C, D, T} represents each task. The relative deadline is assumed to be the same as the period in all illustrative examples.

Processor Model

The tasks are scheduled on a single DVS processor that can operate at any speed/voltage in its range (ideal). Of course, practical DVS processors supports discrete speed/voltage levels (non ideal). So, the desired speed/voltage of the ideal DVS processor is rounded to the nearest higher speed/voltage level the practical DVS processor supports.

The time required to change the processor voltage/speed is very small compared to that required to complete a task. It is assumed that the voltage change overhead, similar to the context switch overhead, is incorporated in the task computation time.
In this paper, it is assumed that the processor's maximum speed is 1 and all other speeds are normalized with respect to the maximum speed.

DUAL SPEED SWITCHING ALGORITHM

Dual speed algorithm (DSA) proposed in (Zhang and Chanson 2002) is a blocking aware scheduling algorithm with non-preemptible critical sections using SRP as resource access protocol.
The authors have noted that the static speed in the constant static slowdown (CSS) algorithm is higher than the required, and CSS consumes a lot of energy, but it is extremely effective when a task is blocked to avoid deadline miss. So, if it is possible to switch between two speeds: high H and low L (where L ≤ H ≤ 1), the algorithm will be more energy efficient, and this is the concept behind DSA.
The high speed is the same as the static speed in CSS, and it is the speed that satisfies all tasks to meet their deadlines when a blocking occurs. The high speed is derived directly from the EDF schedulability test with shared resources and SRP protocol:

$$\overset{\forall i}{i=1,\ldots,n} \quad \frac{B_i}{D_i} + \sum_{k=1}^{i} \frac{C_k}{D_k} \le \mathsf{H} \qquad (6)$$

The low speed is the optimal lowest speed with which all tasks can be scheduled without missing any deadline (no blocking occurs), and it is derived from the plain EDF schedulability test without shared resources:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \le \mathsf{L} \qquad (7)$$

Initially, DSA starts with the low speed L, then it switches to the high speed H as soon as a task is blocked.

An Illustrative Example

An example presented in (Jejurikar and Gupta 2006) is implemented to illustrate the dual speed algorithm and the contribution of this paper. A hard real time system with two tasks is considered as following:

$$\tau_1 = \{2, 5, 5\}, \tau_2 = \{4, 40, 40\}$$

The arrival times and critical sections of the two tasks within the least common multiple (LCM) of periods are shown in Figure 1(a). τ_1 may be blocked by the critical section of the lower priority τ_2, so the blocking time of τ_1 is the length of τ_2's critical section, i.e. $B_1=3$, and in the same manner $B_2=0$ because there are no lower priority tasks to block τ_2.
According to (6) the static (high) speed is H= max (2/5+3/5, 2/5+4/40) =1 which is used to schedule the tasks in CSS algorithm as shown in Figure 1(b), then using (7) let us calculate the low speed L= 2/5+4/40=0.5 to use it with the high (static) speed calculated earlier in the scheduling of the two tasks (τ_1,τ_2) using DSA as shown in Figure 1(c).
The rectangles represent the processing of tasks, where the vertical dimension represents the processor speed, and the horizontal dimension represents the execution time elapsed for processing tasks according to their WCETs and the processor speed. It is clearly noted that the area of the rectangles that represent the jobs of the same task is the same because these jobs always take the same number of execution cycles which equals to processor speed multiplied by elapsed time.

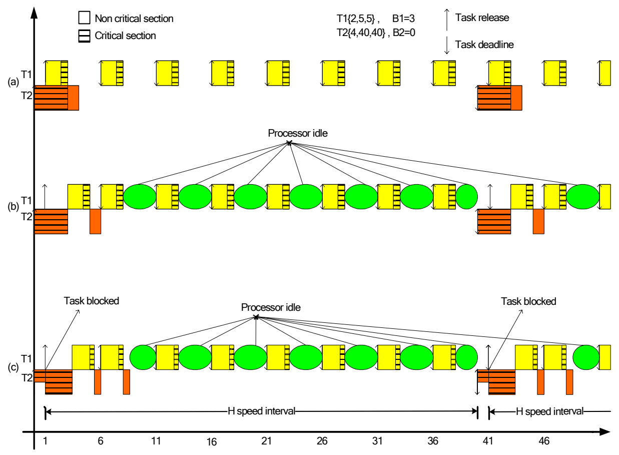


Figure 1: (a) Task Set Description: Arrival Times, Computation Times, and Critical Sections. (b) CSS. (c) DSA.

It is clear in this example that the CSS does not save any energy because it has used the maximum speed. Due to such situations DSA has been developed to achieve more energy savings.

Enhanced Dual Speed Algorithm

It is obvious in Figure 1 that DSA has used the low speed just for one second, and then it has switched to the high speed and continued until the blocking task (the low priority task) deadline which is mostly further than the blocked task (the high priority task) deadline. This highlights the key idea of the first contribution of the enhanced DSA (EDSA) proposed in this paper to end the high speed interval by the blocked task deadline which is mostly earlier.
To test the performance of EDSA, the previous example has been performed using DSA and EDSA. As shown in Figure2, the processor idle time is reduced from 49% in DSA to 12.5% in EDSA which reflects the improvements achieved in the system performance.

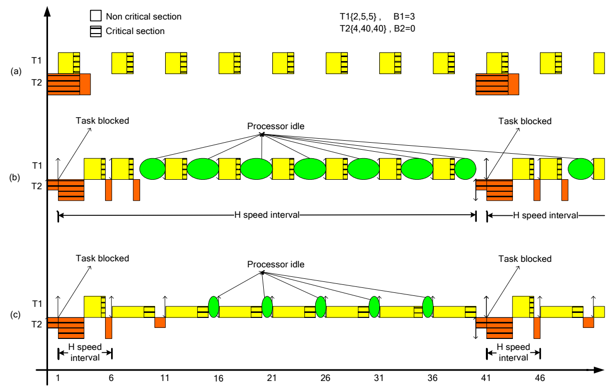


Figure 2: (a) Task Set Description. (b) DSA. (c) EDSA.

To validate the effect of EDSA, Another hard real time system with three tasks is addressed:

$\tau_1=\{1,4,4\}$, $\tau_2=\{2,8,8\}$, $\tau_3=\{3,10,10\}$.

The arrival times and critical sections of the three tasks within the least common multiple (LCM) of periods are shown in Figure 3(a). τ_1 may be blocked by the critical section of the lower priority τ_2 or τ_3, so the blocking time of τ_1 is the length of longest critical section in the worst case, i.e. $B_1=\max(1.5, 2.5)=2.5$, and in the same manner $B_2=2.5$ because τ_2 can be blocked by τ_3, while $B_3=0$ because there are no lower priority tasks to block τ_3.
According to (6, 7), the high speed H=max (1/4+2.5/4, 1/4+2/8+2.5/8, 1/4+2/8+3/10) =0.875, and the low speed L =1/4+2/8+3/10=0.8.

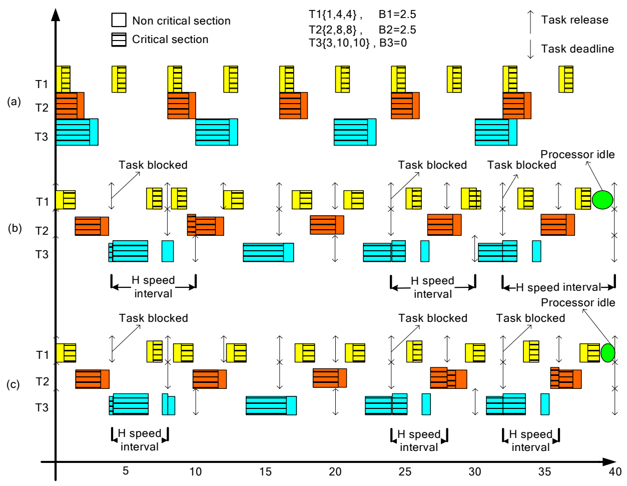


Figure 3: (a) Task Set Description. (b) DSA. (c) EDSA.

Even though the high speed and low speed are close to each other in this example, there is again an improvement in the system performance based on EDSA compared with DSA where the processor idle time is reduced from 4.28% to 2.8%.

DSA and Priority Ceiling Protocols

Jejurikar and Gupta (Jejurikar and Gupta 2006) have shown that the success of DSA requires that the critical sections be non-preemptible, and DSA may cause deadline miss when it is used with dynamic priority ceiling protocol(DPCP) because a high priority task can preempt a critical section of another low priority task if the high priority task does not need to use the shared resource. However, the blocking occurs at the moment when the high priority task needs to access the resource, and the switching to high speed H is delayed until this moment causing deadline miss as shown in Figure 4.

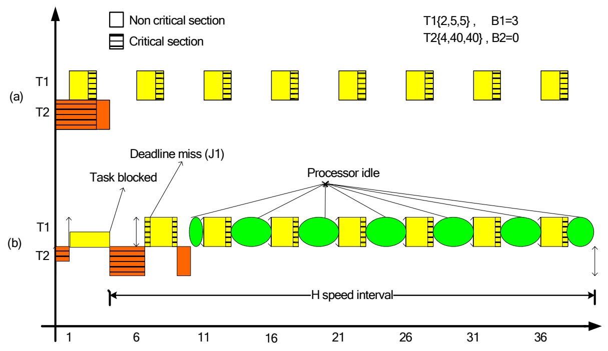


Figure 4: (a) Task Set Description. (b) Deadline Miss When DSA Is Used With DPCP.

At the moment t=1, task τ_1 is released, and task τ_2 that executes its critical section is preempted by the high priority task τ_1. Task τ_1 is just blocked when it needs to access the shared resource, i.e. at the moment t=4, where the high speed interval starts, and τ_2 resume executing until it ends its critical section to be preempted again by τ_1. Then, task τ_1 resume executing, but it can not meet the deadline, where the first job (j_1) misses its deadline because of delaying the start of the high speed interval(blocking instant) due to using DPCP as resource access protocol as shown in Figure 4(b).
It is clear that to avoid the occurrence of deadline miss, the switching to the high speed must occur earlier as in using SRP as resource access protocol, and this is the second contribution of EDSA. To achieve that, EDSA proposes that the switching to the high speed should occur when a critical section is preempted or a blocking takes place.
Figure5 shows the adaptation of EDSA with DPCP as resource access protocol to avoid deadline misses and achieve energy saving as done with SRP, where the switching to the high speed occurs not only when the high priority task is blocked, but also when a critical section is preempted. Unlike SRP, DPCP increases the context switches as it is obvious in Figures 2(c) and 5, and this is the key advantage of SRP over PCPs as mentioned earlier.

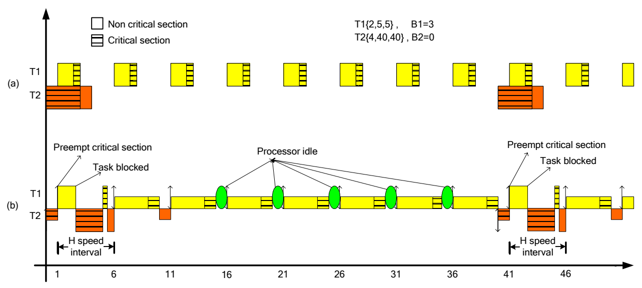


Figure 5: (a) Task Set Description. (b) Deadline Meeting When EDSA Is Used With DPCP.

Figure6 shows the proposed algorithm EDSA which achieves more energy saving and works with SRP and DPCP as resource access protocols.

```
/* Initially the processor speed is L. End_H indicates the end of the high speed interval. If the system isn't in a high speed interval, End_H = -1. Initially End_H = -1 */

When job J $_{i,j}$ arrives:
  if Priority($J_{i,j}$ ) > Priority( current job)
      if Preempt_Current_Job( ) is successful
          if Preempt_Critical_Section( ) is successful
            Set_Speed(H); /* Set the processor speed at H*/
            End_H = max( End_H, $d_{i,j}$ );
              /* $d$ is the deadline of the job */
          end if
          Execute $J_{i,j}$ ;
      else /* $J_{i,j}$ is blocked */
          Set_Speed(H); /* Set the processor speed at H*/
          End_H = max( End_H, $d_{i,j}$ );
      end if
  end if

when the end of high speed interval is reached:
    End_H = -1;
    Set_Speed( L); /* Set the processor speed at L */
```

Figure 6: The Enhanced Dual Speed Algorithm (EDSA)

RESULTS AND DISCUSSION

Referring to Figures 2 and 3, reducing the time during which the processor is idle comes from lowering the processor speed for longer time intervals. This, in turn, reduces the energy consumption dramatically due to quadratic dependency between power and processor speed. To verify that, a comparison study has been performed by computing the energy consumed in CSS, DSA, and EDSA using the simplified power model $P=S^2$ used in (Jejurikar and Gupta 2002), where the blocking time B changes from 0 to the highest amount at which the task set is schedulable (when H=1).

Of course, the high speed H changes from H=L (when B=0) to H=1, while the low speed L does not change.

As it is clear from Figure7, EDSA is the most energy efficient algorithm especially with high blocking times, where the difference between the low and high speeds (L, H) increases significantly.

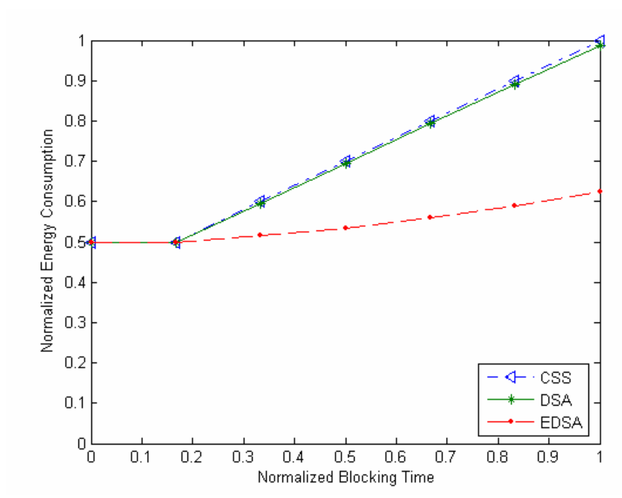


Figure 7: Energy Consumption Versus Blocking Time Changes In Example1

The comparison is repeated for the second example, it is noticed that, as shown in Figure 8, EDSA exhibits a slight improvement over DSA with the highest blocking time due to the small difference between high and low speeds (H, L).

As a result, when the blocking time is low (the high speed is almost the same as the low speed), the three algorithms exhibit the same performance. When the blocking time increases (the difference between the high and low speeds also increases), EDSA behaves better than the other two algorithms (CSS and DSA) especially when this difference is significant.

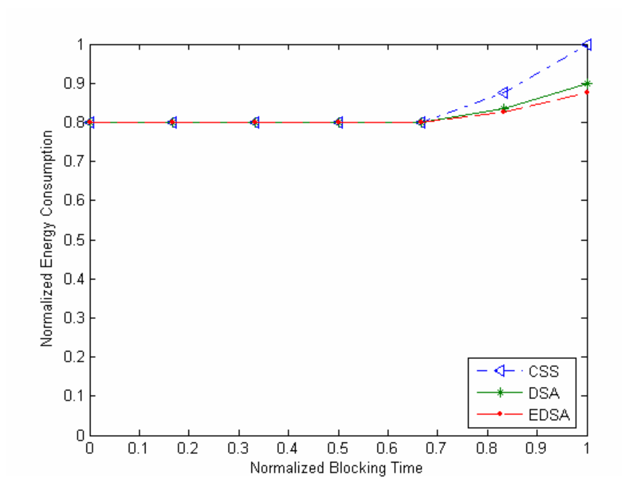


Figure 8: Energy Consumption Versus Blocking Time Changes In Example2

Furthermore, EDSA has the advantage over DSA that it can work with SRP and DPCP as resource access protocols as shown in Figures 2 and 5.

CONCLUSIONS

The paper has addressed the problem of real time scheduling of dependent tasks due to exclusive access shared resources taking into account the reducing of energy consumption as a main goal. The paper has proposed improvements over the existing dual speed switching algorithm (DSA), where the proposed algorithm, EDSA, has shown more energy saving than DSA. furthermore, it has adaptation to work not only with SRP but also with DPCP as resource access protocols.

REFERENCES

Andrei, A., P. Eles, Z. Peng, M. Schmitz, and B. M. Al-Hashimi. 2007. "Voltage Selection for Time-Constrained Multiprocessor Systems" In Proc. Of Designing Embedded Processors – a Low Power Perspective, 259–284.

Baker, T. P. 1991 "Stack-based scheduling of real time processes," Journal of Real-Time Systems, Vol. 3, No. 1, 67– 99.

Butts, J.A. and G.S. Sohi. 2000. "A static power model for architects" In Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture, Monterey, CA, USA, 191-201.

Chen, M. and K. Lin. 1990. "Dynamic priority ceilings: A concurrency control protocol for real-time systems" Real Time Systems Journal, Vol. 2, No. 1, 325–346.

Chen, J. and C. Kuo. 2007. "Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms". 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).

Cottet, F., J. Delacroix, C. Kaiser,and Z. Mammeri. 2002. Scheduling in Real-Time Systems, John Wiley & Sons Ltd, England.

De, V. and S. Borkar. 1999. "Technology and Design Challenges for Low Power and High Performance". In Proceedings of the International Symposium on Low Power Electronics and Design, San Diego, CA, USA, 163-168.

Gruian, F. 2002. Energy-Centric Scheduling for real time systems, PhD thesis , Lund Institute of Technology, Lund University,

Hu, X. and J. Quan. 2007. "Fundamentals of Power-Aware Scheduling". J. Henkel and S. Parameswaran (eds.), Designing Embedded Processors – A Low Power Perspective, 219–229..

Hu, X., J. Quan. 2007. "Static DVFS Scheduling." J. Henkel and S. Parameswaran (eds.), Designing Embedded Proc.of A Low Power Perspective, 231–242.

Jejurikar, R. and R. Gupta. 2002. "Energy aware task scheduling with task synchronization for embedded real time systems". In Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), 164–169.

Jejurikar, R. and R. Gupta. 2002. "Energy aware edf scheduling with task synchronization for embedded real time operating systems," Technical report #02-24. Department of Information and Computer Science, University of California at Irvine, (Aug).

Jejurikar, R. and R. Gupta. 2006 "Energy aware task scheduling with task synchronization for embedded real time systems". IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 25, No. 6, 1024 – 1037

Mishra, R., N. Rastogi, D. Zhu, D. Moss´e, and R. Melhem. 2003. "Energy aware scheduling for distributed real-time systems". In International Parallel and Distributed Processing Symposium, 21-30.

Pillai, P. and K. G. Shin. 2007. "Dynamic DVFS Scheduling." J. Henkel and S. Parameswaran (eds.), Designing Embedded Processors – A Low Power Perspective, 243–258.

Sha, L., R. Rajkumar, and J. P. Lehoczky. 1990. "Priority inheritance protocols: An approach to real-time synchronization," IEEE Transactions on Computers, Vol. 39, No. 9, 1175–1185.

Shin, D. and J. Kim. 2004. "Dynamic voltage scaling of periodic and aperiodic tasks in priority-driven systems". ASPDAC, 635– 658.

Weiser, M., B. Welch, A. Demers and S. Shenker. 1994. "Scheduling for reduced cpu energy". In Proc. of Symposium on Operating system Design and Implementation (OSDI), 13–23.

Yao, F., A. Demers and S. Shenker. 1995. "A scheduling model for reduced cpu energy". In Proceedings of IEEE Annual Symposium on Foundations of Computer Science, 374–382.

Zhang, F. and S. T. Chanson. 2002. "Processor voltage scheduling for real-time tasks with non-preemptible sections". In Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), 235–245.

AUTHOR BIOGRAPHIES

ABDULLAH M. ELEWI graduated from the Electrical and Electronic Engineering Faculty of Aleppo University, Syria, where he studied computer engineering and obtained his Bachelor degree in 2005 and graduate studies Diploma in 2006, then he moved to Egypt to complete his graduate studies. He is currently a master student at the computer engineering department at Helwan University. His research interests include energy efficient real time scheduling techniques, embedded systems and real time operating systems. His e-mail address is: `abdullahelewi@gmail.com`

MEDHAT H. A. AWADALLA has obtained his B.Sc. from Helwan University, 1991 in electronics and communications department and got his M.Sc in the field of reconfigurable computer architecture in 1996 from Helwan university and His PhD from Cardiff university, UK in the filed of mobile robots in 2005. He has a post Doc. at Cardiff university in 2006 and currently he is working as a lecturer in local and private universities in Egypt as Helwan university, Misr international university, and Misr university for science and technology. His research interests include real time systems, multi-core processors, computing grid, mobile robots, and sensor networks. His e-mail address is : awadalla_medhat@yahoo.co.uk

MOHAMED I. ELADAWY graduated from the Department of Electrical Engineering, Faculty of Engineering of Assiut University in May 1974; M.Sc. from Cairo University in May 1979; Ph.D. from Connecticut State University, School of Engineering, in May 1984. He worked as an Instructor at the Faculty of Engineering, Helwan University since 1974. Currently he is a Professor at the Department of Communication and Electronics Engineering and the Vice Dean for Student Affairs in the same faculty. He was working for the general organization for technical and vocational training for 6 years from 1989 to 1995 in Saudi Arabia. His main research interests include signal processing and its medical applications, real time systems, and image processing.

Partitioning and Scheduling Schemes for Grid and Cluster Systems

SCHEDULING STRATEGIES IN FEDERATED GRIDS

Katia Leal
Universidad Rey Juan Carlos
Dep. de Sistemas Telemáticos y Computación
Escuela Sup. de Ciencias Experimentales y Tec.
Tulipán SN, Mósteles, Madrid, Spain
Email: katia.leal@urjc.es

Eduardo Huedo, Rubén S. Montero, Ignacio M. Llorente
Univ. Complutense de Madrid
Dep. Arquitec. de Comp. y Automática
Facultad de Informática 28040, Spain
Email: contact@dsa-research.org

KEYWORDS

Federated Grids, Scheduling

ABSTRACT

The GridWay Metascheduler enables efficient sharing of computing resources managed by different LRM (Local Resource Management) systems, not only within a single organization, but also across several administrative domains. The possibility to access to the services available in different Globus based grids allows the union of grids to create a federation. This scenario has particular characteristics, possibly the most important one is that it has different types of users: internal users, external users, and direct users. Basically, all these users compete for the resources of the federated grid to achieve their own particular goals. However, GridWay is not providing the best scheduling strategy under this scenario, because its current scheduling policy does not take into account resource ownership. In this paper we introduce a variation of GridWay's current scheduling strategy suitable to this new scenario. This variation is based on the parameters provided by a previously proposed performance model. In addition, the results obtained by simulation lead us to conclude that there is a real necessity to enhance scheduling policies in federated grids.

INTRODUCTION

A Federated Grid can be formed of several grid infrastructures. However, the participants in the Federated Grid do not collaborate to achieve the same goal, like the participants of a Global Grid. Here the idea is that each participant shares resources with the rest, but always having in mind that the main user of those resources is the participant itself. GridWay (Huedo et al. (2004)) provides the technology to build Federated Grids, both directly and through *GridGateWays*. A GridGateWay is a WS-GRAM (Web Services Grid Resource Allocation and Management) service hosting a GridWay workload manager that enables remote access to GridWay's metascheduling capabilities through a WSRF (Web Services Resource Framework) interface. However, GridWay applies scheduling strategies that are better fitted to Partner and Enterprise Grids. There is a huge ongoing research effort on grid scheduling (Dong and Akl (2006); Andrieux et al. (2003)), but it is mainly centered on Partner Grids. With this paper we want to drive attention to the particular characteristics of Federated Grids, and in the necessity of new scheduling policies to support them. Thus, we propose an alternative to GridWay's current scheduling policy based on a performance model (Montero et al. (2006)) that allows to parametrize and compare different Grids.

The rest of the paper is structured as follows: we first present and compare other scheduling approaches with our solution. Then, we explain the mapping strategy used by our scheduling proposal to maximize the throughput. Next, we present the design and implementation of the scheduling model. We also show some experimental results. Finally, we explain the conclusions and future work.

RELATED WORK

It is well known that the general scheduling problem is NP-complete (Ullman (1975)). A large number of algorithms have been applied to schedule jobs in computational grids. However, none of them seems suitable to federated grids. Here we enumerate some scheduling algorithms and the drawbacks they present under this scenario.

The Opportunistic Load Balancing (OLB), the Minimum Completion Time (MCT), the Min-min, and the Max-min are similar algorithms and also have very similar drawbacks under a federated grid. The first is a problem of scalability: the time to calculate the selected node increases with the number of nodes. The second problem is that in a federated grid nodes are different, so we should not simply assign a job to the next available node.

The main disadvantage of the Weighted meta-scheduling (Song et al. (2005)), and of the QoS guided Min-Min (He et al. (2003)) algorithms is that the former is specific of data intensive applications, and the latter focuses in long-term applications.

More close to our problem is the work of (Wiriyaprasit and Muangsin (2004)) that analyzes the impact of local policies on the performance of grid scheduling on a computational grid. However, their simulated scenario has

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

certain drawbacks: it is not based in a real testbed, it can only be applied to computational grids, and it doesn't reflect the effects of including the algorithm in a real scheduler, like GridWay. In contrast, we show experimental results obtained from a simulated federated grid based on a real testbed, and using GridWay as the scheduler.

A NEW SCHEDULING MODEL TO MAXIMIZE THE THROUGHPUT

In this section we first analyze the environment conditions of Federated Grids, and then we introduce our proposal of a new scheduling model.

Federated Grid

In a Federated Grid the different participants collaborate by sharing their resources with the whole Grid. However, they do not try to achieve the same goals as they have to satisfy their own users demands. In doing so, each participant can use his own internal resources, but also the grid resources that the rest of participant are willing to share. Each participant decides which resources to contribute with, who can use those resources, and the access policy to them. Of course, all these restrictions can change dynamically, and brokers should be prepared for that. A possible Federated Grid schema could be like the one shown in the Figure 1. As it can be seen, there are different types of resources and users. We have identified two types of resources:

❶ *Internal Resources*: these are the resources directly accessible by the broker through the corresponding local workload manager. That is, the resources owned by the particular research center, laboratory or company.

❷ *External Resources*: we can classify Enterprise Grids, Partner Grids, and Utility Grids provided by third part companies in this category.

Also, we have identified internal, external, and direct users. They differ in the way, and in the rights they have to access resources:

❶ *Internal Users*: the jobs submitted by these users through GridWay can be executed in both the internal and the external resources. Depending on different parameters, such as the local load, GridWay will decide to which resource submit the job.

❷ *External Users*: all the jobs received by GridWay through the GRAM interface will be from external users. GridWay will apply different policies to decide whether to accept or not the jobs received.

❸ *Direct Users*: GridWay cannot control the jobs submitted by this type of internal users. However, they are important since they have an influence in the load of the resources.

As a result, each type of user introduces its own requirements that will affect GridWay scheduling policies.

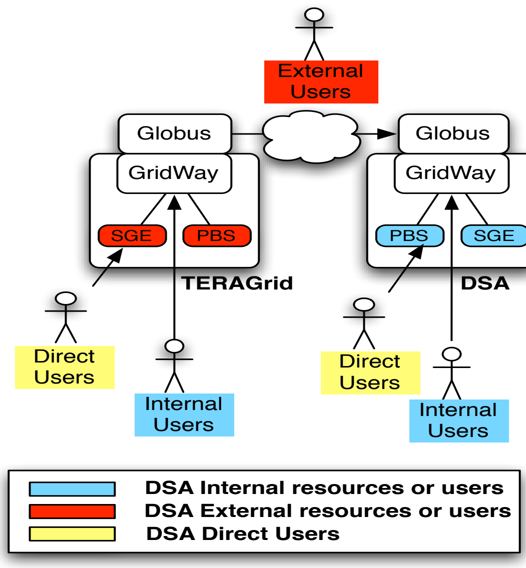


Figure 1: Example of a Federated Grid.

The mapping strategy

GridWay receives jobs directly through the command line interface from internal users, and through the GRAM interface from external users. In this way, GridWay can differentiate the jobs submitted by internal users from those submitted from external users. However, GridWay currently operates in the same way, and applies the same policies for both internal and external jobs.

Next, we explain the modifications already included in our simulated GridWay to support the restrictions introduced by the different type of users on a federated grid. Our simulated GridWay will work in different ways depending on the type of job received.

In scheduling an internal job

Our simulated GridWay almost includes all the configurable restrictions of a real GridWay: i.e. the maximum number of jobs that will be dispatched at each scheduling action and the period (in seconds) between two scheduling actions. In scheduling an internal job, the normal version of the simulated GridWay firstly checks if there are free nodes available in internal resources. If there are free internal nodes, GridWay schedules the job to an internal resource. In contrast, if there are no free internal nodes but free external ones, GridWay schedules the job to an external resource. However, we want to improve the normal scheduling policy to maximize the number of jobs that can be executed while maintaining makespan value. Thus, the scheduling policy should take into account not only which is the next available node. To maximize the throughput we need to obtain the number of jobs that should be submitted to internal resources and to external resources. We have used the equation that represents the best characterization of the Grid to obtain these numbers. The characterization can be obtained if we take the line that represents the average behavior of the system, as proposed by Hockney, and Jesshope (Hockney and Jesshope (1988)):

$$n(t) = r_\infty t - n_{1/2} \qquad (1)$$

In the Equation 1 *n* represents the number of completed tasks as a function of time *t*. The other parameters are:

- ❐ **Asymptotic performance** r_∞: is the maximum rate of performance in tasks executed per second. In the case of an homogeneous array of *N* processors with an execution time per task *T*, we have r_∞= *N/T*.
- ❐ **Half-performance length** $n_{1/2}$: is the number of tasks required to obtain the half of the asymptotic performance. This parameter is also a measure of the amount of parallelism in the system as seen by the application.

The linear relation represented by Equation 1 can be used to define the performance of the system (tasks completed per second). We explain later how we can also use this linear equation to obtain the number of jobs that should be submitted to internal resources and to external resources to maximize the throughput.

In scheduling an external job

As soon as GridWay receives an external job, it has to decide whether to accept it or not. By default, the simulated GridWay accepts all the external jobs received from external users.

When GridWay has to schedule an external job, it employs a different strategy than when scheduling an internal job. In the case of an external job, GridWay applies it's current scheduling policies. That is, it schedules the external job to the next internal resource with free nodes. In this way we avoid the situations on which a participant of the Federated Grid can receive from another one a job previously submitted to it.

DESIGN AND IMPLEMENTATION

We have previously (Vázquez et al. (2008, 2007)) set up a simple, but real infrastructure, where a client runs an instance of the GridWay Metascheduler interfacing internal resources in an enterprise grid, the DSA (Distributed System Architecture) enterprise grid at Complutense of Madrid University, based on Globus Toolkit 4 (GT4) WS interfaces, and a GridGateWay that gives access to resources from a partner grid (*fusion* VO of the EGEE), based on GT pre-WS interfaces found on gLite 3.0. However, prior to run our enhanced scheduling algorithm on a real production infrastructure, we have first implemented the modified algorithm on a simulated environment. The deployment on a real environment will require involvement of a large number of active users and resources, which is very hard to coordinate and build. Thus, the simulation appears to be the easiest way to analyze the modified scheduling policy. Based on the simulation results, we can later encourage or discourage the deployment on a real production environment.

We have used the well known GridSim toolkit (`http://www.gridbus.org/gridsim/`) to simulate our test scenario.

Test Scenario

Since the idea is to finally deploy the new GridWay on a real infrastructure, the simulation results have to be as realistic as possible. Thus, the simulated scenario has to be close enough to reality. We will start with the simple, but more or less realistic scenario depicted in Figure 2.

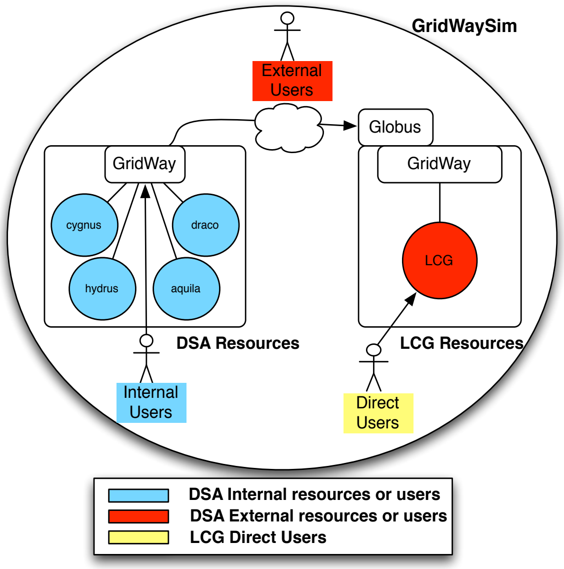


Figure 2: A simple test scenario.

As it can be seen, in this test scenario there are only two grid resources: the DSA (Distributed System Architecture) and the LCG (LHC Computing Grid). The DSA testbed represents the resources of the Distributed System Architecture research group at the Complutense of Madrid University. In the same way, the LCG testbed represents the Large Hadron Collider (LHC) Computing Grid. From the point of view of a *DSA internal user*, the DSA GridWay is her broker, the DSA resources are internal resources, and the LCG resources are external resources. In the same way, all the jobs received by the LCG GridWay through the Globus GRAM interface are from external users. While the GridWay on the DSA site has to apply a policy to submit jobs to internal and/or external resources, the LCG GridWay has to decide whether or not to accept the jobs from external users.

Table 1 shows the number of computing elements, aka PEs (Processing Elements), and MIPS (Millions Instructions Per Second) of each machine in the DSA infrastructure. The Table 2 shows the same values for the machines in the LCG resource. We have calculated the MIPS value based on machine's model, and number of MHz. We use all these characteristics to simulate resources in order to obtain as realistic as possible results.

Machine	PEs	MIPS/PE
hydrus	4	9787
aquila	5	9787
orion	1	9787
cygnus	2	6536
draco	1	6536

Table 1: Characteristics of the machines in the DSA research testbed.

Machine	PEs	MIPS/PE
machine0	800	9787
machine1	640	6536
machine2	560	4902

Table 2: Characteristics of the machines in the LCG research testbed.

GRIDWAYSIM ENTITIES

We have called *GridWaySim* to our simulation of the scenario shown in Figure 2. We explain the different participating entities of GridWaySim in the next sections.

GridWaySim

This entity represents the whole simulation, and is responsible of the creation of the main simulated entities: GridWay brokers, users, DSA and LCG resources, and workload (or LCG direct users).

GridWay

The *GridWay* entity represents a generic GridWay meta-scheduler. Since we need to interconnect the DSA, and LCG grids to form a federation, we have to instantiate two GridWay brokers: one for DSA, and the other for the LCG. Thus, from the point of view of the DSA GridWay, DSA resources are internal resources, and LCG is an external resource. On the other hand, for the LCG GridWay, DSA is an external resource, and LCG is an internal resource. For this first test scenario, the flow of jobs is only from DSA GridWay to LCG GridWay. However, communication can be done in both directions. Also, the DSA GridWay only receives experiments (a collection of jobs) from her internal users, and the LCG GridWay only receives jobs from DSA GridWay. Finally, to simulate a real environment, we have also introduced direct users in the LCG resource by means of the *Workload* entity.

Testbed: DSATestbed, LCGTestbed

A *Testbed* represents a generic set of grid resources. The resources of the DSA research group are represented y the *DSATestbed* entity, and the LCG ones by the *LCGTestbed* entity. Each follows the configurations depicted in Tables 1 and 2, respectively. The main difference between these two entities is that the LCG resources are represented by an unique resource, while DSA ones are represented as they really are, that is, by five resources. We have instantiated all the resources to use the spaced-shared policy as the internal management. As defined in the GridSim API, this policy uses the First Come First Serve (FCFS) algorithm.

User

The *User* models an user that submits experiments to a GridWay broker. We use this entity to represent internal as well as external users. The functionality of each user includes the submission of experiments to the correspondent broker, and waiting for it completion.

Experiment

An *Experiment* is a collection of jobs. We use this entity to recover important information about the experiment (such as the start and end times), and of all its jobs.

Job

The *Job* entity represents a generic job submitted to the grid. This entity provides specific information about each job: start time, end time, and CPU time among others. We can represent jobs of different computation times, and with different input and output file sizes.

Workload: The LCG Grid log

Since the main purpose of our simulation is to create a realistic environment, we have used the *Workload* entity in our tests. The Workload entity submits jobs by reading resource traces from a file. Thus, our jobs are competing with the jobs submitted by the Workload entity. For this reason, the LCG grid resources might not be available at certain times. The file follows the standard workload format as specified in `http://www.cs.huji.ac.il/labs/parallel/workload/`. As trace file, we have used the LCG Grid Log that contains 11 days of real activity from multiple nodes that make up the LCG (Large Hadron Collider Computing Grid, `http://lcg.web.cern.ch/LCG/`). Next, we enumerate some details about this testbed:

- ❒ **Number of jobs submitted**: 188,041. The log specifies the submit time and the run time of each job.
- ❒ **Start time**: Sun Nov 20 00:00:05 GMT 2005.
- ❒ **End time**: Mon Dec 05 10:30:24 GMT 2005.
- ❒ **Maximum number of machines**: 170.

- ❒ **Maximum number of computing elements**: 24,515.

Although the number of PEs of the real LCG testbed is 24,515, we do not know the real number of PEs involved in this experiment. So, after running some simulations, we decided to reduce the number of PEs in our simulated LCG testbed to those in the Table 2. We have reduced the number of PEs in order to force LCG saturation scenarios.

EXPERIMENTS

We have implemented two versions of GridWaySim that only differ in the scheduling policy they implement: the normal, and the enhanced scheduling policy. As a result, the scheduling policy is the only factor that can cause throughput variations between the different GridWaySim versions. Apart form that, all versions rely on the same configuration, with the same number of users that submit at the same time the same experiment with the same number of jobs (each with the same length, and input and output files size) to the same broker. Also, the number of brokers and resources is the same across the different GridWaySim versions.

Next we describe the exact configuration of the simulation:

- ❒ **Entities**: when we start the simulation GridWaySim creates 11 Users, 2 GridWay brokers, 1 DSATestbed, 1 LCGTestbed, and 1 Workload. Each of which is an independent thread attending petitions in their `body()` method.
- ❒ **Experiment**: every Experiment is a collection of 300 equal Jobs.
- ❒ **Job**: the main parameters of each Job are the length or size (in Million of Instructions, MI) of the Job to be executed, the input files (in bytes), and the output files (also in bytes) to be submitted to the corresponding resource. All Jobs have the same values for the three parameters: the size is 6,000,000 MI, the input file size is 1,000,000 bytes, and the output file size is 2,000,000 bytes.
- ❒ **User**: when we create the User, we have to indicate a submit time for her Experiment. Each user only submits one Experiment. In this simulation, each User submits her Experiment 24 hours after the previous one. The first User submits her Experiment at 12:00 of the first day of the simulation. Thus, each User submits her experiment to the DSA GridWay at 12:00 of the corresponding day of simulation.
- ❒ **Workload**: the Workload entity submits 188,041 jobs to the LCGTestbed at the time specified in the trace file.
- ❒ **DSATestbed**: simulates the resources described in Table 1. All the DSA resources uses the spaced-shared policy as the internal management policy (as defined in the GridSim API, this policy uses the First Come First Serve (FCFS) algorithm).
- ❒ **LCGTestbed**: simulates the resource with the machines described in Table 2. The LCG resource also uses the spaced-shared policy as the internal management policy.

As we mentioned before, to maximize the throughput we need to obtain the number of jobs that the DSA GridWay should submit to internal resources and to partner resources. We have used the equation that represents the best characterization of the Grid to obtain these numbers (Montero et al. (2006)). Thus, we need to run GridWaySim to obtain the linear equations of each infrastructure. Figure 3 shows throughput achieved by using the normal scheduling policy in DSA, LCG, and Federated Grid infrastructures for User-0 and User-3. It can be also seen the linear equations of both, DSA and LCG infrastructures (as function of time). We can represent the tasks executed by DSA, and LCG as follows:

$$t^{DSA}(x) = m^{DSA}x + b^{DSA} \tag{2}$$

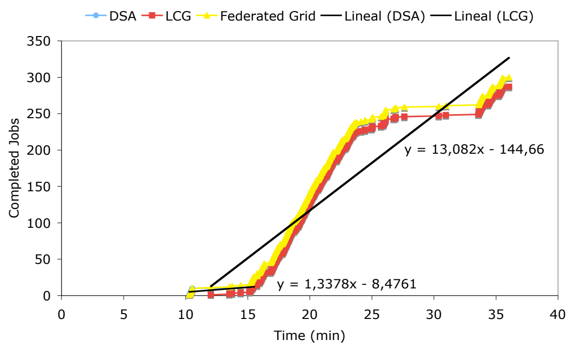
$$t^{LCG}(N-x) = m^{LCG}(N-x) + b^{LCG} \tag{3}$$

The *minimum* number of tasks that should execute DSA infrastructure is the point of intersection of these two lines. To determine this point we have to equal the linear Equations 2 and 3, which are functions of the completed tasks, and work out the values of m and b from Equation 1, which is function of time,

$$min = \frac{r_\infty^{DSA} n_{1/2}^{LCG} - n_{1/2}^{DSA} r_\infty^{LCG}}{r_\infty^{DSA} + r_\infty^{LCG}} + \frac{r_\infty^{DSA}}{r_\infty^{DSA} + r_\infty^{LCG}} N \tag{4}$$

Being N the total number of jobs (300), in the Equation 4 the min represents the maximum number of tasks that should be executed in the DSA infrastructure without increasing the makespan. Consequently, $N - x$ is the number of tasks that should be executed in the LCG infrastructure. Since there are only 2 participants in our proposed test scenario, the *minimum* method is enough to calculate the number of tasks to be executed in each infrastructure. However, in case of having 2 or more participants, we can determine the number of tasks to be executed in each participant by using the *aggregation* or *federation* model proposed in Vázquez et al. (2008).

Table 3 summarizes the number of executed and estimated tasks of User-0 and User-3. As mentioned before, the simulation creates 11 Users each one submitting 1 Experiment with 300 Jobs to the DSA GridWay. Instead



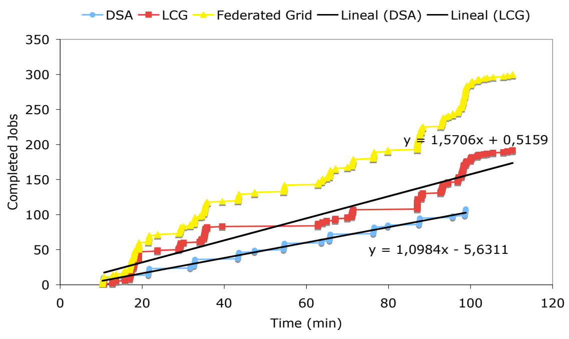
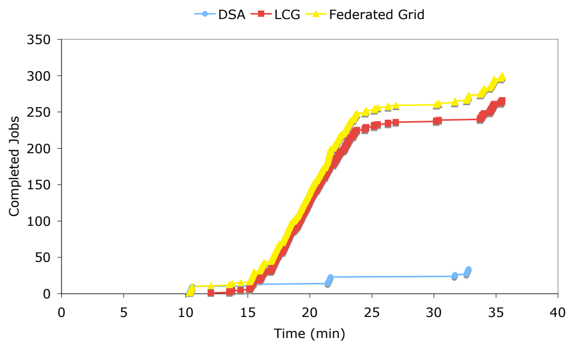


Figure 3: Throughput achieved by using the normal scheduling policy in DSA, LCG, and Federated Grid infrastructures for User-0 (left) and User-3 (right).



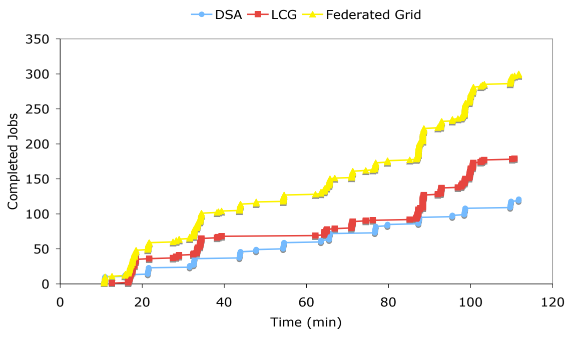


Figure 4: Throughput achieved by using the enhanced scheduling policy in DSA, LCG, and Federated Grid infrastructures for User-0 (left) and User-3 (right).

of providing the results of every User, we concentrate in two of them that represent different LCG saturation scenarios. Thus, simulation results show that User-0 represents an ideal scenario in which the LCG infrastructure always presents free PEs: *low saturation scenario*. As it can be seen in the column *Normal - DSA* of Table 3 the normal scheduling policy submits only 13 of 300 jobs to the DSA infrastructure. The *medium saturation scenario* is the one suffered by User-3, in this case the LCG resource has less free PEs, therefore 108 of 300 jobs are executed in the DSA infrastructure.

	Normal DSA – LCG	Estimated DSA – LCG
User-0	13 – 287	34 – 266
User-3	108 – 192	121 – 179

Table 3: Summary of the number of executed and of estimated jobs in both resources.

Column *Estimated* of Table 3 summarized the number of tasks that should be submitted to both infrastructures to increase the throughput, as depicted in Figure 4. Since we have changed the scheduling goal of GridWay, but not how GridWay achieves it, the completion time of the Experiments of every User obtained in the normal as well as in the enhanced scheduling simulation were the same, as you can see in Table 4. Thus, the estimation enhances GridWay normal scheduling policy: it maximizes DSA throughput while maintaining the makespan. Moreover, the enhanced algorithm not only maximizes the throughput of the DSA infrastructure under the same conditions, it also provides a fairness distribution of jobs between both resources compared with the normal policy: instead of abusing of the external grids, the DSA GridWay submits more jobs to its internal resources.

	Normal Makespan (min.)	Enhanced Makespan (min.)
User-0	36.05	35.52
User-3	110.27	111.75

Table 4: Experiment completion time for User-0 and User-3 in the normal and enhanced scheduling simulation.

CONCLUSIONS AND FUTURE WORK

In this paper we have presented two variations of GridWay's current scheduling policy that adapt to Federated Grids. Thus, the new scheduling policies has been built having in mind restrictions, such as the different types of users, and resources. We have also included the GridWaySim simulated environment to demonstrate that our enhanced scheduling strategy maximizes the throughput of internal resoruces, but without increasing the computational time, and provides a fairness distribution of the jobs by means of the r_∞, and $r_{1/2}$ parameters. Finally, the simulation results provided by GridWaySim show that the enhanced scheduling policy proposed improves GridWay's normal one.

Our current work focuses on the implementation of a scheduling policy that dynamically works out the values r_∞ and $n_{1/2}$. Also, we are adding more entities to our testbed to simulate more complex scenarios.

ACKNOWLEDGE

This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia and through the research grant TIN2006-02806.

Research work also supported by the Madrid Regional Research Council under project TIC-000285-0505.

REFERENCES

Andrieux, A., Berry, D., Garibaldi, J., Jarvis, S., MacLaren, J., Ouelhadj, D., and Snelling, D. (2003). "Open Issues in Grid Scheduling". Technical Report ISSN 1751-5971, UK e-Science Institute.

Dong, F. and Akl, S. G. (2006). "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems". Technical Report 2006-504, Ontario Queens University.

He, X., Sun, X., and von Laszewski, G. (2003). "QoS guided min-min heuristic for grid task scheduling". *Journal of Computer Science and Technology*, 18(4):442–451.

Hockney, R. and Jesshope, C. (1988). *"Parallel Computers 2: Architecture, Programming, and Algorithms"*. Adam Hilger Ltd.

Huedo, E., Montero, R. S., and Llorente, I. M. (2004). "A Framework for Adaptive Execution on Grids". *Software – Practice and Experience*, 34(7):631–651.

Montero, R. S., Huedo, E., and Llorente, I. M. (2006). "Benchmarking of High Throughput Computing Applications on Grids". *Parallel Computing*, 32(4):267–269.

Song, J., Koh, C.-K., See, S., and Leng, G. K. (2005). "Performance Investigation of Weighted Meta-scheduling Algorithm for Scientific Grid". In *Proceedings of the 4th International Conference on Grid and Cooperative Computing(GCC 2005)*, volume 3795, pages 1021–1030. LNCS.

Ullman, J. D. (1975). "NP-Complete Scheduling Problems". *Journal of Computer and System Sciences*, 10(3):384–393.

Vázquez, C., Fontán, J., Huedo, E., Montero, R. S., and Llorente, I. M. (2008). "A Performance Model for Federated Grid Infrastructures". In *Proceedings of the 16th Euromicro International Conference on Parallel, Distributed and network-based Processing (PDP 2008)*, pages 188–192.

Vázquez, C., Huedo, E., Montero, R. S., and Llorente, I. M. (2007). "Evaluation of A Utility Computing Mode based on Federation of Grid Infrastructures". In *13th International Euro-Par Conference (Euro-Par 2007)*. Lecture Notes in Computer Science (LNCS).

Wiriyaprasit, S. and Muangsin, V. (2004). "The Impact of Local Priority Policies on Grid Scheduling Performance and an Adaptive Policy-based Grid Scheduling Algorithm". In *Proceedings of the Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region (HPCAsia04)*, volume 0-7695-2138-X/04. IEEE.

SIMPLE NEAR OPTIMAL PARTITIONING APPROACH TO PERFECT TRIANGULAR ITERATION SPACE

Nedal Kafri
Department of Computer Science
Al–Quds University
Palestine
Email: nkafri@science.alquds.edu

Jawad Abu Sbeih
Department of Computer Science
Al Quds Open University
Palestine
Email: jabusbeih@qou.edu

KEYWORDS

Parallel Computing, Nested Loops, Static Partitioning, Load Balancing.

ABSTRACT

One of the most critical issues in parallel computing is the efficient distribution of a workload and data (workload balancing) amongst networked processors in multiprocessor and multicomputer systems to achieve optimal performance. Vast large scale scientific computing, such as numerical and Digital Signal Processing (DSP) problems have the nested loops as the main parallelized code segment. The main concern in partitioning the iteration space is the trade off between load balance, data locality, and minimizing scheduling overheads. Therefore, it is important to study and implement efficient decomposition techniques, which play an important role in achieving optimal performance and efficient use of multiprocessor and multicomputer systems. In this work, we focus on static decomposition of perfect triangular iteration space to achieve load balancing across given processors in a homogenous system. This paper introduces an intuitive near-optimal partitioning approach to triangular iteration space of a loop nest along the outermost loop index. Furthermore, the obtained partitions thus consist of contiguous non-overlapping parts which preserve data locality.

INTRODUCTION

Parallel computing is an efficient technique used to achieve high performance and efficient use of multiprocessor and multicomputer systems which speedup many applications over sequential processing on a single processor. The major goal of parallelization is to minimize the *elapsed time* (ie., overall computation time) by distributing the computation workload amongst the available processors evenly. This distribution can be done automatically by compilers (Chaundhary et al., 1996; Haghighat and Polychronopoulos, 1996; Hudak and Abraham, 1990; Jialin, 1998; Petkov et al., 2002), or manually by programmers using compiler directives and other different *decomposition* programming techniques (Fahringer, 1998; Hancock et al., 2000b; Kejariwal et al., 2005; Li et al., 2000; Sakellariou, 1996).

Significant amount of applications such as numerical, large scale scientific problems, Digital Signal Processing (DSP) (Li et al., 2000), computer vision, high-definition television medical imaging, remote sensing and many cryptographic algorithms, such as unchained Skipijak and DES (Petkov et al., 2002), are considered to be multi-dimensional problems. It is generally agreed by researchers that most of the computation time is spent in loops, which can be single (one-dimensional) or nested ones (multidimensional). In the case of nested loops, the iteration space (number of iterations of the loop body or workload) of these loops can be of constant, or iteratively either increasing or decreasing size. Thus, nested loops (loop nest) are the most important portion of these applications. Therefore, most researchers pay much of their attention to loop parallelization (D'Hollander, 1992; Chaundhary et al., 1996; Haghighat and Polychronopoulos, 1996; Hancock et al., 2000b; Hudak and Abraham, 1990; Jialin, 1998; Kejariwal et al., 2004, 2005; Li et al., 2000; Petkov et al., 2002; Polychronopoulos et al., 1986; Sakellariou, 1996; Xue et al., 2005).

Efficient parallel execution of these applications requires efficient *partitioning* and *mapping* of the iteration spaces of these nested loops across available processors to achieve perfect load balance. Therefore, it is important to study effective mapping techniques to gain significant speedup from parallelization. Thus, mapping of loop nests have received extensive attention in literature; mapping of loop nests with rectangular iteration spaces has received coverage in (D'Hollander, 1992; Polychronopoulos et al., 1986), whereas partitioning of loop nests with non-rectangular iteration space has been covered in (Haghighat and Polychronopoulos, 1996; Kejariwal et al., 2005; Sakellariou, 1996). However, (Haghighat and Polychronopoulos, 1996; Sakellariou, 1996) do not partition the iteration space uniformly across different processors and have several limitations, such as trade-off between *parallelism* and *data locality*. Furthermore, these approaches do not address the problem of partitioning iteration spaces with variable densities, i.e., loops with non-constant strides. Whereas in

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

(Kejariwal et al., 2005), they address the distribution of iteration spaces with variable densities based on geometric approach for computing the iteration space before mapping.

Based on whether the workload is distributed before or during run-time, loop partitioning can be classified as *static*, (where, usually partitioning is the most important aspect) or *dynamic* (Hancock at al., 2000a; Hancock et al., 2000b) (where, usually scheduling is the most important aspect), respectively. It should be noted that dynamic scheduling techniques may require additional communication and (runtime) overheads to achieve the load balanc. Furthermore, the static load balancing on *heterogeneous* systems, where the processors have different speeds and capacities is discussed in (Beaumont et al., 2002).

This paper focuses on *static* partitioning of loop nest with perfect *triangular* iteration space in *homogenous platforms*. It introduces an approximation–based approach that partitions the iteration space along the axis corresponding to the index of the outermost loop to achieve near optimal load balancing. The analytical and experimental results show that the proposed approach competes the best known techniques by its simplicity. Furthermore, the partition thus obtained consists of contiguous and disjoint subsets, which facilitates exploitation of data locality.

The rest of this paper is organized as follows: The next section presents a background relevant to the loop nest iteration space, partitioning problem, the terminology, notions, and definitions used in this paper. Thereafter, we introduce a new *approximation–based near optimal partitioning* approach to perfect triangular iteration space (ANOP). Next to that section, evaluation and experimental results are presented. Finally, we conclude and propose directions for future research.

BACKGROUND

In some applications, the workload balancing of the iteration space is primitive. This can be achieved by dividing the iteration space among P processors evenly into P parallel tasks (possibly into embarrassingly or trivially parallel tasks i.e., tasks that can be done independently of any other computation, without any communication among them). In other words, the load balance can be achieved by the decomposition of the iteration space into a collection of equivalent disjoint subsets (parts) of the iteration space, whose union is all of the iteration space (Scott et al., 2005). To illustrate this definition, consider a very simple summation example.

$$A = \sum_{i=1}^{N} a_i$$

This kind of operation is called reduction; it reduces the vector $(a_1, a_2, \ldots, a_N)$ to the scalar A. Assume for simplicity that P divides N $(P|N)$ i.e., N is an integer multiple, c, of P processors: $N = c.P$. Then we can divide the reduction operation into P disjoints partial sums:

$$A_k = \sum_{i=(k-1)c+1}^{ck} a_i, \;\; for\, k = 1 \ldots P,\; Then \qquad (1)$$

$$A = \sum_{k=1}^{P} A_k \qquad (2)$$

Considering Equation (1) and (2), we have managed to create P embarrassingly parallel tasks, having $c = N/P$ addition operations (workload) to do on c data points.

Similarly, one can use this simple decomposition technique of the workload of loop nests having a rectangular structure. Such loop nests and their iteration spaces can be represented by the pseudocode and its representation as a polytope in Figure 1(a). This geometric shape approach i.e., polytope representation, which has been used often since the early years of Lamport's hyperplane method (Lamport, 1974), allows us to deal with the problem from a geometric point of view hoping to provide a clearer understanding. Partitioning an iteration space along an axis corresponding to the outermost loop can achieve a perfect workload balance. More detailed discussion on rectangular loop partitioning techniques can be found in (D'Hollander, 1992; Polychronopoulos et al., 1986; Sakellariou, 1996).

However, those simple decomposition approaches in parallelizing cannot lead to workload balancing in many problems containing *non-rectangular* nested loops, such as *triangular* loop nests Figure 1(b). Vast number of triangular loop nests can be found in some matrix operations, for example, adding lower/upper triangular matrices, *LU* factorization problems, and prime numbers discovery.

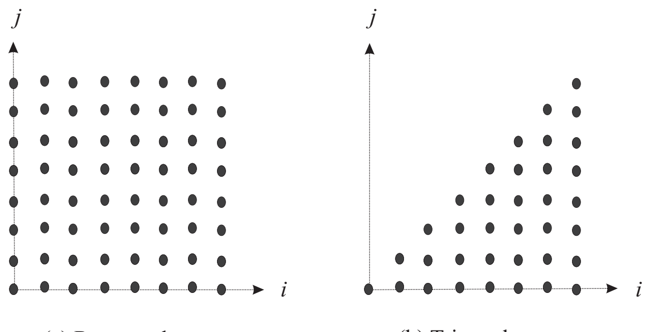
Triangular Loop Nests

Triangular loop nests (or triangular iteration space) means a loop nest consisting of an outer loop and an inner loop having bounds dependent on the index of the outer loop. Moreover, the operation(s) in the inner loop can be done independently of all others. Such triangular loop nests and their iteration spaces can be represented by the pseudocode and its representation as a polytope in Figure 1(b) (Haghighat and Polychronopoulos, 1996; Kejariwal et al., 2005; Sakellariou, 1996), where dots indicate the number of iterations of the inner loop.

It should be noted that the operation(s) in the inner loop can be as simple as shown in Figure 1(b), which adds two lower triangular matrices, constant sequence of operations or a multidimensional/multilevel independent nested loops with constant iteration spaces i.e., invariant iterator (Kejariwal et al., 2005).

```
for i = 1, N
    for j = 1, N

        (independent operations)
        (eg., C[i,j]=A[i,j]+B[i,j])

    enddo
enddo
```

```
for i = 1, N
    for j = 1, i

        (Independent operations)
        ( e.g., C[i,j]=A[i,j]+B[i,j])

    enddo
enddo
```



(a) Rectangular (b) Triangular

Figure 1: An Example to Illustrate Iteration Space of Loop Nests (a) Rectangular (b) Triangular

It is known that the total workload W (loop iterations) of such triangular loop nest shown in Figure 1(b) is equivalent to the sigma notation in Equation (3), where c represents the constant number of operations in the inner loop body (which can be assumed 1 for simplicity), and l and u are the lower and upper bounds of the outermost loop (i.e., along the i axis), respectively.

$$W = \sum_{i=l=1}^{u=N} \sum_{j=1}^{i} c = c \sum_{i=1}^{N} i = c\frac{N(N+1)}{2} \tag{3}$$

Detailed work on nested loops and computing loop iterations can be found in (Sakellariou, 1996).

Static Load Partitioning and Load Balancing

Partitioning the workload amongst processors of a multiprocessors or a multicomputer system plays a critical role in parallel processing. One of the requirements for decomposition is that the workload to be balanced across all the processors. If the work is not distributed equally, then one processor may end up taking longer time than the others. Since we are doing a cooperative project, the overall job cannot be accomplished until the slowest subtask is finished. Thus, Workload balancing is one of the critical issues and goals which plays an important role in parallel computing. Efficient and perfect distribution of the workload will enable to achieve this goal, and consequently will affect the overall performance and the efficiency of the parallel processing. The author in (Sakellariou, 1996) discusses the necessary conditions for partitioning a loop nest into *equal* workload. It should be noted that this work discusses static workload partitioning on *homogenous* systems, where the processors have *equal* speeds and capacities.

Assume that a set of P parallel tasks (indexed by $k = 1, \ldots, P$), executes its assigned workload W_k in an amount of time t_k. Furthermore, assuming that W'_k represents the set of those operations whose workload is W_k, then $W' = \cup_{k=1}^{p} W'_k$ and, for any two subsets, $W'_i, W'_j, 1 \leq i, j, \leq P, i \neq j$, it must be the case that $W'_i \cap W'j = \theta$. Consequently, it is clear that $W = \sum_{k=1}^{p} W_k$.

In (Scott et al., 2005) they define the average execution time and the load balance in term of time. Since we rarely predict the exact execution time in advance which is almost a factor of the workload in homogenous platform (i.e., $t_k = cW_k$), whereas we can frequently predict the workload in advance, we can redefine the average workload:

$$ave\{W_k : 1 \leq k \leq P\} = \frac{1}{P}\sum_{k=1}^{P} W_k$$

Furthermore, one can define the load balance β, resulting from a predetermined decomposition strategy, to be the ratio of the average workload to the maximum workload (i.e., to the biggest task):

$$\beta = \frac{ave\{W_k : 1 \leq k \leq P\}}{max\{W_k : 1 \leq k \leq P\}} \tag{4}$$

Therefore; a set of tasks is said to be load balanced if β is closed to one. The *optimal* (*perfect*) load balance W_{opt} is achieved if, for all $k, 1 \leq k \leq P$,

$$W_k = \frac{W}{P} = W_{opt} \tag{5}$$

in this case, β equals one.

On the other hand, inefficient distribution leads to *load imbalance D*. Whenever there is a process k for which the *difference* or *deviation* $D_k = W_k - W/p$ is non-zero, then the workload assigned to *k-th* processor exhibits an imbalance equal to D_k. Negative value of D_k means that *k-th* processor is assigned less workload than the average share, while positive value of D_k indicates that processor k is assigned more workload than the average workload and can be considered *'overloaded'*. As a result of this overload, the latter processor determines the overall parallel computation time. Consequently, the exhibited load imbalance, D, resulting from a given distribution of the workload W amongst P processors (Sakellariou, 1996), given by

$$D = \max_{1 \leq k \leq p}(D_k) = \max_{1 \leq k \leq p}(W_k - \frac{W}{p}) = W_{max} - \frac{W}{p} \tag{6}$$

In order to assess the impact of the *load imbalance* D on the overall parallel elapsed time, Sakellariou in (Sakellariou, 1996) introduces a *relative load imbalance* D_R:

$$D_R = \frac{D}{W_{max}} = \frac{W_{max} - \frac{W}{P}}{W_{max}} = 1 - \frac{W}{PW_{max}} \quad (7)$$

It is clear that D_R takes values in the interval $[0, 1 - 1/p]$. Values close to zero denote perfect load balance and a closed to linear-speedup is obtained. Whereas, non-zero values denote load imbalance, values closed to $1 - 1/p$ denote highly imbalanced workload decomposition, and no significant speedup from parallelization can be expected. Thus, in order to increase the gains from parallelization, the workload requires distribution of the workload as evenly as possible. Therefore, we must search for a strategy for mapping the workload such that the *relative load imbalance* will be close to zero, and the load balance β will be close to one.

In order to achieve static workload balance in a homogenous architecture, several approaches were introduced in the literature. The most common approaches for static loop partitioning on homogenous systems are:

- Block Partitioning (BP) (Kruskal and Weiss, 1985): A simple way to distribute the iteration space (workload) of a single loop or a rectangular loop nest is *block partitioning* BP. Assume that P is the number of processors and N is the iteration determined by the outer loop, where $(P|N)$, one can gain perfect workload partitioning using this technique. Block partitioning distributes contiguous equal iterations onto processors in a consecutive manner, which in turn preserve *data locality*. Thus, Processor 1 executes iterations 1 through N/P. Process 2 executes iterations $N/P + 1$ through $2N/P$. In general the *k-th* processor, for all $k, 1 \leq k \leq P$ executes iterations $((k-1)N/P + 1)$ through (kN/P).
- Cyclic Partitioning (CP): The second common distribution approach that can be used to distribute loop workload is called *cyclic partitioning* CP or *stride mapping*. Stride technique is a widely used technique, to decompose the loop iterations as evenly as possible with each process being assigned a fixed number of iterations of the outer loop in a round robin fashion; process 1 executes iterations 1, $P+1$, $2P + 1$,.., process 2 executes iterations 2, P+2, 2P+2, $\ldots$. In general, the *k-th* processor, for all $k, 1 \leq k \leq P$ executes iterations $k + iP$, $i = 0, 1, 2, \ldots, (N/P - 1)$. Similar to BP approach, the partitions of a rectangular iteration space thus obtained consist of equal workload. However, CP generates a fragmented partition (i.e. each individual set is a collection of non-contiguous subsets). Consequently, this approach may exhibit poor performance in many applications due to *false sharing* (a well known phenomenon in computer architecture).

With *triangular loop nest* partitioning, (BP) and (CP) approaches are no longer sufficient; These approaches may generate *unequal* parts. More efficient approaches for partitioning triangular iteration space consisting of independent loops are:

- Balanced Chunk Scheduling (BCS) (Haghighat and Polychronopoulos, 1993): Unlike block and cyclic partitioning which distribute only the iterations of the outer loop, BCS attempts to partition the total number of iterations of the loop nest body among processors as evenly as possible. An example of the latter is shown in Appendix B of (Sakellariou, 1996). However, Haghighat and Polychronopolous restrict their discussion to double loop. Moreover, requires predetermination of the total number of index points in the iteration space.
- Canonical Loop Partitioning (CLP) (Sakellariou, 1996): Sakellariou introduces a notion of *canonical loop nest* for loop mapping. CLP assumes that the outermost loop can be partitioned into $2P^{m-1}$ equal parts, where P is the number of processors, and m is the depth of a loop nest. However, this may generate empty subsets which lead to a load imbalance. Moreover, CLP generates a non-contiguous partition. CLP employs an enumeration-based approach to determine the total number of index points in an iteration space. It relies on loop normalization in the presence of non-unit strides. However, the introduction of floors and ceilings renders this approach nonviable in practice. Furthermore, determination of the set boundaries in CLP is very cumbersome.
- Weight-Based Partitioning WBP (Kejariwal et al., 2005): This approach discusses the partitioning of loop nest with N-dimensional non rectangular iteration space, with variable densities following a geometric approach. Based on the assumption that there do not exists any invariant iterator, it introduces a procedure for partitioning an iteration space. The procedure consists of three major steps. First, it computes a partial weight of the convex polytope as a function of the outermost index variable. They follow a weight-based approach for estimating the number of index points in a polytope. Next, the algorithm computes the total weight of the convex polytope corresponding to loop nest. Finally, it determines the breakpoints along an axis corresponding to the outermost index. WBP algorithm achieves *near-optimal* and *contiguous* partitions of an iteration space.

APPROXIMATION–BASED PARTITIONING

This work focuses on a perfect loop nest of depth 2 as shown in Figure 1(b), where the bound of the inner loop depends on the index of the outer loop. We attempt to discover an efficient, intuitive, and simple approach to distribute total iteration space among P processors as evenly as possible (i.e., optimizing the load balance), so that the load balance β is maximized, and the load imbalance D as well as the relative load imbalance D_R are minimized as a result.

For simplicity, assume that the constant $c = 1$ (in Equation 3), our goal becomes equivalent to finding the optimal workload (Equation 5) for all $k, 1 \leq k \leq P$ such that

$$W_k = W_{opt} = \frac{W}{P} = \frac{N(N+1)}{2P}$$

Now, Consider Figure 1(b) and the corresponding workload in Equation (3). Partitioning such loop nest (along the axis corresponding to the outermost loop, where the lower bound $l = 1$ and the upper bound $u = N$ into equi-workload (E_k) across P processors, is equivalent to discovering the perfect lower bound l_k and the upper bound u_k of the outer loop assigned to the *k-th* processor (i.e., the *k-th* partition), for all $k, 1 \leq k \leq P$, such that

$$|E_k - \frac{W}{P}|$$

is minimized (Sakellariou, 1996), where

$$\sum_{i=l=1}^{u=N} \sum_{j=1}^{i} 1 = \sum_{k=1}^{P} \sum_{i=l_k}^{u_k} \sum_{j=1}^{i} 1 = \sum_{k=1}^{P} W_k$$

$l_1 = l = 1, u_p = u = N$

$$E_1 = \sum_{i=l_1}^{u_1} i = \frac{u_1(u_1+1)}{2} \tag{8}$$

and, for all $2 \leq k \leq P$, $l_k = u_{k-1} + 1$

$$E_k = \sum_{i=l_k}^{u_k} \sum_{j=1}^{i} 1 = \frac{u_k(u_k+1)}{2} - \frac{u_{k-1}(u_{k-1}+1)}{2} \tag{9}$$

Recalling that $l_1 = l = 1$ and $l_k = u_{k-1} + 1$, for all $2 \leq k \leq P$, we can express the problem of finding the perfect load balance as that of finding integer u_k, $1 \leq k \leq P$, such that

$$E_k = \sum_{i=l_k}^{u_k} \sum_{j=1}^{i} 1 \approx \frac{W}{P}$$

Based on Equation (9), it is clear that

$$\sum_{j=1}^{k} E_j = \sum_{i=l_1}^{u_k} i = \frac{u_k(u_k+1)}{2}$$

and

$$\frac{k}{P}W = \frac{k}{P}\frac{N(N+1)}{2},$$

we can formulate the following approximation equation:

$$\sum_{j=1}^{k} E_j = \sum_{i=l_1}^{u_k} i = \frac{u_k(u_k+1)}{2} \approx \frac{k}{P}W = \frac{k}{P}\frac{N(N+1)}{2}$$

Consequently, we obtain

$$u_k^2 + u_k = \frac{k}{p}N^2 + \frac{k}{p}N \tag{10}$$

Assuming that the difference between the terms u_k and $\frac{k}{p}N$ in Equation (10) is very small (i.e., $u_k \approx \frac{k}{p}N$), it has no significant impact on the equation in front of the exponent terms (u_k^2 and $\frac{k}{p}N^2$). Therefore, they can be omitted from the equation, for the purpose of simplicity. Thus, the direct solution for the *k-th* upper bound u_k is

$$u_k = N\sqrt{\frac{k}{p}}$$

Clearly, it appears that integer solutions for u_k are not common; thus rounding u_k to nearest integer, the formula becomes:

$$u_k = round(N\sqrt{\frac{k}{p}}) \tag{11}$$

In the following section, we introduce an example using this approach with the gained analytical and experimental results.

EVALUATION AND EXPERIMENTAL RESULTS

To illustrate and evaluate the proposed partitioning approach, consider the code segment shown in Figure 1(b) for $N = 800$, which adds two upper triangular matrices (800x800 matrices). A similar example was provided in (Sakellariou, 1996)(sec. 2.3.1.1), to illustrate Balanced Chunk Scheduling originally approached in (Haghighat and Polychronopoulos, 1996). In order to partition the iteration space along the outer loop (i loop) across 8 processors, the iterations are distributed using the proposed approach following Equation (11). As shown in Table 1, processor 1 executes iteration $l_1 = 1$ through $u_1 = 283$, processor 2 executes iterations 284 through 400, and so on. Also, the table shows the corresponding workload E_k using Equations 8 and 9, the load deviation assigned to the *k-th* processor ($D_k = E_k - \frac{W}{P}$),

Table 1: Partitioning Loop Nest (for $N = 800$) Across 8 Processors, Corresponding Workload E_k, Workload Deviation D_k, and D_k/E_{max}

P_k	Iteration Distribution			
	$l_k - u_k$	E_k	D_k	D'_k
1	1-283	40186	136	0.0034
2	284-400	40014	-36	-0.0009
3	401-490	40095	45	0.0011
4	491-566	40166	116	0.0029
5	567-632	39567	-483	-0.0119
6	633-693	40443	393	0.0097
7	694-748	39655	-395	-0.0098
8	749-800	40274	224	0.0055

and the ratio of the *k-th* workload deviation to the maximum workload D'_k (where $D'_k = \frac{D_k}{E_{max}}$).

As a result of this distribution, the achieved average workload is equal to the perfect workload $\frac{W}{P} = 40050$, and the maximum assigned workload $W_{max} = W_6 = 40443$. Consequently, the exhibited load balance $\beta = 0.994438$ (equation 4) is close to 1, which indicates that the *near-optimal* partitioning of the workload W amongst P processors is achieved.

Similarly, it is clear that the resulting load imbalance is equal to ($D = 393$), according to Equation 6 and the *relative load imbalance* D_R using Equation 7 is equal to (0.0097), which can be considered close to zero.

It should be noted that when implementing our approach in the provided example in (Haghighat and Polychronopoulos, 1993) (for Balanced Chunk Scheduling BCS approach), as well as in the provided case study in (Kejariwal et al., 2005) for Weight-Based Partitioning WBP approach, it obtains same results.

In order to illustrate the *Weight-Based Partitioning* WBP technique for a triangular non-uniform iteration space (Kejariwal et al., 2005), they consider the well known *Sieve of Eratosthenes* (TSoE) algorithm that identifies all prime numbers up to a given number M . The following shows the kernel (loop nest) of the (TSoE) for $N = \sqrt{M}$:

```
doall i = 3, N, 2
   doall j = i, M, 2*i
        LOOP BODY
   end doall
end doall
```

To distribute the workload amongst P processors according to the WBP, we have to follow several steps: begin with checking the existence of an invariant iterator. If exists, then determine a partial weight of the convex polytope corresponding to the iteration space using a geometric approach. Next, determine the total weight of the convex polytope. Finally, the breakpoints (i.e., bounds) for the partitions can be determined. Table 2, shows a comparison of the determined upper bounds (u_k, for $1 \leq k \leq P$) (before rounding the results into integer numbers for comparison purpose) using WBP and our proposed (ANOP) mapping approaches for $M = 10000$ and $N = 32$ on different number of processors P.

Table 2: The Determined Upper Bounds (u_k) for Partitioning Loop Nest (of $N = 32$) Across 2 Processors, 3 Processors, and 4 Processors

	Determined Upper Bounds (u_k)	
	$P = 2$	
k	$ANOP$	WBP
1	22.63	22.66
2	32.00	32.00
	$P = 3$	
k	$ANOP$	WBP
1	18.48	18.57
2	26.13	26.14
3	32.00	32.00
	$P = 4$	
k	$ANOP$	WBP
1	16.00	16.13
2	22.63	22.67
3	27.71	27.72
4	32.00	32.00

The results of the provided examples show that the obtained partitions using our ANOP, the BCS, and the WBP mapping approaches are almost the same. Thus, we can expect equivalent performance of these mapping approaches and better performance than the BP and the CP techniques.

To analyze and evaluate the complexity of parallel computing, a number of performance metrics have been used over the years. The most common being *elapsed time*, *speedup*, and *efficiency*. The *elapsed time* (parallel runtime), T_P, is the time elapsed from the start of parallel computation to the moment when the last processor finishes execution. Beside workload W (problem size), it depends on the number of processors, P, the architecture of the parallel computing platform, and the algorithm. Therefore, the parallel runtime is measured by counting *computational* time and various classes of overheads: *unparallelised code*, *parallel start-up*, *synchronization*, *load imbalance*, and *communication* (routing) overheads (Sakellariou, 1996). The *speedup*, S, is defined as the ratio of the time taken to solve a problem on a single processor, T_S, using the best known version of a program to the time required to solve the same problem on a set of P parallel processors T_S/T_P. Finally, efficiency is defined as the ratio of of speedup to the number of processors S/P.

In order to evaluate the performance gains of the partitioning approach introduced in this work, several experiments have been conducted. Our goal has been to justify and assess the effectiveness of the load balance achieved by the proposed approach, and to justify the theoretical results.

In essence, three mapping techniques have been evaluated; namely Cyclic Partitioning CP, Block Partitioning, and our ANOP technique. The benchmark used in our experiments contains a loop nest shown in Figure 1(b) (for $N = 300000$). These experiments were implemented using the *Java Parallel Virtual Machine* (JPVM) (Ferrari, 1997), which is an explicit message–passing based parallel programming interface library and similar to the well known (PVM) library. Furthermore, the experiments were carried out on a platform that consists of 8 personal computers (PC's). These computers were of 3000 Mhz Pentium 4 processors with 512 MB of DDR2-RAM and each runs its own *Windows XP* operating system. The nodes were interconnected using $100Mbps$ Ethernet local area network (LAN).

The experiments were run several times on different number of computers. Figure 2 shows the average total parallel computational time (elapsed time) of the loop nest (for $N = 300000$) on single, 2, 4, 6, and 8 computers. The figure depicts the performance of the three tested mapping approaches. It is clear that the proposed approach ANOP performs better than the BP and CP approaches.

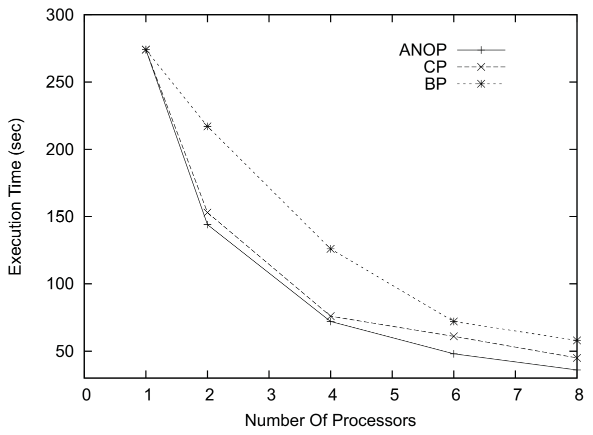


Figure 2: Execution Time by Different Partitioning Approaches and Number of Processors

Furthermore, Figure 3 reveals the *speedup* obtained by increasing number of processors by different approaches. It is clear that the ANOP achieves linear speedup and performs better than the other techniques. Though there is no interprocess communication and synchronization during runtime of the used benchmark program, the speedup is significantly lower than the number of processors, which is obvious. This can be related to the communication overhead caused by the used network platform. On the other hand, these figures show the impact of the load imbalance overhead, when using the BP and the CP techniques, on the elapsed time and speedup.

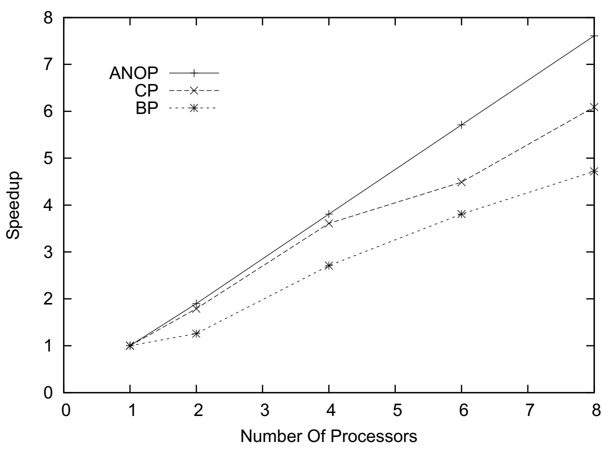


Figure 3: Speedup by Different Partitioning Approaches and Number of Processors

CONCLUSION AND FUTURE WORK

In this paper, we introduced a novel static simple intuitive approach for partitioning the iteration space of a perfect triangular loop nest (double loop nest) for homogenous environment of processors. We partition an iteration space along an axis corresponding to the outermost loop among processors and achieve a near–optimal partitioning. The partition thus obtained consists of contiguous and disjoint subsets, which facilitates exploitation of data locality. Moreover, the proposed approach can be easily implemented: unlike the other near optimal studied techniques, it does not need precomputation of the workload; a near optimal partitioning can be achieved by determining the lower and upper bounds (of the outermost loop index for each processor to carry out) as a function of the processor's number, total number of involved processors, and the upper bound of the original outermost loop.

The analytical results show that, using the proposed approach, we can achieve near optimal load balance and can minimize the load imbalance in parallel processing of a perfect triangular loop nest. Furthermore, the conducted experiments assess and justify the analytical results. The experimental results show the impact of the load imbalance on the elapsed time and the speedup achieved by different approaches.

As a future work, we would like to extend our approach to partition iteration space dynamically in a heterogenous environment.

REFERENCES

Beaumont, O.; V. Boudet; A. Legrand; F. Rastello; and Y. Robert. (2002). "Static Data Allocation and Load Balancing Techniques for Heterogeneous Systems". In *C.K. Yuen, editor, Annual Review of Scalable Computing*, volume 4, chapter 1. World Scientific.

Chaundhary, V. et al. (1996). "Design and Evaluation of an Environment APE for Automatic Parallelization of Programs".

In *Proceedings of the 1996 International Symposium on Parallel Architectures,* Algorithms and Networks Page: 77.

D'Hollander, E. H. (1992). "Partitioning and Labeling of Loops by Unimodular Transformations". *IEEE Transaction on Parallel and Distributed Systems,* 3(4):465-476.

Fahringer, T. (1998). "Efficient Symbolic Analysis for Parallelizing Compilers and Performance Estimator". *The journal of Super Computing*, 12, 227-252.

Ferrari, A. J. (1997). "JPVM:Network Parallel Computing in Java". Technical Report CS-97-29 Department of Computer Science University of Virginia, Charlottesville, VA 22903, USA. (December), available from: http://www.cs.virginia.edu/jpvm/doc/jpvm-97-29.pdf.

Haghighat, M. and C. Polychronopoulos. (1993). "Symbolic Analysis: A Basis for Parallelization, Optimization, and Scheduling of Programs". In *Proceedings of the Sixth Workshop Languages & Compilers for Parallel Computing*, (Aug.) 1993. available from: http://citeseer.ist.psu.edu/51605.html.

Haghighat, M. and C. Polychronopoulos. (1996). "Symbolic Analysis for Parallelizing Compilers". *ACM Transactions on Programming Languages and Systems*, 18(4):477-518,(July).

Hancock, D. J.; J. M. Bull; R. W. Ford; and T. Freeman. (2000). "Feedback Guided Dynamic Scheduling of Nested Loops". In *Proceedings of IEEE International Workshop on Parallel Processing* IEEE Computer Society Press, Pages 315-321, Elsevier Science (January), ISBN 0769507719.

Hancock, D. J.; J. M. Bull; R. W. Ford; and T. Freeman. (2000). "An Investigation of Feedback Guided Dynamic Scheduling of Nested Loops". In *Proceedings International Workshop on Parallel Processing 2000*, Toronto, Ont., Canada, pp 315-321. ISBN:0-7695-0771-9.

Hudak, D. E. and S. G. Abraham. (1990). "Compiler Techniques for Data Partitioning of Sequentially Iterated Parallel Loops". *ACM.*

Jialin, Ju. (1998). "Automatic Parallelization of Non-uniform Loops". PHD thesis, Jan, 1998 , Wayne State University, 155 pages; AAT 9915677, ISBN 9780599143999.

Kejariwal, A.; P. DAlberto; A. Nicolau; and C. D. Polychronopoulos. (2004). "A Geometric Approach for Partitioning N-Dimensional Non-Rectangular Iteration Spaces". In *Proceedings of the 17th International Workshop on Languages and Compilers for Parallel Computing*, West Lafayette.

Kejariwal, A.; A. Nicolau; U. Banerjee; C. Plychronopoulos. (2005). "A Novel Approach for Partitioning Iteration Spaces with Variable Densities". *Symposium on Principles and Practice of Parallel Programming*, (PPoPP),(June).

Kruskal, C. P. and A. Weiss. (1985). "Allocating Independent Subtasks on Parallel Processors". *IEEE Transactions on Software Engineering*, 11(10):10011016.

Lamport, L. (1974). "The Parallel Excecution of Do Loops". *Communication of the ACM*, 17-2,(Feb.), pp. 83-93.

Li, Y.; T. Callahan; E. Darnell; R. Harr; U. Kurkure; and J. Stockwood.(2000). "Hardware-software Co-design of Embedded Reconfigurable Architecture". Proceedings *37th conference on Design Automation*, pp 507-512, Los Angeles, Ca.

Petkov, D.; R. Harr; and S. Amarasinghe. (2002). "Efficient Pipelining of Nested Loops: Unroll-and-Squash". In *16th International Parallel and Distributed Processing Symposium*, Fort Lauderdale, Florida, (April).

Polychronopoulos, C.; D. J. Kuck; and D. A. Padua. (1986). "Execution of Parallel Loops on Parallel Processor Systems". In *Proceedings of the 1986 International Conference on Parallel processing*, page 519-527, (August).

Sakellariou, R. (1996). "On the Quest for Perfect Load Balance in Loop-Based Parallel Computations". PhD Thesis, Department of Computer Science, University of Manchester,1996. available at http://citeseer.ist.psu.edu/sakellariou98quest.html.

Scott, L. R.; T. Clark; and B. Bagheri. (2005). *Scientific Parallel Computing.* Princeton University Press.

Xue, C.; Z. Shao; M. Liu; and E. H.-M Sha. (2005). "Iterational Retiming: Maximize Iteration-Level Parallelism for Nested Loops." *Proceedings of the 2005 ACM/IEEE/IFIP International Conference on Hardware - Software Codesign and System Synthesis* (ISSS-CODES'05), New York, (Sept.).

AUTHOR BIOGRAPHIES

NEDAL M.S. KAFRI was born in Attil, Palestine and received his education at the Technical University of Plzeň in Czech Republic, where he obtained his MSc. degree in Control Systems and Installation Management in 1982. He worked at Al-Quds University as a lecturer at the Computer Science Department for fifteen years. Then he moved to the Czech Technical University in Prague in 1997, where he obtained his Ph.D. degree in Distributed Systems in 2002. Now he is a member of the academic staff of the Department of Computer Science of Al-Quds University in Palestine. His research interest is in Distributed and Parallel Computing. His email is `nkafri@science.alquds.edu` and his Webpage is `http://www.alquds.edu/staff/kafri`.

JAWAD ABU SBEIH was born in Hebron-Palestine in 1969 and obtained his first degree in Computer Science from Al-Quds Open University (QOU). His MSc degree was obtained from Al-Quds University in 2008. He is interested in parallel and distributed systems. He worked at Al-Quds Open University since 1993 as Teacher Assistant and the Continuous Learning Center coordinator at QOU. Now he is a lecturer in the department of computer science at QOU. His email address is `jabusbeih@qou.edu`. The website of QOU is `http://www.qou.edu`.

Databases in Grid and High Performance Computing Environments

GRID DATABASE MANAGEMENT: ISSUES, REQUIREMENTS AND FUTURE DIRECTIONS

Sandro Fiore, Salvatore Vadacca, Alessandro Negro and Giovanni Aloisio
University of Salento & SPACI Consortium
Euro Mediterranean Centre for Climate Change
viale Gallipoli, 49 – 73100 Lecce - Italy
E-mail: {sandro.fiore, salvatore.vadacca, alessandro.negro, giovanni.aloisio}@unile.it

KEYWORDS

Data Grid, Grid Database, Database Access, Database Management, Database Integration.

ABSTRACT

Data grids allow to store, manage and share large data collections, huge amount of files, geographically distributed databases, etc. across virtual organizations. Grid data and metadata management is becoming more and more important as the number of involved data sources is continuously increasing and decentralizing. Grid database management services represent a basic and fundamental building block for the next generation petascale production grids. In this paper we present the fundamental concepts related to grid-database access, management and integration, highlighting main requirements and issues, describing research activity in this area and dealing with new Open Grid Forum related standards.

INTRODUCTION

Grids encourage and promote the publication, sharing and integration of scientific data, distributed across Virtual Organizations (Foster and Kesselman 1998). Grid computing is widely regarded as a new field, distinguished from traditional distributed computing owing to its main focus on large-scale resource sharing and innovative high-performance applications.

It can be considered as an enabling paradigm allowing organizations to easily share, virtualize, integrate and efficiently manage wide spread resources (computational and data sources, sensors and instrumentations, etc.) in a grid environment. Grids link together servers, data sources, tools, services and applications into a single environment by means of a specific glue named *grid middleware*.

In the last years many efforts were devoted to the management (both coarse and fine grained) of data (grid-storage services, storage resource managers, metadata services, replica catalogues, grid-database access and integration services, etc.). Data management represents the real challenge for the next generation petascale grid environments since current production grids are able to produce hundreds of petabytes of data in the next years.

In the last few years, there was an increasing interest in fine grained (database related) grid data management activities and services connected with database access, metadata management, data integration, data transformation, data flow, etc.

Grid Services for database access and integration play a strategic role and provide added value to a production grid environment since they allow to aggregate data, join datasets stored at different sites, infer new knowledge by analyzing structured and distributed data, manage monitoring and accounting information, etc.

In this paper we will talk about the fundamental concepts related to grid-database access, management and integration, highlighting main requirements and issues, describing research activity in this area and dealing with new Open Grid Forum related standards.

The outline of the paper is as follows: in the 2nd section we talk about data grids, whereas in the 3rd section we describe in detail key issues. The 4th section concerns data requirements for grid database services as well as the 5th section shortly introduces new standards and specifications in this area. The 6th section recalls related works whereas the 7th section describes a specific case study (GRelC Project). Finally, we draw our conclusions.

GRID PARADIGM AND DATA GRIDS

Production grids (i.e. EGEE, Teragrid (Pennington 2002)) produce huge amounts of scientific data that must be available “on the grid” to scientific and engineering applications for further analysis and computations.

Data Grids provide the proper data service framework for Computational Grids, creating virtualized access to widespread data sources (both files and databases). Even if in the last years research activities related to Data Grids have mainly focused on coarse grained data management (files), accessing and integrating legacy/new databases (records) is becoming a fundamental issue.

A grid infrastructure is basically made up of two components: computational and data grids.

While a *computational grid* provides the computing power needed to run applications, a *data grid* provides a robust framework for data management services that enables data access, integration, transfer, replication, virtualization, distribution, etc.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

GRID-DBMS: KEY ISSUES

A grid database management system (Aloisio et al. 2004) should provide transparent, secure and efficient *management* (in terms of database access, integration, federation, transformation, etc.) of data sources (relational, hierarchical, object-oriented, etc.) in a grid environment. Since the beginning of the *grid era* many efforts were directed towards *computational access* (e.g. job submission/monitoring, Globus GRAM (Foster 2005), etc.) and *storage management* (e.g. file transfer, GridFTP (Allcock et al. 2005) based storage, SRB (Rajasekar et al. 2003)). Grid database management was addressed starting from the year 2000 (EDG-Spitfire (Bell et al. 2002), GRelC (Aloisio et al. 2005), and OGSA-DAI (Antonioletti et al. 2005)).
In the following we describe some basic elements connected with database management in distributed environments, highlighting how they impact on the application domains (i.e. e-Science) and why they are so relevant for end users (i.e. scientists). In particular, next subsections will be devoted to the discussion of data representation, data organization, data models, query languages, data access, data integration, access control and data flow.

Data Representation

In order to be domain-independent, data grids must provide support (in terms of access and management) to every type of data format, structure and representation. Data can be both structured and unstructured, characterized by different formats, coding, precision, accuracy and semantics.
Some examples concern bioinformatics (i.e. textual files, relational data sources), astrophysics (i.e. relational DBMS with postGIS extensions), climate scientists (i.e. XML) data banks.

Data Organization, Data Models and Query Languages

Data can be organized following several data models such as relational and hierarchical. Support in terms of relational or XML engines is widely provided by existing systems: Postgres, MySQL, IBM/DB2, etc. as well as XIndice, eXist, etc. Such DBMSs provide full support in terms of database access and management functionalities, API, SDK, CLI, etc. Different data models adopt different query languages such as SQL (for the relational one) and XPath and XQuery (for the hierarchical one); data grids must provide support to all of them.

Data Access

Even if DBMSs provide a lot of functionalities for the management of data sources, they are not fully compatible with existing grid middleware (i.e. gLite or Globus (Foster and Kesselman 1997)). They can be accessed in grid by using a "*grid-DBMS*" interface. This grid interface (which has to play a front-end role) must, obviously, provide full support to all of the query languages (SQL, XQuery, XPath, etc.) concerning the target data resources (transparency requirement with regard to the query language).
The specific part of the grid-DBMS that makes a data resource accessible in grid (or "grid enabled") is called Grid Database Access Service (Grid DAS).
It must provide secure, transparent, robust and efficient access to heterogeneous and distributed databases exposing standard interfaces to enable interoperability with other grid components and/or services.
Several research projects exploit the *service-in-the-middle* or *front-end* approach to provide such kind of functionalities, that is, they focus on the development of a transparent, secure and robust grid interface to existing DBMSs. On the contrary, vendor-specific products (i.e. Oracle 11g) generally exploit an *embedded approach* providing within the product, software modules to run on a grid environment (e.g. GSI (Tuecke 2001) support).

Data Integration

While the Grid DAS is a basic service to expose databases in grid (it provides a first level of virtualization), the Grid Data Integration Service (Grid DIS) is a further necessary building block if we want to provide aggregation capabilities (second level of virtualization). An interesting example is the OGSA-DQP (Alpdemir et al. 2003).
A Grid DIS can be centralized or distributed and in some cases it is integrated into the related Grid DAS providing what we call a Grid DAIS (e.g. GRelC DAIS which will be described later).
Data integration is strongly challenging since it allows both to integrate data within several *application-level* domains (bioinformatics, astrophysics, financial, etc.) and *system-level* distributed environments for monitoring and accounting purposes, etc.

Data Access Control

Data access control is more important to ensure that the confidentiality of the data is preserved/maintained against unauthorized accesses.
The facilities that the Grid provides to control access must be very flexible in terms of the combinations of restrictions, available policies, etc. User-centric and VO-centric data access control allow managing policies at each level of granularity addressing local site autonomy and user-level policies management (in the first case) and flexibility, scalability and manageability in the VO-level policies management (in the second case).
A combined User-VO data access control allows mixing the benefits related to the two approaches (any combination of insert, update, and delete privileges can be defined with the right level of granularity).
Moreover, the Grid must provide the ability to control access based on user role (as it usually happens for DBMSs). Role based access control is fundamental for collaborative working, when several individuals may perform the same role at the same time and provides a

scalable and manageable way to split users in sub-classes with specific and well-known privileges. Granting and revoking activities must be dynamically performed by administrators and should be easily carried out by using high level interfaces such as data grid portals.
Data access policies should be managed at the Grid-DBMS layer, without entirely relying on the back-end framework. This could (i) enable data access control for trivial data resources such as text files and (ii) prevent the access attempts to the back-end systems for unauthorized users.

MAIN REQUIREMENTS

In the following we highlight key requirements related to grid-database management, taking into account the most important ones: transparency which is strongly connected with data virtualization, security which is fundamental to protect data, efficiency as a performance index, and, finally, interoperability to ease grid data service composition/interaction.

Transparency

Transparency is a common requirement for grid services and fundamental to make virtualization a reality. There are various possible types of transparency in a distributed environment. In particular, it relates to:

1. physical data location: the physical location of a database in the grid must be hidden/virtualized by the grid service;
2. naming: an application must be able to access a data source without knowing its name or location. These kind of information must be managed by means of mapping, alias, etc. which conceal data that are not relevant to the end-user, such as connection string for the databases, DBMS port, login and password, etc.;
3. data replication: replication of data improves performance, availability and fault tolerance. The user must not be aware of the existence/management of multiple physical copies of the same data source; she has just to deal with the logical (virtualized) data source name.
4. DBMSs heterogeneity: today many different RDBMSs exist, such as ORACLE, IBM/DB2, PostgreSQL, MySQL, SQLite, etc. Moreover, an increasing number of applications interact with not relational databases such as flat files and XML-based documents in the bioinformatics and climate change domains. This kind of heterogeneity (which includes different APIs, data types, physical support) must be properly handled in order to provide a uniform access interface to different data sources and a grid database access service independent of the back-end systems.

Efficiency

Performance plays a fundamental role in the data grid environment. High throughput, concurrent accesses, fault tolerance, reduced communication overhead, etc., are important goals that must be achieved by exploiting among the others data localization and query parallelism. Moreover, efficient data delivery mechanisms can reduce the connection time (parallel streams) and the amount of transferred data (data compression).

Security

Security is crucial for the management of a database in data grid environment. Data security aims at protecting data against unauthorized accesses by (i) preventing unauthorized users from accessing data and (ii) protecting information exchanged in the data grid network. Authentication is strongly required to check user's identity; authorization concerns privileges and read/write permissions. Most important production/research grids adopt the de-facto standard for security Globus Grid Security Infrastructure (GSI). It provides full security support concerning data encryption, data integrity, protection against replay attacks and detection of out of sequence packets. GSI is widely used both in gLite and Globus based grid environments.

Interoperability

Interoperability can be achieved by standard adoption. Today the adopted paradigm is basically service oriented; more specifically WS-I approach (which means based on SOAP, XML and WSDL W3C standards) is well suited for basic interoperability. OGF specifications issued by the DAIS-WG and discussed in the following section mainly focus grid database access aspects.

NEW STANDARDS AND SPECIFICATIONS

From a standardization point of view, in 2002 the Global Grid Forum (now Open Grid Forum) established a working group named DAIS (Data Access and Integration) to define a complete and effective set of specifications about these challenging topics. Since the beginning, the DAIS Working Group provides an umbrella under which many efforts are undertaken and people from all around the world are grouping together, giving a strong contribution in this area to the scientific grid community.
Interesting activities about Database Access and Integration recently produced the WS-DAI (Web Service Data Access and Integration) family of specifications (WS-DAI, WS-DAIR and WS-DAIX) (Antonioletti et al. 2006) which defines a set of web service interfaces to relational or XML data resources.
The base interfaces and properties for data access services are described in the Web Services Data Access and Integration (WS-DAI) specification. The WS-DAIR specification extends WS-DAI interfaces to allow access to and provide descriptions of relational data resources. Relational data resources are assumed to be composed of tabular data structures such as relations

and resultsets which are typically accessed either using SQL queries or by row iteration, respectively. The WS-DAIX specification extends WS-DAI interfaces to allow access to and provide descriptions of XML databases. XML data resources are assumed to consist of collections of XML documents that are accessed and modified using XPath, XQuery and/or XUpdate. Interfaces are provided for these languages in this specification.
The keyword highlighted by this standardization activity is interoperability, which can be achieved by different grid middleware providing reference implementations of these specifications.

MAIN PROJECTS

In the DataGrid area several projects addressed *grid database management*. In particular, the first three were: GRelC (which will be widely described in the next Section), Spitfire and OGSA-DAI.
The *Spitfire Project* was part of the Work Package 2 of the European Data Grid Project and provided a means to access relational databases from the grid. It was a very thin layer on top of an RDBMS (by default MySQL) that provides a JDBC driver. It used Web Service technology (Jakarta Tomcat) to provide SOAP-based RPC (through Apache Axis) to a few user-definable database operations.
The *Open Grid Services Architecture Data Access and Integration* (OGSA-DAI) (Karasavvas et al. 2005) is another project concerned with constructing middleware to assist with access and integration of data from separate data sources via the grid. It is engaged in identifying the requirements, designing solutions and delivering software that will meet this purpose. The project was conceived by the UK Database Task Force and is working closely with the Global Grid Forum DAIS-WG and the Globus team. OGSA-DAI provides a more complex framework with regard to Spitfire and it is currently used in several e-Science projects.

CASE STUDY: GRELC PROJECT

The Grid Relational Catalog (GRelC) is a research project started at the University of Salento (Italy) and addressing grid-database management issues. It basically aims at providing data grid solutions to access, manage and integrate data sources (e.g. relational databases).
Currently, the top service provided by the GRelC project is GRelC DAIS. It is a general purpose data grid service for database access and integration. This service acts as a standard front-end based on the well-known SOA approach. It provides both basic and advanced primitives to access, query, integrate, manage and interact with different data sources, providing a high level of transparency, concealing the back-end heterogeneity, middleware details related to Globus GSI and VOMS (Alfieri et al. 2003) and other low level issues. It is WS-I based, runs both on Globus and gLite grid middleware/environments and it provides efficient grid-enabled query mechanisms. From a security point of view, it supports both global (by means of VOMS) and local (on the GRelC DAIS side) authorization levels which means (i) fine and coarse grained data policies support and (ii) role-based management.
The GRelC DAIS provides a wide set of functionalities which includes: query submission, grid-database management related to user/VO/ACL, etc. advanced functionalities to transparently and securely integrate heterogeneous, distributed and geographically spread grid data sources (through P2P (Aloisio et al. 2007) connected GRelC DAIS nodes), etc. Moreover, it offers efficient data delivery (query resultsets) exploiting compression and streaming. Support for synchronous and asynchronous queries, is also provided.
The GRelC DAIS is able to transparently and securely integrate heterogeneous, distributed and geographically spread grid data sources, through a connected P2P-based network of GRelC DAIS nodes. The GRelC DAIS is very versatile so it can be used both at VO and site level. It can/is used in both ways depending on VO/user/database constraints and requirements. There is no single point of failure and no centralized management for this service due to the scalable P2P architecture.
Finally, by means of the GRelC Portal, GRelC DAIS nodes can be managed via web. The GRelC Portal eases the access and integration of grid-databases. It completely replaces the Command Line Interface, does not need additional configuration/installation of software and it provides a seamless and ubiquitous way to manage data sources in a grid environment.
Currently the GRelC DAIS is used as core service of the Euro-Mediterranean Centre for Climate Change (CMCC) (Fiore et al. 2008) grid metadata handling framework. Moreover, a wide deployment on the GILDA t-Infrastructure (Andronico et al. 2005) is also available for tutorials and training activities.

CONCLUSIONS

Data Grids represent the basic framework for next generation petascale grid environments. They provide a set of services to store, access, share, manage, distribute, synchronize and integrate massive amounts of data distributed across heterogeneous and geographically spread grid resources.
In this survey we presented the basic concepts related to grid-database access, management and integration, highlighting main requirements (with particular emphasis on transparency/virtualization) and issues.
We also described research activity in this area, in particular describing the GRelC project as case study.
Finally, we discussed the novel Open Grid Forum WS-DAI family of specifications as a key to address interoperability among grid database access services.

REFERENCES

Alfieri, R.; R. Cecchini; V. Ciaschini; L. dell'Agnello; A. Frohner; A. Gianoli; K. Lorentey; and Fabio Spataro. 2003. “VOMS, an Authorization System for Virtual

Organizations”. In *Proceedings of European Across Grids Conference*. 33-40.

Allcock, W.; J. Bresnahan; R. Kettimuthu; M. Link; C. Dumitrescu; I. Raicu; and I. Foster. 2005. “The Globus Striped GridFTP Framework and Server”. *ACM Press*.

Aloisio, G.; M. Cafaro; S. Fiore; and M. Mirto. 2004. “The GRelC Project: Towards GRID-DBMS”. In *Proceedings of the IASTED PDCN Conference* (Innsbruck, Austria, Feb. 17-19). IASTED, 1-6.

Aloisio, G.; M. Cafaro; S. Fiore; and M. Mirto. 2005. “The Grid Relational Catalog Project”. In *Advances in Parallel Computing - Grid Computing: The New Frontiers of High Performance Computing*. L. Grandinetti. Elsevier, 129-155.

Aloisio, G.; M. Cafaro; S. Fiore; M. Mirto; and S. Vadacca. “GRelC Data Gather Service: a Step Towards P2P Production Grids”. 2007. In *Proceedings of 22nd ACM Symposium on Applied Computing* (Seoul, Korea, Mar. 11-15). Vol. I, 561-565.

Alpdemir, M.N.; A. Mukherjee; N.W. Paton; P. Watson; A.A.A. Fernandes; A. Gounaris; and Jim Smith. 2003. “OGSA-DQP: A service-based distributed query processor for the Grid”. In *Proceedings of UK e-Science All Hands Meeting Nottingham* (EPSRC, Sep. 24).

Andronico, G.; V. Ardizzone; R. Barbera; R. Catania; A. Carriera; A. Falzone; E. Giorgio; G. La Rocca; S. Manforte; M. Pappalardo; G. Passaro; and G. Platania. 2005. “GILDA: The Grid INFN Virtual Laboratory for Dissemination Activities”. *TRIDENTCOM*, 304-305

Antonioletti, M.; M.P. Atkinson; R. Baxter; A. Borley; N.P. Chue Hong; B. Collins; N. Hardman; A. Hume; A. Knox; M. Jackson; A. Krause; S. Laws; J. Magowan; N.W. Paton; D. Pearson; T. Sugden; P. Watson; and M. Westhead. 2005. “The Design and Implementation of Grid Database Services in OGSA-DAI”. *Concurrency and Computation: Practice and Experience*, Vol. 17, Issue 2-4, 357-376.

Antonioletti, M.; A. Krause; N.W. Paton; A. Eisenberg; S. Laws; S. Malaika; J. Melton; and D. Pearson. 2006. “The WS-DAI Family of Specifications for Web Service Data Access and Integration”. *ACM SIGMOD Record*, Vol. 35, No. 1, 48-55.

Bell, W.H.; D. Bosio; and W. Hoschek. 2002. “Project spitfire - towards Grid Web Service databases”. *Database Access and Integration Services Working Group, 5th Global Grid Forum* (Edinburgh, UK).

Fiore, S.; Vadacca S.; Negro A.; and G. Aloisio. 2008. “Euro-Mediterranean Centre for Climate Change Data Grid”. Submitted to “DAPSYS 08 (Debrecen, Hungary, Sep 3-5)

Foster, I.; and C. Kesselman. 1997. “Globus: A Metacomputing Infrastructure Toolkit.” *International Journal on Supercomputer Applications* Vol. 11, No. 2, 115-128.

Foster, I. and C. Kesselman. 1998. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.

Foster, I. 2005. “Globus Toolkit Version 4: Software for Service-Oriented Systems”. In *Proceedings of IFIP International Conference on Network and Parallel Computing*. 2-13.

Karasavvas, K.; M. Antonioletti; M.P. Atkinson; N.P. Chue Hong; T. Sugden; A.C. Hume; M. Jackson; A. Krause; and C. Palansuriya. 2005. “Introduction to OGSA-DAI Services”. *Lecture Notes in Computer Science*, Vol. 3458, 1-12.

Pennington, R. 2002. “Terascale clusters and the teragrid”. In *Proceedings of High Performance Computing Asia* (Dec 16-19), 407–413.

Rajasekar, A.; M. Wan; R. Moore; W. Schroeder; G. Kremenek; A. Jagatheesan; C. Cowart; B. Zhu; S.Y. Chen; and R. Olschanowsky. 2003. “Storage Resource Broker - Managing Distributed Data in a Grid”. *Computer Society of India Journal, Special Issue on SAN*, Vol. 33, No. 4 (Oct), 42-54.

Tuecke, S. 2001. “Grid Security Infrastructure (GSI) Roadmap”. *Internet Draft*.

AUTHOR BIOGRAPHIES

SANDRO FIORE was born in Galatina (LE) in 1976. He received a summa cum laude Laurea degree in Computer Engineering from the University of Lecce (Italy) in 2001, as well as a PhD degree in Informatic Engineering on Innovative Materials and Technologies from the ISUFI-University of Lecce in 2004. Research activities focus on parallel and distributed computing, specifically on advanced grid data management. Since 2004, he is a member of the Center for Advanced Computational Technologies (CACT) of the University of Salento and technical staff member of the SPACI Consortium. Since 2001 he has been the Project Principal Investigator of the Grid Relational Catalog project. Dr. Fiore was involved in the EGEE project (Enabling Grids for E-science) and is currently involved in the EGEE-II project and other national projects (LIBI). Since June 2006, he leads the Data Grid group of the Euro-Mediterranean Centre for Climate Change (CMCC) in Lecce (Italy). He is author and co-author of more than 40 papers in refereed journals/proceedings on parallel and grid computing and holds a patent on advanced data management.
His e-mail address is sandro.fiore@unile.it and his web-page can be found at http://grelc.unile.it/ person.php?surname=Fiore

SALVATORE VADACCA was born in Galatina (LE) in 1982. He received summa cum laude bachelor and master degrees in Computer Engineering from the University of Lecce, Italy in 2003 and 2006, respectively. His research interests include data management; distributed, peer-to-peer and grid computing; as well as web design and development. Since 2003, he has been a team member of the GRelC Project. In 2006 he joined the Euro-Mediterranean Centre for Climate Change (CMCC) in Lecce, Italy, where he works in the Data Grid group. Since April 2008 he is a Ph.D. student in Interdisciplinary Science and Technology at ISUFI, Lecce.
His e-mail address is salvatore.vadacca@unile.it and his web-page can be found at http://grelc.unile.it/ person.php?surname=Vadacca

ALESSANDRO NEGRO was born in S. Pietro Vernotico (BR) in 1981. He received a summa cum laude bachelor degree in Computer Engineering from the University of Lecce, Italy in February 2004 and a summa cum laude master degree in Computer Engineering from the same University in April 2006. His research interests include GUI Development, Data

Management, Distributed, Grid Computing and Web Services. He is also interested in web and pattern-oriented design. Since 2003 he is a Team Member of the GRelC Project. Since September 2006 he holds a contract position in the LIBI project for the University of Lecce. Since April 2008 he is a Ph.D. student in Interdisciplinary Science and Technology at ISUFI, Lecce.
His e-mail address is alessandro.negro@unile.it and his web-page can be found at http://grelc.unile.it/person.php?surname=Negro

GIOVANNI ALOISIO is Full Professor of Information Processing Systems at the Engineering Faculty of the University of Lecce. His research interests are in the area of High Performance Computing, Distributed and Grid Computing and are carried out at the Department of Innovation Engineering of the University of Lecce. He is also the Director of the CACT (Center for Advanced Computational Technologies) of the National Nanotechnology Laboratory (NNL/CNR-INFM). As director of the CACT, which is an international partner of the US National Partnership for Advanced Computational Infrastructure (NPACI), he leads joint research projects on Grid both at a national and international level. He has been a co-founder of the European Grid Forum (Egrid) which then merged into the Global Grid Forum (GGF). He has founded SPACI (Southern Partnership for Advanced Computational Infrastructures), a consortium on ICT and grid computing among the University of Lecce, the University of Calabria and HP Italia. The Consortium is a follow-up of the SPACI project funded by the Italian Ministry of Education, University and Technological Research, to pursue excellence in the field of Computational Science and Engineering. He is also the Vice-Director of the NanoSciences and Grid Computing Section of the "Scuola Superiore ISUFI" and CEO of the SPACI Consortium. He is Director of the Division "ICT and Operations" of the Euro-Mediterranean Centre for Climate Change (CMCC). He is the author of more than 100 papers in refereed journals on parallel & grid computing.
His e-mail address is giovanni.aloisio@unile.it and his web-page can be found at http://grelc.unile.it/person.php?surname=Aloisio

REPRESENTING UNCERTAINTY IN SPATIAL DATABASES

Erlend Tøssebro
Dep. of Electrical Engineering and Computer Science
University of Stavanger
NO-4036 Stavanger, Norway
E-mail: erlend.tossebro@uis.no

Mads Nygård
Department of Computer and Information Science
The Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
E-mail: mads@idi.ntnu.no

ABSTRACT

Due to lack of accurate measurements, or rapid changes in time, spatial data are often uncertain. This paper presents a new abstract model for uncertain spatial information. The model is based on the principle that one knows that the uncertain object, regardless of type, must be within a certain area. The model also incorporates probability functions so that it is possible to determine the probabilities that various operations are true. This paper contains mathematical definitions of uncertain points, lines and regions. The paper also contains definitions of some relevant operations on these types. These operations are also evaluated for their usefulness with regard to uncertain data. A corresponding discrete model is already published.

1. INTRODUCTION

Databases which store information about geographic objects are becoming increasingly common in modern society as high-performance computer systems become available and positioning systems become more common and more accurate. However, many forms of spatial data cannot be measured exactly, or they may vary with time in such a manner that one cannot know exactly where the spatial object is at any given time. Examples of these two are given below:

Example 1: A lake is used as a reservoir for a hydroelectric power plant. Because of differences in energy demand and precipitation in the area, the water level, and thus the extent of the lake may vary considerably. Although one could store the exact size of the lake at any time by taking measurements frequently enough, this would be costly both in terms of manpower (taking the measurements) and space. A better solution might be to store the lake in a manner that indicates that it is uncertain. This uncertainty includes both position and the exact shape of the object.

Models for static uncertain regions exist already, and are well documented. See Section 2 for examples. However, other types of spatial data may also be uncertain. The following examples illustrate this for points and lines:

Example 2: If one is tracking a submarine, the sonars may give only an approximate position of the submarine, especially if the submarine is close to the sea floor and irregularities in the sea floor give off false readings.

Example 3: Simulations of the behaviour of an oil reservoir as well as other simulations relating to geological or geographical data may well yield results with some uncertainty. The model presented in this paper can be used to store such uncertain results.

Example 4: There are three different types of lines in geological databases assuming a two-dimensional view, and all of them may be uncertain because they are underground and therefore difficult to measure. The first type is a contact between two different rock types. This is really the boundary of two regions. The second is a fault line, either active or inactive. Because inactive fault lines are not necessarily tied to continental plates or to differences in rock type (there may be the same type on both sides), this is a true uncertain line.

The following example illustrate the need for a system that handles uncertainty in all the spatial data types.

Example 5: Imagine that you have scientists who are driving around making measurements in the Sahara desert to determine the extent of underground water reservoirs. The scientists themselves are uncertain points due to the imprecision of the positioning system that they use. The roads are uncertain lines because the roads in the Sahara desert are more like routes that shift as the sand dunes move than paved roads. The water reservoirs that the scientists are studying are uncertain regions because they are located deep underground and it is therefore not feasible to do more than a few measurements at each site. The scientists therefore lack the necessary information to define them precisely. Such a database would be useful for the scientists mentioned. If they could query such a database while on site using a wireless device, they could coordinate their efforts better.

This example shows that one may need to store uncertain data of all the three types in the same database. Most existing systems handles only one type or two types.

In (Duckham et al. 2001), an ontology of different kinds of uncertainty is defined. The hierarchy of forms of uncertainty, or imperfection, is shown in Figure 1.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

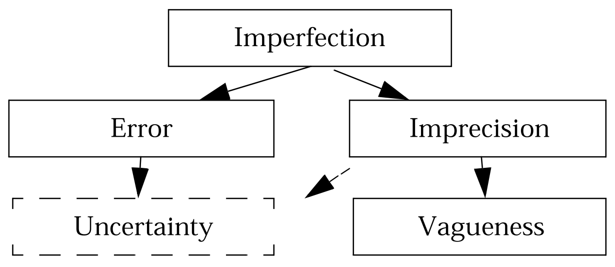


Figure 1: Hierarchy of the Types of Imperfection

Imperfection is considered to be the general form of uncertainty. Error is when measurements do not reflect reality. Imprecision is when measurements are lacking in specificity or are incomplete. (Duckham et al. 2001) considers vagueness to be a subcategory of imprecision. The basic goal of this paper is representing uncertainty in the position or extent of an object, regardless of the source of that uncertainty. In the rest of this paper, uncertainty therefore means either measurement error or imprecision due to incomplete knowledge, but does not cover vagueness. This definition of uncertainty is shown by the dashed box in Figure 1.

This paper will attempt to create a set of data types and operations for uncertain data, building on earlier work in spatial databases and models for vague and uncertain data.

In the rest of this paper, the word “crisp” will be used as the opposite of indeterminate (uncertain or vague).

2. RELATED WORK

There are several different types of models for spatial data. For spatiotemporal data, (Erwig et al. 1999) describes two modelling levels, abstract and discrete. Discrete models for spatiotemporal data can be directly implemented and are based on discrete representations such as vector or raster models. Abstract models are higher-level and usually model spatiotemporal data with point sets. In many abstract models, such as the one described in (Güting et al. 2000), lines and regions are modelled as infinite point sets in the Euclidean plane. This makes the model simpler, and may provide ideas for query operation designs that might be missed if one immediately went to the discrete level.

Abstract models also usually contain some rules to ensure that it is possible to store the data, although discrete models contain a lot more such rules. This paper contains an abstract model. Two distinct discrete models developed by the present authors have already been published in (Tøssebro and Nygård 2002b) and (Tøssebro and Nygård 2003). There is also a partial implementation of the model from (Tøssebro and Nygård 2003).

In (Tøssebro and Nygård 2002a), we outlined our current work on uncertainty in spatial and spatiotemporal database. A spatiotemporal extension to an abstract model like the one presented here is presented in (Tøssebro and Nygård 2002c).

One early model for uncertain points and lines is presented in (Dutton 1992). In this model, a point is represented as a central point with a circular deviation and a Gaussian distribution function over this area. A line is represented as a series of such points. The line segments between the points are represented by the union of the straight line segments going between all possible positions of the two points. The paper then shows that such a line will have the greatest variance in the points themselves, and the least variance is in the centre of the lines between the points. This is contrary to what one might expect. The uncertainty is usually smallest in the sample points and greater between them.

(Mark and Csillag 1989) describe a way to model uncertainty in the location of the boundary of a region that uses probabilistic error bands. This means that on each side of the estimated border there is an area with a certain width in which the border can be. Additionally, the probability that a point p is inside the area is a function of the distance from the estimated border to p.

The egg-yolk model described in (Gohn and Gotts 1996) and (Clementini and Di Felice 1996) models an uncertain region with only one face as two regions, one inside the other. The inner region is referred to as the ‘yolk’ and the outer region as the ‘white’ in the egg. This representation is then used to find a lot of different topological relations between uncertain regions, each consisting of only one component. A model for broad lines is presented in (Clementini 2005). A broad line in this model is a region that could result from the continuous deformation of a line as well as two broad points that represent its end points. A broad point is represented by the area that it might be in.

Models based on fuzzy sets have been frequently used to model vague regions. Fuzzy sets (Zadeh 1965) are sets in which the membership of any individual point in the set is not either yes or no, but rather a number between 0 and 1. Many of these models, such as the discrete models presented in (Lagacherie et al. 1996) and (Lowell 1994), use rasters to represent the fuzzy sets. The models described in (Schneider 1999) and (Erwig and Schneider 1997) represent another type of fuzzy set model, because these, like the abstract models for crisp objects, use infinite point sets. The most comprehensive model for vague data using fuzzy sets is the one presented in (Schneider 1999), which models all the standard spatial types (points, lines and regions) using fuzzy sets. (Schneider 1999) models a vague region as a fuzzy set where places which are certainly members of the region have values of 1 and regions which are only partially members have values between 0 and 1. A vague line is a line with a crisp position but uncertain membership. In Schneider's model, this uncertain membership is indicated by a function which gives values between 0 and 1 for each point in the crisp line.

Although these fuzzy models cannot be used as they are to model positional uncertainty, some of the ideas from them may be adopted.

(Wang and Hall 1996) describe a model for fuzzy boundaries between regions in which the fuzzy membership function indicates how sharp the boundary is. A membership of 1.0 indicates a crisp boundary.

(Cheng et al. 1997) describe several ways to extract fuzzy objects from observations. These methods use a combination of fuzzy sets and probability theory. The model that they use is a raster model because fuzzy membership values are stored in each cell. However, they also group the cells into objects according to different criteria.

Another possible model for uncertain regions, regardless of the type of uncertainty, is the vector-based discrete model presented in (Schneider 1996). This model bases itself on two boundaries, like the egg-yolk models.

(Worboys 1998) uses rough sets to define the outer and inner boundaries of possibly imprecise spatial objects. (Worboys 1998) defines resolution objects that are partitions on the underlying space, and shows how to convert objects from one resolution to another. This process may introduce imprecision even if the original representation was precise, because the object may only partially overlap one of the new partition parts. Another approach using rough sets is presented in (Beaubouef and Petry 2001). This paper essentially shows that rough sets can be used to create a more general version of the egg-yolk approach.

There has been an effort to create a comprehensive type system for different kinds of spatial databases. (Güting et al. 2000) describes such a type system for spatiotemporal databases. (Schneider 1999) describes a similar kind of model for vague spatial data.

3. BASIS FOR THE NEW MODEL

The new model presented in this paper takes ideas from several of the models described earlier. The model in (Dutton 1992) is adequate for modelling digitization error, but not adequate for some other applications. One example of this is Example 1 from the introduction. This example cannot be modelled by the one in (Dutton 1992). because the region may have an arbitrary shape. However, the concept that the point is known to be located within a region and has a certain probability distribution can be used in the new model. Another example is that the approach suggested in (Dutton 1992) cannot model uncertainty about the length of a line. As with points, the concept of a line with a probability distribution function is useful for our work.

(Schneider 1999) describes an abstract model for vague spatial data. The region model in that paper may be used as a basis for a model for uncertain regions. An uncertain region may be modelled as a probability function where points which are certainly members have a value of 1 and points for which membership is uncertain have values between 0 and 1. Schneider's model for a vague lines or vague points, however, is not so useful for uncertain data. An uncertain line typically has uncertainty about exactly where it is, which means that a different type of model must be used. However, an uncertain line may also have uncertainty about whether it exists or not. This existence uncertainty may be modelled in the same way as vagueness. The difference between vague and uncertain points is the same as for lines.

An important difference between our new model and Schneider's is that his model uses somewhat different mathematics. While Schneider uses fuzzy sets, our new model uses probability theory. This is both because uncertainty is best modelled by probabilities, and because the probabilities for uncertain points and lines must be modelled by probability density functions. The authors do not know of a similar concept in fuzzy set theory.

Some of the types from (Güting et al. 2000) are used as building blocks for the types presented here. Therefore, a brief description of these types is given now. The basic type for points is A_{points}, which is a finite set of points. The type for a single point is A_{point}. A line in (Güting et al. 2000) (of type A_{line}) is defined as a set of curves forming a graph. A curve is defined by a function from a variable t, which is between 0 and 1, to the X-Y plane. Curves cannot intersect with themselves. The carrier set of this type is called A_{curve}. A region (Güting et al. 2000) is an infinite set of points in the plane with the condition that there may be no singleton points or lines. That is, the region must be a valid result of a regularized set operation. A region may consist of a finite set of disjoint components, or faces. These again can have a finite number of holes. The carrier set of faces is called A_{face}, while the carrier set of regions is A_{region}.

4. DATA TYPES FOR UNCERTAIN SPATIAL INFORMATION

This section describes a set of data types for modelling uncertain spatial information. The first subsection will describe how to model the basic datatypes such as numbers. The other subsections will describe uncertain

Table 1: Carrier Sets for the Data Types from (Güting et al. 2000)

Individual	Set
A_{point}	A_{points}
A_{curve}	A_{line}
A_{face}	A_{region}

points, lines and regions. All the types will be defined by their carrier sets.

To define the data types that follow, the operation support is needed. This operation comes from fuzzy set theory, but has a slightly wider application here. In this paper, support is defined as follows for any function $f{:}z \to \Re$:

$$Support(f) \equiv \{z \mid f(z) > 0\}$$

The z in this formula is a member of whatever type or set of types the function f accepts as input values. This means that support is defined for all uncertain types, whether they are spatial or not. A more complete definition and discussion of this operation can be found in Section 5.3.

All the uncertain data types defined in this paper rely on probabilities or probability density functions. The properties of these are defined by the following functions:

- Probability Density:
  $ProbDens(P) \equiv (\forall x{:}P(x) \geq 0) \wedge \int_x P(x) \leq 1$
- Spatial Probability Density:
  $SProbDens(P) \equiv (\forall x \forall y{:}P(x, y) \geq 0)$
  $\wedge \int_x \int_y P(x, y) \leq 1$
- Probability Function:
  $ProbFunc(P) \equiv \forall x{:}(P(x) \geq 0 \wedge P(x) \leq 1)$
- Spatial Probability Function:
  $SProbFunc(P) \equiv \forall x \forall y{:}(P(x, y) \geq 0 \wedge P(x, y) \leq 1)$

4.1 Base Types

An uncertain number can easily be modelled by a probability distribution function. For a real number, this function would have to be defined as a probability density function, whereas for integers, it might be just a collection of probabilities for the number having particular values.

Definition 1: An *uncertain number* is defined as follows.

$$A_{UNumber} \equiv \{NP(x) \mid ProbDens(NP) \wedge ((PieceCont(NP) \wedge Support(NP) \in A_{Range(number)}) \vee (DiracDelta(NP) \wedge Support(NP) \in A_{number}))\}$$

PieceCont(F) is true if the function F is piecewise continuous. DiracDelta(F) is true if F is a dirac delta function.

Many queries in spatial databases return Boolean values for data without uncertainty. Because a single Boolean value cannot indicate uncertainty, different ways of answering these queries must be found. The most appropriate way to answer such queries for uncertain data is to give the probability that the answer is "True". However, there are some cases in which this probability is difficult to determine. An example of this is the "Cross" operation from Section 5.2. In such cases a third "Boolean" value, Maybe, is used to indicate uncertainty. This last approach was used in (Erwig and Schneider 1997).

These two forms of Boolean values are treated as two different types in this paper. The uncertain Boolean is the version with three values, and the other is called a probability. A third type which is useful for uncertain data is a type which indicates to which degree a statement is true. Some operations may return a degree of truthfulness which cannot be interpreted as a probability:

$$A_{UBool} \equiv \{False, Maybe, True\}$$

$$A_{Prob} \equiv [0, 1]$$

$$A_{Degree} \equiv [0, 1]$$

4.2 Uncertain Points

An uncertain point is a point for which an exact position is not known. However, one usually knows that the point is within a certain area. One may also know in which parts of this area the point is most likely to be. For instance in Example 2, one knows that the submarine is somewhere within the sonar reflection (a region) and by looking at the varying intensities of the reflection one might have an idea of where the submarine is most likely to be. An uncertain point is therefore defined as a probability density function P(x, y) on the plane. The support of this function is the area in which the point may be. To be able to store the function P(x, y) in a computer, it must be piecewise continuous. The probability that the uncertain point exists at all is the double integral of P(x, y) over the plane. To be able to model crisp points, P(x, y) must be allowed to be a dirac delta function.

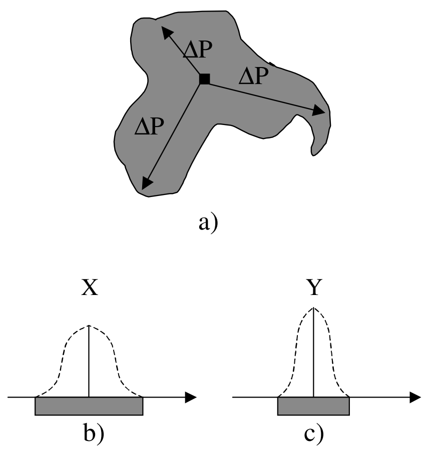


Figure 2: Uncertain Point

Definition 2: An *uncertain point* is defined as follows.

$$A_{UPoint} \equiv \{PP(x, y) | SProbDens(PP) \wedge ((Support(PP) \in A_{region} \wedge PieceCont(PP)) \vee (Support(PP) \in A_{point} \wedge DiracDelta(PP)))\}$$

A possible uncertain point is shown in Figure 2a. Figures 2b and 2c show views of the X and Y directions. The central spikes indicate the expected value of the points. The thick bar underneath indicates the area of uncertainty.

The model described here can model Example 2 because it allows the point to be inside an arbitrarily shaped region, and not just a circle like (Dutton 1992). It also enables a point to be modelled where its existence is not certain.

One problem with this model is how to determine the probability density function so that the double integral of it over the universe becomes 1 if the point is certain to exist.

Definition 3: The *uncertain points* set type is defined as follows.

$$A_{UPoints} \equiv \left\{ UP \subseteq A_{UPoint} | Finite(UP) \right\}$$

The Finite function returns True if the set contains a finite number of elements and False otherwise.

4.3 Uncertain Lines

The line type as defined in (Güting et al. 2000) is a set of curves where each member is a simple curve. The first step in developing a model for an uncertain line is therefore to create a model for an uncertain curve. An uncertain curve is a curve for which the exact shape, position or length is not known, but it is known in which area the

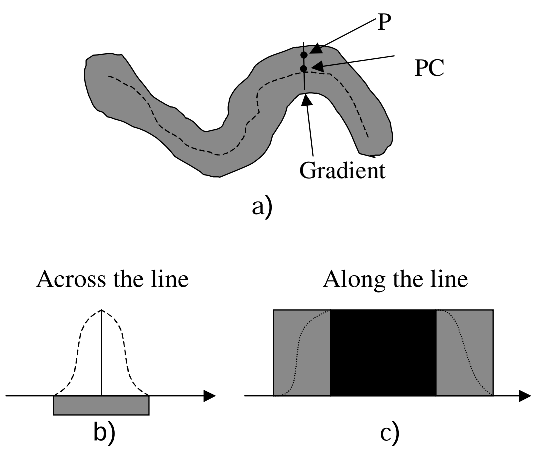


Figure 3: Uncertain Curve

curve must be. An example of an uncertain curve is shown in Figure 3a. It may also be known where in this area the curve is most likely to be. The dashed line in Figure 3a exemplifies this.

When seen along a line crossing it, a crisp curve would look like a point, or a set of points in the case of multiple crossings. When seen along the same line, an uncertain curve should be a probability density function indicating where the curve is most likely to cross. Such a function is shown in Figure 3b. This function may apply along the line marked “Gradient” in Figure 3a. Formally this line and its probability density function may be defined as follows:

$$A_{gradient} \equiv \{(gc, fg) | gc \in A_{curve} \wedge \forall (p \in gc):(fg:p \rightarrow \Re) \wedge ProbDens(fg)\}$$

When seen along its length, the uncertain curve has a probability of existing at each point. In Figure 3c, one common example of such a probability function is shown. In this example, there is uncertainty about the length of the line. This means that the line is certain to exist in the middle, and the probability of the line existing becomes lower the closer one comes to the ends.

One way of modelling this probability is that the uncertain line has a central line with a probability function associated with it. This probability function should not have areas in the middle where it is 0, because a curve with such a function is really two curves and not one, and should therefore be modelled as two curves. Such an illegal function is shown in Figure 4.

If there is uncertainty as to the number of curves, this may be modelled by a function which is less than 1 in a period between two places in which it is 1. This is shown in Figure 5.

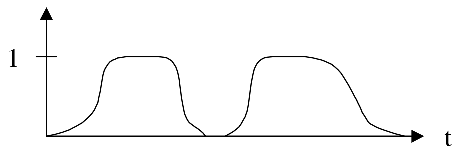


Figure 4: Illegal Probability Function for Uncertain Curve

This property of a function may be expressed mathematically as follows:

$$NoDip(f) \equiv (\forall x \forall y \forall z:((f(x) > 0) \wedge (f(z) > 0) \wedge x < y < z) \rightarrow f(y) > 0)$$

An uncertain line may be defined as a central line and a set of gradient lines. This set of gradient lines models the uncertainty in position and shape of the line. For each point of the central line there should be one and only one gradient line crossing it. The expected values of the probability functions of all the gradients should be somewhere on the central line. This is ensured by the following three conditions:

- The gradients do not share points or parts:

$$NoCross(G \subseteq A_{gradient}) \equiv (\forall (g_1, g_2 \in G):(g_1 \cap g_2 \neq \emptyset) \rightarrow (g_1 = g_2))$$

- For each point *p* on the curve, there is a gradient. The expected value of this gradient is *p*:

$$ExpectedCurve(ec \in A_{Curve}, G \subseteq A_{gradient}) \equiv (\forall (p \in ec) \exists (g \in G):E(g.fg) = p)$$

- The expected value of all the gradient lines are on the central line:

$$CurveExpected(ec \in A_{Curve}, G \subseteq A_{gradient}) \equiv (\forall (g \in G) \exists (p \in ec):E(g.fg) = p)$$

For all of these functions, E(x) is the expected value for a probability density function.

To ensure that the type is implementable, the probability density values of points that are close to one another should have similar values. To ensure this, we use the

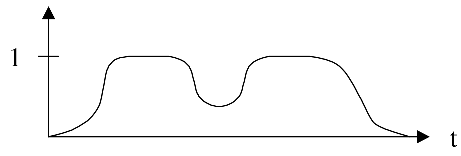


Figure 5: Probability Function Indicating Uncertainty about the Number of Curves

condition that all iso-lines of probability must be continuous. This means that for all possible probability density values, the set of points formed from the points along all the gradient lines that have this probability density should form either a continuous line along the central curve or a set of continuous cycles. The set of points from all the gradient lines that have a given probability density value is returned by the *ISet* function, which is defined as follows:

$$ISet(i > 0, G \subseteq A_{gradient}) \equiv \{x | \exists (g \in G):(x \in g.gc \wedge g.fg(x) = i)\}$$

To ensure that the iso-lines are continuous cycles, the following condition is used:

$$ContIso(ec \in A_{Curve}, G \subseteq A_{gradient}) \equiv (\forall i:((Iset(i, G) \subseteq Points(ec)) \vee \exists (C \subseteq A_{cycle}):Finite(C) \wedge points(C) = Iset(i, G)))$$

The function points(C) returns a set containing all the points which are parts of at least one cycle in the set of cycles.

To compute the area in which the uncertain line may be, one can take the union of all the gradient lines. The following condition ensures that the union of all the gradient lines forms a crisp face:

$$FormFace(G \subseteq A_{gradient}) \equiv (\{x | \exists (g \in G):x \in g.gc\} \in A_{Face})$$

Definition 4: An *uncertain curve* is defined as follows.

$$A_{UCurve} \equiv \{(ec, fe, G) | ec \in A_{curve} \wedge G \subseteq A_{gradient} \wedge \forall (p \in ec):(fe:p \rightarrow A_{Prob}) \wedge NoDip(fe) \wedge NoCross(G) \wedge ExpectedCurve(ec, G) \wedge CurveExpected(ec, G) \wedge ContIso(ec, G) \wedge FormFace(G)\}$$

This type definition is quite complex, and is the most complex type of the three main ones. The reason for this is that a point is a probability density function, a region is a probability function where each point has a probability of being in the region. A curve, however, is a little of both, as shown in Figure 3.

Note that this definition of a curve does not allow a curve that is partially crisp and partially uncertain. This is because there would be a point where the uncertain area ends and the crisp area begins where the probability density function of the gradients rises until it becomes infinite. In this place, some of the iso-lines would not be cycles as they will end right next to the point where the line becomes crisp.

This problem can be solved by defining such a curve as a line with several curves, some uncertain and some crisp. The crisp curve is defined by having all its gradient lines have length 0 and their probability functions being dirac delta functions.

The probability densities along gradient lines that are near to each other are dependent on each other in such a fashion that the line must be continuous. The extent of this dependence depends on the line in question, but all of the gradient lines must obey the following principle:

Let us say that along gradient line *A* the line passes through point *p*. For any line *B* in the neighbourhood of *A*, the following holds:

$$\left(\lim_{B \to A} E(B|(A = p)) = p\right) \wedge \left(\lim_{B \to A} V(B|(A = p)) = 0\right)$$

In this formula, *E*(*X*) is the expected value of *X* and *V*(*X*) is the variance of *X*.

Both points and regions are defined as functions over the plane. To make it simpler to define operations which are common to all uncertain spatial types, a view of the uncertain curve as a function over the plane is therefore also given:

Computational definition. An uncertain curve may be defined as a function over the plane:

$$C.f(x, y) = gl.fg(x, y) \cdot C.fc(cp)$$

In this function, gl is the member of C.G on which the point (x,y) lies and cp is the point at which gl crosses C.ec.

The line type is a set of curves for the same reasons as given for points.

Definition 5: The *uncertain line* is defined as a set of uncertain curves.

$$A_{ULine} \equiv \left\{ UC \subseteq A_{UCurve} | Finite(UC) \wedge \forall(ac \in UC)\forall(bc \in UC): (ac \neq bc \rightarrow \neg Cross(ac, bc)) \right\}$$

The requirement that two curves should not cross is there to ensure the uniqueness of the representation. If two curves that cross are added to the same set, they must be divided so that all four get the crossing as their end points. The Cross operator is defined in Section 5.2.

One problem with this model for lines is that it involves fairly complex mathematics, such as finding gradients of a function. Also, some operations, such as testing whether two lines cross each other, are much more complex in this model than in models for crisp or vague lines. This complexity exists because the uncertain curve is neither a simple probability density like for the uncertain point nor a simple probability function for each point like in an uncertain face.

4.4 Uncertain Regions

An uncertain region is a set of uncertain faces. An uncertain face is one where the location of the boundary or even the existence of the face itself is uncertain. This may be modelled as a probability function P(x,y) which gives the probability that the point (x,y) belongs to the face. Support(P) must be a valid crisp face. Additionally, an alpha-cut operation must yield a valid crisp region for all input values between 0 and 1. The alpha-cut function is defined as follows:

$$\alpha cut(f, i) = \{z | f(z) > i\}$$

A more complete definition may be found in Section 5.3. Note that the Support operation is the same as an alpha-cut with i=0.

Definition 6: An *uncertain face* is defined as follows.

$$A_{UFace} = \{FP(x, y) | SProbFunc(FP) \wedge Support(FP) \in A_{Face} \wedge \forall(i \in [0, 1]): \alpha cut(FP, i) \in A_{Region} \wedge PieceCont(FP)\}$$

This definition is used because it is very general, and gives the capability of modelling uncertain regions in which the exact number of faces is unknown. This is possible because the uncertain face can have a core which contains multiple crisp faces. It also allows holes which are not certain to exist (such as the submerged islands in Example 1) because there may be an area with a function value less than one inside an area with function value one. An uncertain face is known to exist if at least one point has probability one of being a member of the face

Figure 6a shows an example of an uncertain face where the black area is the area in which the face is certain to exist and the grey area is the area of uncertainty. Figures 6b and 6c show views of the probability distribution along the X and Y axis..

For regions, a dependence condition similar to the one described for gradient lines in Section 4.3 holds for the individual points that the uncertain face contains. If one

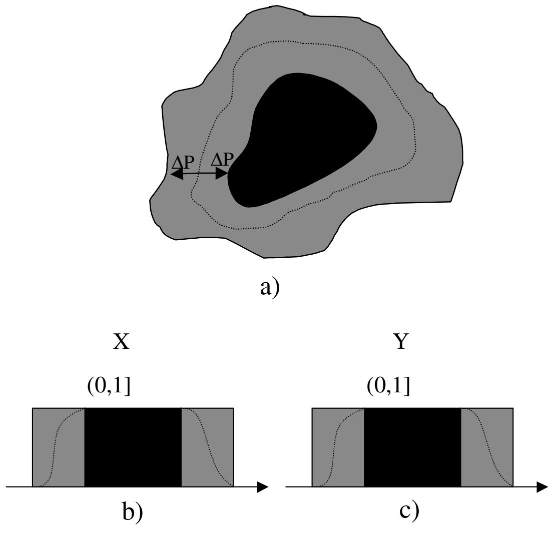


Figure 6: Uncertain Face

knows that point *p* is in the face *F*, the following condition holds for all points *q* in the neighbourhood of *p*:

$$\lim_{q \to p} P((q \in F) | (p \in F)) = 1$$

In this formula, *P*(*X*) is the probability that *X* is in the region.

A similar condition also holds if it is known that *p* is not in the face:

$$\lim_{q \to p} P((q \in F) | (p \notin F)) = 0$$

These condition only holds for the continuous parts of the probability function of the face and not across any discontinuities.

Definition 7: The *uncertain region* is defined as a set of uncertain faces.

$$A_{URegion} \equiv \left\{ UF \subseteq A_{UFace} | Finite(UF) \wedge \forall(af \in UF)\forall(bf \in UF):(af \neq bf \rightarrow Disjoint(af, bf)) \right\}$$

Disjoint for uncertain types is defined as follows:

$$Disjoint(A, B) \equiv Union(Support(A), Support(B)) = \varnothing$$

This type of model for faces and regions has the advantage that such faces and regions are well documented for the vague case using fuzzy sets in (Schneider 1999) and (Erwig and Schneider 1997). It is also very general, capable of modelling any kind of uncertainty. Error-band based models can only model uncertainty about the position and shape, not the number of components or holes such as in Example 1 above.

5. OPERATIONS ON UNCERTAIN DATA

An important part of a set of data types is a general definition of the operations that can be applied to them. Some of the operations from (Güting et al. 2000) as well as a few new ones are described here. For a more complete overview of operations for uncertain spatial data, see (Tøssebro 2002). The operations are divided into three categories, those that are applied to data with no uncertainty, but which cannot be determined with certainty for uncertain data, those that can be applied to both kinds of data, and new operations for uncertain data.

Table 2: Type Designators[a]

Letter	Type
Po	Point
C	Curve
F	Face
S	Spatial (Point, Curve or Face)
Ss	Spatial Set (Points, Line or Region)
N	Number
T	Any non-spatial or spatial type
B	Boolean
Pr	Probability
D	Degree
CX	Crisp X
CI	Crisp interval

a. All these stand for uncertain data types except for CX and CI

In this section, the letter name of the variable describes its type as given in Table 2. A signature of the type $S \times S \rightarrow S$ means that both the inputs must be of the same type, and the output is of the same type as the input. In a signature of the type $S \times F \rightarrow B$, S may denote a type other than *F* or *B*.

In the semantics for the operations, the letter *R* is used for the result, *I*1 for the first input and *I*2 for the second input.

The Core and Support operations from fuzzy set theory will be used for operations on uncertain data. These have slightly different semantics than in the vague case

Table 3: Operations for which a Positive Answer is Impossible for Uncertain Data

Operation	Signature	Semantics
Equal	$S \times S \to B$	Maybe: $(core(I1) \cap support(I2) = core(I1)) \wedge (core(I2) \cap support(I1) = core(I2))$ False otherwise
Touch	$F \times F \to B$	Maybe: $(core(I1) \cap core(I2) = \emptyset) \wedge (support(I1) \cap support(I2) \neq \emptyset)$ False otherwise

because of the differences between uncertainty and vagueness. Core is defined as follows.

- *Core($T \to CT$):* For a region, this operation returns the crisp set containing all the points or values having membership 1 in $I1$. For a complete definition, see Section 5.3.
- For *Support*, see Section 4 and Section 5.3.

5.1 Operations on Crisp Data which cannot be Determined with Certainty for Uncertain Data

The operations described in this subsection are listed in Table 3. They are much less useful for uncertain data because they cannot be determined with certainty. However, one may determine whether the operation is certainly false or not. The formula for determining this is given in the table.

Equal: One cannot determine equality between two uncertain objects. Even if the two objects have exactly the same type and probability function, they are not necessarily equal, because the real objects may be different even if the uncertain representations are "equal". For instance, if two regions both have the representation given in Figure 7, one of them can be bordered by line A and the other by line B. The regions produced by A and B are clearly not equal, but they can both correspond to the same uncertain region. The only way one can know that two uncertain objects are equal is if they are in fact the same object, with the same object identity or primary key value. The *Resemble* operation from Section 5.2 may be used to test approximate equality.

Touch: In the uncertain case this operation determines the possibility that two faces have a common border. Even if the supports overlap and the cores do not, one cannot be sure whether they actually have common borders or the borders just cross each other. Therefore, the operation cannot return "True" when there is uncertainty, unless the fact that the two faces have a common border is explicitly stored.

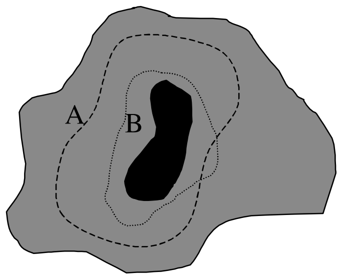


Figure 7: An Uncertain Region and two Possible "Real Regions"

5.2 Operations which may be used on both Crisp Data and Uncertain Data

The operations in this section are divided into five categories, depending on the types of their input and output: set operations, operations applicable to all uncertain spatial data types, operations for uncertain regions, operations for uncertain lines and projections. There are no operations that are only applicable to uncertain points and cannot also be applied to other types as well.

Set Operations
The set operations for the points and line data types are the same as in the crisp case. Using a set operation on the individual points and curves does not make sense. The point data type is not a set. Performing a set operation on a curve will most likely produce an illegal value. An uncertain region is in essence an infinite point set where the individual points have a certain probability of being

Table 4: Operations Applicable to All Uncertain Spatial Data

Operation	Signature	Semantics
Intersection	$Ss \times Ss \to Ss$	Points, Line: $I1 \cap I2$ Region: $R(x, y) = I1(x, y) \cdot I2(x, y)$
Resemble	$S \times S \to A_{Degree}$	$(area(min(I1, I2)))/(area(max(I1, I2)))$

members. Each set operation should therefore return a set that for each point gives the probability of the operation being true. This is easiest to do by combining the probability functions of the two input sets. Probability theory has been used to arrive at the formula given in Table 4. The events that a point belongs to regions $I1$ and $I2$ are considered to be independent.

The intersection operator returns a result of the lowest dimension of the two inputs. An intersection between a point and a line does not make sense in the uncertain case because the probability that the two are at exactly the same place is 0. An intersection between either a point or a line and a region uses the same semantics as the intersection of two regions. The output type is point or line.

Other Operations applicable to all Uncertain Spatial Data Types
Only one such operation is defined here. For a complete list, see (Tøssebro 2002). Its semantics is defined in Table 4.

Resemble: This operator determines how much two uncertain spatial objects resemble one another. It may be used to replace equal for uncertain objects. For crisp regions it is used to determine similarity in shape. The function min returns the minimum probability or probability density value of $I1$ and $I2$. Max returns the maximum.

Table 5: Operations for uncertain regions and lines

Operation	Signature	Semantics
Intersect	$S \times F \rightarrow A_{Prob}$	$Existence(I1 \cap I2)$
Cross	$C \times C \rightarrow B$	See text

Operations for Uncertain Regions
Intersect: This operator determines the probability that $I1$ and $I2$ intersect. This is the “overlap” criterion used in many spatial searches. The semantics of this operation is given in Table 5.

Operations on Uncertain Curves
Only the *Cross* operation is defined here. For a complete list, see (Tøssebro 2002). The semantics of the *Cross* operation is given in Table 5.

Cross: Determining whether or not two uncertain curves cross each other is far more complex than for crisp curves because one does not know quite where the curves are. If $support(I1) \cap support(I2) = \emptyset$, the two curves cannot cross. Otherwise they may cross. Computing the exact or even approximate probability that they cross is complex. In many cases a “Maybe” answer is sufficient. To know for sure, the following conditions must be checked:

- Both curves must exist in the entire area in which they may cross. The following formula test whether curve $I1$ exists in the entire area. The test is analogous for $I2$.

  $$\forall(g \in I1.G):(g \cap ca \neq \emptyset) \rightarrow (I1.fe(g \cap I1.ec) = 1)$$

  In this formula, $ca = support(I1) \cap support(I2)$.
- Let $ba = boundary(ca) \cap boundary(support(I1))$ and $bb = boundary(ca) \cap boundary(support(I2))$. Both ba and bb must consist of at least two disjoint crisp curves. This condition prevents line A in Figure 8 from getting “Yes” to the question $Crosses(A, L)$.
- Each component of ba and bb must cross line ec of the other line an odd number of times. This applies to both lines. This prevents lines such as line B in Figure 8 from getting “Yes” to the question $Crosses(A, L)$.

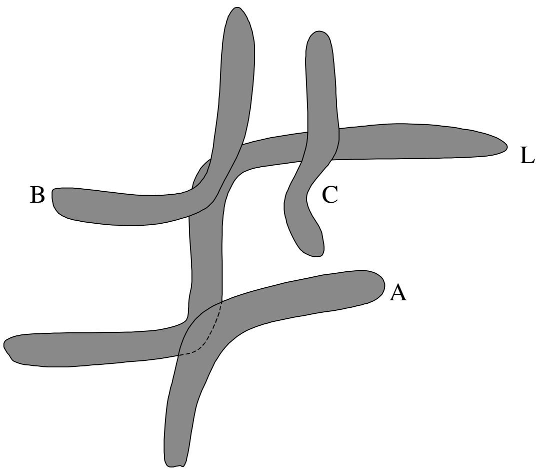


Figure 8: Two Curves which May Cross, and a Curve which Certainly Crosses, the Curve L

5.3 New Operations for Uncertain Data

These operations are new for uncertain data because they determine different aspects of that uncertainty. A list of these operations and their semantics is given in Table 6.

Alpha_Cut: This operation is described for fuzzy sets in (Zhan 1998). It returns a crisp set which contains all the points which have a membership value or probability density value above I2 in I1. The support operation can be seen as a special case of the alpha cut operation with I2=0.

Core: For regions, this operation returns the set of values which the object must contain. For lines it returns the central line. It is defined for lines because some other operations, such as Equals, need this definition. For points and numbers, the Core does not exist.

Support: This operation returns the set of values which the object might possibly contain or be at.

6. DISCUSSION

The type system described is uniform in that all the types are defined in roughly the same way. For all the types there is a "region" indicating where the object might be. In this sense all the uncertain types are based on the crisp region. For all the types there is also a probability function indicating where the object is most likely to be. This means that many of the same operations may be run on all three types with little alteration needed.

Because the model presented here models points, lines and regions in a uniform way, it is better suited to a database like Example 5 than many previous models. Our model for uncertain regions is similar to Schneider's model from (Schneider 1999), but in our work that method is applied for all the data types. The models from (Mark and Csillag 1989), (Gohn and Gotts 1996) and (Wang and Hall 1996) only handle regions while (Dutton 1992) only handles points and lines. The models in (Lagacherie et al. 1996), (Cheng et al. 1997) and (Worboys 1998) are essentially rasters and therefore are poor at representing lines. The types presented here are better integrated than a system consisting of a point model, a line model and a region model from different authors chosen because they are good models for the individual types. We are currently working on the issue of the completeness and computational closeness of this model. We have already managed to convert from line to region (*enclosed_by* operation) and from region to line (*border* operation). This work will be published elsewhere.

Additionally, some of the operations from (Güting et al. 2000) are evaluated for use in the uncertain case. Some of the operations cannot give a "certainly true" result for uncertain data, but many can be used for both crisp and uncertain data. Some new operations (and operations from other sources) are also introduced to deal with the uncertainty. For uncertain Boolean values, two methods are used. The simplest is a three-value logic, which has been used earlier. However, due to the definitions of the types, many functions may instead return the likelihood of the answer being true.

One potential problem with our model is that the mathematical complexity will make it a challenge to implement, and that simpler models might be better in certain cases. This is particularly true for the uncertain line model. The reason for the complexity of the uncertain line is that the probability distribution for an uncertain line is neither a probability density function like for points nor a probability distribution function like for regions, but something in between. This and other issues related to implementation of our model and similar models are discussed in (Tøssebro and Nygård 2002b).

Part of the mathematical complexity discussed in the previous paragraph may be easily removed from the model at the cost of a decrease in expressiveness. The probability functions may be arbitrarily simple or complex. The simplest variant is the function with equal probability over the entire point or line, and with probabilities 1, 0.5 and 0 indicating the core, uncertain boundary and the outside of a region. The advantage of a simple function is that it is easier to store and faster to compute.

The advantage of complex functions is increased expressiveness. In some cases, the geologists making the measurements may make good educated guesses as to the probability function. Thus it would be an advantage to be able to store these. The advantages and disadvantages of models of different complexity are discussed further in (Tøssebro and Nygård 2002b) and (Tøssebro and Nygård 2003).

Table 6: New operations for selected uncertain data types

Operation	Signature	Semantics
Alpha_Cut	$S \times [0, 1] \to \{CPo\}$ $N \times [0, 1] \to \{CN\}$	Spatial types: $R = \{P \in CPo \mid A(P) > B\}$ Number: $R = \{N \in CN \mid A(N) > B\}$
Core	$S \to \{CPo\}$ $N \to CN$	Point, Number: $\varnothing$ Line: $R = \{P \in CPo \mid P \in A.ec \wedge A.fc(P) = 1\}$ Region: $R = \{P \in CPo \mid A(P) = 1\}$
Support	$S \to \{CPo\}$ $N \to CI$	Spatial types: $R = \{P \in CPo \mid A(P) > 0\}$ Number: $R = \{N \in CN \mid A(N) > 0\}$

REFERENCES

T. Beaubouef and F. Petry. 2001. Vagueness in Spatial Data: Rough Set and Egg-Yolk Approaches. In *Proc. 14th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, 367-373

E. Clementini. 2005. A model for uncertain lines. In *Journal of Visual Languages and Computing*, 16, 271-288

E. Clementini and P. Di Felice. 1996. An Algebraic Model for Spatial Objects with Indeterminate Boundaries. In *Geographic Objects with Indeterminate Boundaries*, GISDATA series vol. 2, Taylor & Francis, 155-169.

A. G. Cohn and N. M. Gotts. 1996. The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries. In *Geographic Objects with Indeterminate Boundaries*, GISDATA series vol. 2, Taylor & Francis, 171-187.

T. Cheng, M. Molenaar, T. Bouloucos. 1997. Identification of Fuzzy Objects from Field Observation Data. In S. C. Hirtle and A. U. Frank (eds.) *Spatial Information Theory: A Theoretical Foundation for GIS*, LNCS vol. 1329, Springer-Verlag, 241-259.

M. Duckham, K. Mason, J. Stell, M. Worboys. 2001. A formal approach to imperfection in geographic information. In *Computers, Environment and Urban Systems*, 25, 89-103.

G. Dutton. 1992. Handling Positional Uncertainty in Spatial Databases. *SDH'92*, vol. 2, 460-469.

M. Erwig, R. H. Güting, M. Schneider, M. Vazirgiannis. 1999. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. In *GeoInformatica* 3, no. 3, 269-296.

M. Erwig and M. Schneider. 1997. Vague Regions. In *Proc. 5th Symp. on Advances in Spatial Databases (SSD)*, LNCS 1262, 298-320.

R. H. Güting, M. F. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, M. Vazirgiannis. 2000. A Foundation for Representing and Querying Moving Objects. In *ACM Transactions on Database Systems* 25, no. 1.

P. Lagacherie, P. Andrieux and R. Bouzigues. 1996. Fuzziness and Uncertainty of Soil Boundaries: From Reality to Coding in GIS: In *Geographic Objects with Indeterminate Boundaries*, GISDATA series vol. 2, Taylor & Francis, 155-169.

K. Lowell. 1994. An Uncertainty-Based Spatial Representation for Natural Resources Phenomena. *SDH'94* vol. 2, 933-944.

D. M. Mark and F. Csillag. 1989. The nature of boundaries of 'area-class' maps. In *Cartographica* 26, 65-77.

M. Schneider. 1996. Modelling Spatial Objects with Undeterminate Boundaries using the Realm/ROSE Approach. In *Geographic Objects with Indeterminate Boundaries*, GISDATA series vol. 2, Taylor & Francis, 155-169.

M. Schneider. 1999. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. In *Proc. 6th Int. Symp. on Advances in Spatial Databases (SSD)*, LNCS 1651, Springer Verlag, 330-351.

E. Tøssebro and M. Nygård. 2002a. Abstract and Discrete Models for Uncertain Spatiotemporal Data. In Proc. 14th Int. Conf. on Scientific and Statistical Databases (SSDBM), 240.

E. Tøssebro and M. Nygård. 2002b. An Advanced Discrete Model for Uncertain Spatial Data. In *Proc. 3rd Int. Conference on Web-Age Information Management (WAIM02)*, 37-51.

E. Tøssebro and M. Nygård. 2002c. Uncertainty in Spatiotemporal Databases. In *Proc. 2nd Biennial Int. Conference on Advances in Information Systems (ADVIS)*, 43-53.

E. Tøssebro. 2002. Representing Uncertainty in Spatial and Spatiotemporal Databases. Dr. Ing. Thesis at IDI, NTNU. IDI Report 2002:07.

E. Tøssebro and M. Nygård. 2003. A Medium Complexity Discrete Model for Uncertain Spatial Data. *To be published in Proc. 7th Int. Database Engineering and Applications Symposium (IDEAS),* 376-384.

F. Wang and G. B: Hall. 1996. Fuzzy representation of geographical boundaries in GIS. In *Int. Journal of Geographical Information Systems* 10, no. 5, 573-590.

M. F. Worboys. 1998. Imprecision in Finite Resolution Spatial Data. In *GeoInformatica* 2, no.3, 257-279.

L. A. Zadeh. 1965. Fuzzy sets. *Information and Control* 8, 338-353.

F. B. Zhan. 1998. Approximate analysis of binary topological relations between geographic regions with indeterminate boundaries. *Soft Computing* 2, 28-34.

Code Generation, Libraries and Programming Environments

OPENMP CODE GENERATION BASED ON A MODEL DRIVEN ENGINEERING APPROACH*

Julien Taillard
LIFL / INRIA
Email: Julien.Taillard@lifl.fr

Frédéric Guyomarc'h
University of Rennes / INRIA
Email: Frederic.Guyomarch@inria.fr

Jean-Luc Dekeyser
LIFL / INRIA
Email: Jean-Luc.Dekeyser@lifl.fr

KEYWORDS

MDE, HPC, Code generation, OpenMP

ABSTRACT

In this paper, we present a methodology which allows OpenMP code generation and makes the design of parallel applications easier. The methodology is based on the Model Driven Engineering (MDE) approach. Starting from UML models at a high abstraction level, OpenMP code is generated through several metamodels which have been defined. Results show that the produced code is competitive with optimized code.

INTRODUCTION

A physical barrier has been attained by processor suppliers which implies that frequency can not be increased like in the past. Previously, a basic processor replacement allowed to gain performance thanks to the higher frequency. Nowadays, the only way to reach performance is to carry out parallel computing and the multi-core processors development is making it even more important.

Unfortunately parallel computing is not easy for non-specialists. It requires knowledge of parallel algorithms and parallel coding for different kinds of architectures. From shared memory to distributed memory, the range of parallel machines is wide and program optimization is machine dependent. The complexity is even increasing now with the mix of all these concepts into multi-core machines joined into a grid.

Higher levels of parallel languages (like Fortress (2)) are made to simplify writing, but they are still for specialists. Methods to help the non-specialists to write parallel code must be defined.

The trend in software engineering is to model software at a high abstraction level in order to be more productive and to make the software durable. A high abstraction level enables to be independent from any language and the designers do not have to handle all implementation details. In the Model Driven Engineering (MDE) conception flow, input models (expressed at a very high level) are transformed into lower abstraction levels to finally generate code.

Our contribution is twofold. First, we propose a metamodel to model OpenMP programs that could be used in any model approach. This metamodel is based on a metamodel of procedural language wherein OpenMP concepts have been added. Secondly, we present a way to generate OpenMP programs starting from a high abstraction model using the OpenMP metamodel.

The paper is organized as follows. The next section presents some works about the automatic generation of parallel code. Then, the main concepts of the MDE are introduced. Later, an approach based on the MDE to produce OpenMP code is presented. Afterwards, some results about the produced code is analyzed. Finally, conclusions are given and some on-going works are presented.

RELATED WORK

The automatic parallelization is a widely studied domain. Lots of tools are available to handle it. Two approaches can be distinguished: the generation of parallel code and the parallelization during the compilation. Since our goal is to generate OpenMP code, we describe here only a few works closely related, which also deal with parallelization using OpenMP.

The classical approach is to generate OpenMP code starting from a sequential code. Tools such as CAPO (11; 10) and the POST project (1) are based on this approach. Starting from a sequential code, and through a data dependencies analysis, the tools will automatically identify the main characteristics of the code including the different types of loops. Then, there are two approaches: either the code is automatically generated with the insertion of the appropriate OpenMP directives or the tool indicates the user which loops can be parallel and helps him to decide.

Another approach is to generate OpenMP code from another parallel language. Thus, Krawezik et al. (13) propose to generate OpenMP code starting from MPI code. The resulting program is a program written in the same style as the MPI code: the *Single Program Multiple Data* (SPMD) style. Such approach is used because OpenMP is often better on shared memory architecture than MPI.

Our approach promotes the use of visual programming (the models), thus users don't need to handle code directly. As the models are made at a very high abstraction level, they can be reused for any other parallel languages. Code generation is made automatically without any user's intervention.

*This work has been partially supported by the CNRT FuturElec and the CNRS PEPS program

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

MODEL DRIVEN ENGINEERING (MDE)

Models are being used since a long time. Painting and sculpture can be considered as models of what they represent (8). They are used in computer science since time now. The first use of models in computer science is to make a system understandable for different developers. It is used as documentation and specification for the system developers. These kind of models is named *contemplatives*. In order to raise productivity, the idea is to make models *productive* and *executable*, and the MDE (18) has been introduced. It consists of using models at each level of conception, and to make models transformations between these levels. So that, once the system is modeled at a high abstraction level, only refinements are needed instead of rebuilding all the system at the new level. Such an approach allows to be independent of implementation details since they are managed by the transformations.

Model

A model is an abstraction of the reality. It is composed by concepts and relations in order to represent the system. When a designer models a system, he thinks about what he is interested in the system. Models are made with the designer's point of view. The same system could be modeled in different ways depending on the designer.

Models are usually made using the Unified Modeling Language (UML) (14) which is the standard from the Object Management Group (OMG) for visual programming.

In the HPC field, the designers are interested in the parallelism. Models of application and architecture need to express all the available parallelism to take advantage of the high number of processors available.

Metamodel

A metamodel defines the available concepts and relations that can be used to create a model. It could be compared to the grammar of a programming language. Designers are guided by the metamodel in order to produce same kind of models. A model made using a metamodel is said *conform* to this metamodel.

In our framework the metamodels must propose a mechanism to express all the parallelism in the models and also the mapping of software onto hardware architectures. The distribution of tasks on hardware is an important point to raise performances as different mappings of the same application can highly impact these performances.

Transformation

As the MDE promotes the use of models in each level of conception (abstraction level), an automatic generation of the different intermediate models is mandatory. A *model transformation* is performed between a source model and a target model, respectively conformed to the source and target metamodels. It could be seen as a compilation process. A transformation relies on a set of rules: a rule expresses how to transform a source concept (resp. relation) into an equivalent target concept (resp. relation). It facilitates the writing, extensions and maintainability since each rule is independent of others and can then be modified independently. The transformation of models are typically used to reach a low level model (the code) from a high abstraction level model. Each transformation is designed in order to add details to the models. It is a refinement used to be able to obtain the code with the appropriate implementation details. Several approaches of model transformations have been proposed. The Query/View/Transformation (QVT) (15) approach is an OMG standard for transformation. It proposes a declarative and an imperative language to write transformations. Unfortunately transformations tools are not yet mature and no complete implementation of QVT is available. SmartQVT (20) is a partial implementation of QVT based on the declarative language but it was not entirely developed when we started development. So, we developed our transformation tool called MoMoTE (Model to Model Transformation Engine). It is build as an Eclipse plugin using the EMF tools (7).

For the HPC area, the goal is to produce parallel code starting from a high abstraction level. Thus, a language independent model to express parallelism will be transformed into a language dependent model to express the parallelism. The high level expression of parallelism will be refined in order to produce optimized code in the targeted language, here OpenMP (C and Fortran).

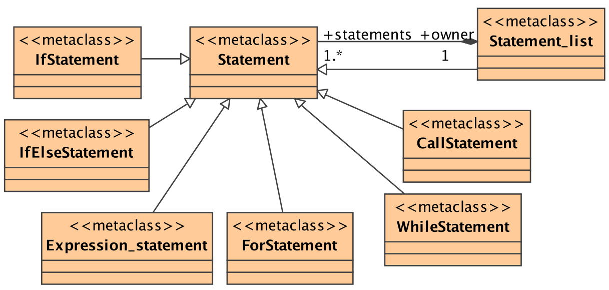
METAMODEL OF PROCEDURAL LANGUAGE USING OPENMP

As the targeted languages were Fortran and C, the most efficient strategy was to define a metamodel which allows to generate both languages. Since both are procedural languages, a metamodel of a general procedural language wherein the OpenMP concepts are added has been specified. The metamodel has been inspired by the C-ANSI Yacc grammar (12). An overview of the metamodel of procedural language is first presented and then the introduction of OpenMP statements in the procedural language metamodel is exhibited. Other procedural languages than C and Fortran have not been studied in detail to create the metamodel, but should be compatible with this metamodel with only minor changes.

Procedural language metamodel description

This metamodel aims to be of general-interest. It is not specific for our goal and could be used for any other purposes. It permits to declare programs, libraries, routines, functions, variables, expressions and all the available constructions. Pointers are not yet supported in the metamodel because we do not need it at the language level. In fact, in the higher level, all data are organized as multidimensional arrays which can be natively translated in languages we target.

The figure 1 presents an excerpt of the statements available in the procedural language. The most common expressions of languages can be modeled: the conditional statements (*if* and *ifelse*), the loop statements (*for* and *while*), the *call* of subroutines/functions and the construction of *expressions*.

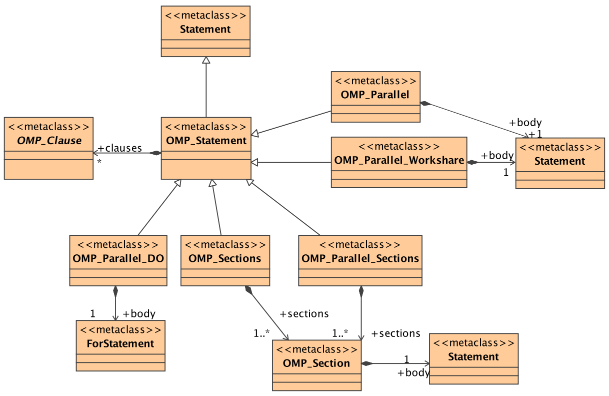


<<metaclass>> IfStatement
<<metaclass>> Statement
+statements
+owner
1..*
1
<<metaclass>> Statement_list
<<metaclass>> IfElseStatement
<<metaclass>> CallStatement
<<metaclass>> WhileStatement
<<metaclass>> Expression_statement
<<metaclass>> ForStatement

Figure 1: Statements in the procedural language metamodel

Adding OpenMP statement to procedural language metamodel

As the goal is to generate OpenMP code, OpenMP statements have to be available to generate OpenMP directives and function calls. OpenMP statements are added to the metamodel while OpenMP functions, like *omp_get_num_thread()*, are modeled in a library which is referenced when OpenMP procedural language model makes a function call.



<<metaclass>> Statement
<<metaclass>> OMP_Parallel
<<metaclass>> OMP_Clause
+clauses
*
<<metaclass>> OMP_Statement
<<metaclass>> OMP_Parallel_Workshare
+body
1
<<metaclass>> Statement
<<metaclass>> OMP_Parallel_DO
<<metaclass>> OMP_Sections
<<metaclass>> OMP_Parallel_Sections
<<metaclass>> ForStatement
+sections
1..*
<<metaclass>> OMP_Section
<<metaclass>> Statement

Figure 2: OpenMP statement added

The figure 2 illustrates the OpenMP statements added in the classical statement. An OpenMP statement can have *OMP_Clause* (such as *private* or *shared)*.

The figure 3 illustrates a model made with the metamodel (as a graphical representation is not generated with the metamodel). In this example, we illustrate the tree organization of a model: an OMP Parallel statement contains an ordered list of statements (S1,S2,S3) and two synchronization barriers (B). Each statement is also hierarchic: S2 contains an *if* statement.

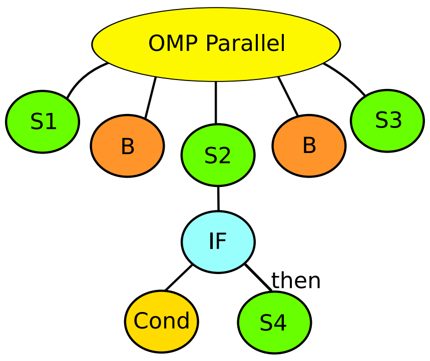


Figure 3: Tree representation of a model

OPENMP CODE GENERATION

The OpenMP code generation is implemented in a framework called Gaspard2 (21). It provides an Integrated Development Environment (IDE) for Multiprocessor System-On-Chip (MpSOC) co-modeling and for High Performance Computing application design (figure 4). Gaspard2 is able to manage both targets because the modeled systems are the same: MpSOC are massively parallel architectures and so are computers used for HPC, targeted applications are parallel in both cases and mapping of application on hardware architecture also needs to be expressed. Thus Gaspard2 has different targets (*Synchronous, SystemC, VHDL, OpenMP*) starting from the same high level model. The *Synchronous* target allows to do verification and validation of an application with the help of synchronous language. The *SystemC* target is able to make System On Chip co-simulation with SystemC. The *VHDL* one allows the generation of an hardware accelerator on a FPGA for a part of the application. The OpenMP target generates shared memory code executable on supercomputers. In the following, we focus on this later.

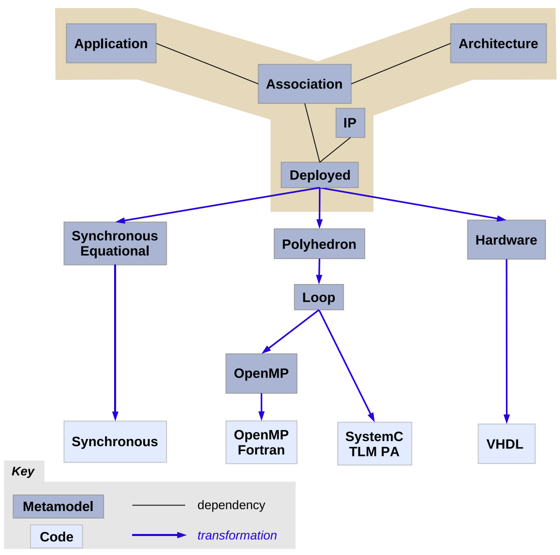


Figure 4: The Gaspard Y Chart

Gaspard2 is based on the component assembly in order to make components reusable. As a component, we define an elementary or composed system with fixed size input and output ports (the data). There is no state in a component, output values (on output ports) depend only on input values. The basic units of models are *Elementary Components* which are deployed to a function (usually from a library), they can be compared to a *black box* associated to a piece of code.

The global methodology, as seen in figure 4, is as follows: the first step is to design application and hardware architecture independently of each other. This separation of concepts allows to reuse these models and to keep them human readable. Then, with the help of an allocation mechanism, the computation tasks are distributed over processors and the data are mapped onto memories[1]. After that, a transformation chain (a succession of models transformations) leads to the generation of the desired code.

This MDE approach permits a great flexibility: targets can be added, reusing a part of the actual chains. Moreover, it encourages reuse of models and components. Once a component has been designed, it could be reused in several models. An application could also be reused to target several languages and hardware architectures. Thus, to target a new hardware architecture, users have to do the association between application model and hardware architecture. Code will be automatically generated for this architecture with the specified mapping. Similarly, several distributions of an application on an architecture could be tested easily. User just has to change the distribution specification, the new code , with the new distribution, will be automatically generated.

The integration of a new standard of OpenMP, such as the future version 3.0 (17), could also be easily done since user just needs to modify the last transformation generating the code itself, and also maybe between the loop and the OpenMP model (and add the new OpenMP features in the corresponding metamodel). In fact the transformations before these are mainly used to express the parallel loops. Concept will always be present in data-parallel languages.

OpenMP code is generated in the SPMD style. SPMD style is one of the most efficient styles compared to the classical loop level one (13) because there are less OpenMP directives since all the code is in a `PARALLEL` directive. As the mapping of application tasks on threads is explicit (using the number of threads), data locality can be assured whereas it cannot be with OpenMP directives which depends on the OpenMP compiler efficiency. In the code generation, a thread is considered to be associated to a processor in order to use the cache efficiently.

[1]for OpenMP, we do not place data onto different memories but for other targets of code like *SystemC*, we could simulate the effect of different placements

The high level model

The high level models are made with a profile (4) which is a subset of the UML Modeling and Analysis of Real-Time and Embedded systems (MARTE) profile (16; 19). A profile allows to add semantics to UML elements. This profile allows the modeling of software, hardware architecture and the association between them. It is a component based approach using UML 2 (14) components. Using a concept called *repetitive structure modeling*, the profile allows to express the repetitions of the same component instance (details could be found in (6)). This concept could be used in software as well as in hardware. It is based on the Array-Oriented Language (Array-OL) expression (5) which is a specification language to express all the parallelism available. Data-parallelism and task-parallelism could be modeled in the application model and available parallelism in the hardware architecture model is also expressed.

An application model can be seen as a hierarchical Directed Acyclic Graph (DAG) of tasks where nodes are tasks and edges are data dependencies between tasks. An application model is illustrated in figure 5. This is a simple application which consists of the initialization of two matrices, making the multiplication between both and checking the result obtained.

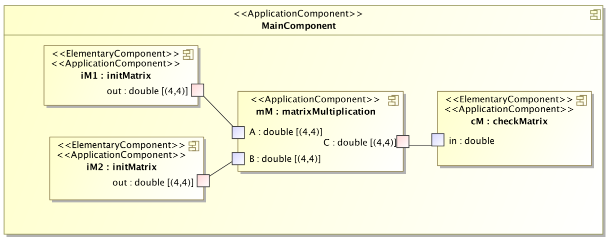


Figure 5: A simple application model

The task parallelism is expressed by several tasks without data dependencies between them. Then, if all the inputs data of the tasks have already been produced, all the tasks can be executed in parallel. Thus, in the figure 5, tasks *iM1* and *iM2* can be executed in parallel: there is no data dependency between them. A model which expresses data-parallelism is shown in figure 6. It represents a matrix multiplication, used in the application model of figure 5, using dot product algorithm. The *dP* instance is repeated (4,4) times to compute the (4,4) output matrix. There is no dependency between each repetition of the *dP* instance. This means the 16 computations can be done in parallel. The connectors stereotyped as *Tiler* express which part of the input/output arrays each repetition uses. Detailed information about *Tiler* can be found in (5).

Tasks distribution over processors is expressed using Array-OL expression. It expresses the distribution of the repetition of tasks over the repetition of processors (see details in (6)). Classical High Performance Fortran distributions such as `BLOCK` or `CYCLIC` distribution can easily be expressed.

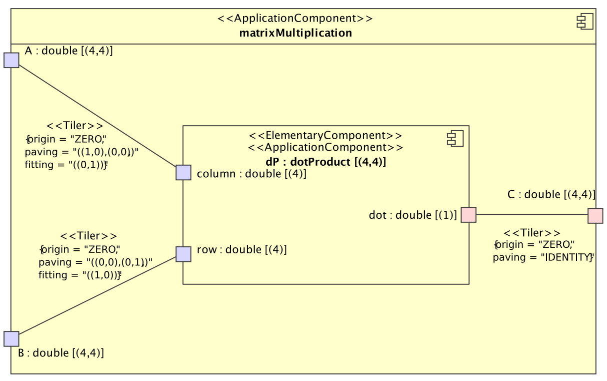


Figure 6: Matrix multiplication expressed with data-parallelism

Only "complete" models can be used for code generation. A "complete" model is an application model associated with an hardware architecture. To target OpenMP, each elementary component needs to be deployed on a function and each instance of elementary component has to be mapped onto a processor (directly or one component instance it belongs to using the hierarchy tree).

Brief explanation from the high level to loop model

The transformation starts from a "complete" high level model. Transformations are decomposed into small ones which are easier to develop, debug and can be reused for several targets. Both transformations briefly explained here are used for the OpenMP code generation as well as for the SystemC code generation.

The first step (high level model to polyhedron model) is to merge high level models into one model: the polyhedron one. During this transformation the expression of the tasks distribution over processors is transformed into the polyhedral model. This is a classical model in the data parallelism area. Considering that the repetition of a task can be seen as a multidimensional polyhedron, the mapping expression defines how to scan this polyhedron depending on the processor number (or the thread number in OpenMP). A polyhedron is generated for each mapping information and is given to the concerned tasks which are linked with the processor (or repetition of processors).

Once polyhedron model is obtained, the second step is to get closer to the implementation. Polyhedrons are transformed into loops using a tool called CLooG (3; Chunky Loop Generator) which generates loops scanning the polyhedrons. In the application part of the model, each polyhedron will be replaced by a nested loop scanning the polyhedron. The resulting application model is still a DAG where the mapping of tasks is expressed in loop expressions.

Generation of OpenMP procedural language model

This is the last transformation of the chain. The goal is to obtain a model in OpenMP procedural language metamodel starting from a loop model. Only the application is used until now, but hardware could be used to optimized code depending on the architecture.

Different tasks have to be done from this model to generate OpenMP code, they are:

- task scheduling
- generate OpenMP directives
- allocate variable
- determine shared variables
- put synchronization barrier

As we generate SPMD code style, all the generated code, which depends on thread number, is included in an OpenMP parallel section. The allocation and affectation of the thread number (p0) is also done automatically.Variables are private by default and shared variables have to be declared.

The transformation is made as follow: transformation rules first analyze the components at the highest hierarchical level, afterward each of these components is analyzed again to determine the sub-levels of component hierarchy.

The mapping between application tasks and processors expresses where each task or repetition of tasks will be executed but it does not express an execution order. Therefore a scheduling of the tasks of the graph is needed in order to produce a valid application. A scheduling is determined for each level of the hierarchy. The basic rule to determine the scheduling is: a task can be scheduled only if all its inputs have already been produced. Once a component can be scheduled, the transformation analyzes the sub-level of the component hierarchy.

The model in the OpenMP procedural language metamodel contains variables which have to be allocated. As we deal with shared memory architecture, data placement are not taken into account but data needs to be allocated and declared shared or private between threads. The basic way is to declare each port as variable, but this causes lot of variable allocation and worse, many useless memory copies penalizing performance. Ports which connect a sub-task to its containing task can be discarded: we just need to know where to read the data into the memory (they are sub-values of the superior port actually), instead of copying the data in a private variable. A variable is identified as shared variable and put on the `SHARED` clause declaration when it is used on tasks mapped on several threads. Variables names depend on the port name and a generated number is used to assure the unicity of variables.

When the transformation deals with an *elementary component*, it generates a call to the function on which the component is deployed. Tilers are computed or optimized in order to reduce the number of intermediary variables.

Insertion of synchronization barrier is needed in order to respect the data dependencies between parallel tasks.

The determination of synchronization barrier is done on the same principle. A synchronization is needed when tasks with data dependencies are not in the same thread (this could be determined with analysis of the loops). Once a hierarchical level has been mapped on a thread (as this is a sequential part), no synchronization are needed and variables used in this part should be private.

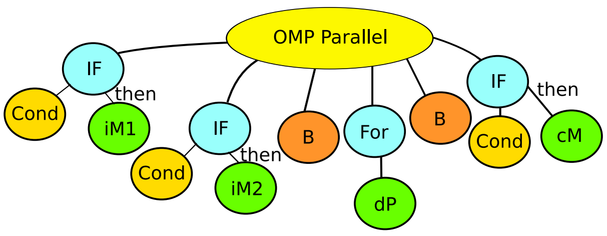


Figure 7: Tree representation of the OMP model

Based on a tree representation of the model, the generated model for the example given on figure 5 will look like in figure 7. It illustrates the OpenMP procedural language model generated for the application mapped on four processors. The mapping expressed that the *iM1* instance is placed on processor 0, the *iM2* instance is placed on processor 1 and the *cM* instance is placed on the processor 0. The distribution of the (4,4) repetitions of the *dP* instance expresses that each processor has to compute a column of computation.

We can observe that two synchronization barriers have been generated: one before the matrix multiplication (threads are waiting at the end of the initialization) and one after the matrix multiplication (waiting for all threads to finish the tasks they have to compute). The *if* statements are generated by the mapping of a single task on a single processor whereas the *for* statement is generated to distribute repetition of the (4,4) *dP* instance on the processor.

Code generation from OpenMP procedural language model

Since the model of lowest level is actually very close to the code itself (at least in structure), the code generation from the OpenMP procedural language model is nothing but a “pretty printer”; it translates the model representing the code into the code itself using templates.

The figure 8 illustrates the code generated from the model presented in figure 7. *Elementary components* are deployed on a routine corresponding to the component name (initMatrix routine for the initMatrix component).

RESULTS

In order to illustrate the use and the efficiency of such an approach, we have compared the execution time of different implementations of a classical program: the matrix multiplication (with (2000,2000) matrices). We have compared automatically produced code with the hand-written library GotoBLAS (9). Code was executed on a 3Ghz bi-Xeon dual core processor, running Linux with a total of 2Gb of shared memory. Parallel programs run over four threads.

```
program matrixMultiplication
  double precision , dimension(4,4) :: out1
  double precision , dimension(4,4) :: out2
  double precision , dimension(4,4) :: C
  integer :: p0 ! processor number
  integer :: x
  integer :: y

  !$omp parallel default(private)
  !$shared(out1,out2,C)
    p0 = omp_get_num_thread()
    !init matrices
    if (p0==0) then
      call initMatrix(out1)
    end if
    if (p0==1) then
      call initMatrix(out2)
    end if
    !$omp barrier
    do y=0,3
      x=p0
      call dotProduct(..)
    end do
    !$omp barrier

    if (p0==0) then
      call checkMatrix(C)
    end if
  !$omp end parallel

end program
```

Figure 8: Generated code example

Algorithm	Best execution time	Average
Row-column algorithm	0:21.11	0:21.60
Block multiplication	0:12.59	0:13.17
Block multiplication with GotoBLAS task	0:01.25	0:01.26
Parallel GotoBLAS	0:01.03	0:01.05
Sequential GotoBLAS	0:03.34	0:03.39

Figure 9: Square matrices - Execution times on four threads - Average is made on 100 executions

Three codes have been generated through Gaspard2: row-column multiplication, block multiplication and the block multiplication using GotoBLAS tasks on sequential parts. The two first generated algorithms go to the scalar operation (addition and multiplication) where as the last one use GotoBLAS function as soon as we are in a sequential part. Results are given in figure 9. We can observe that both codes which go to scalar operation are not competitive compared to the sequential and parallel hand written code. This is due to the fact that we are not able to optimize sequential code. The block multiplication using GotoBLAS task for sequential part is competitive with the parallel hand-written function. Results show that the best way to use Gaspard2 is to let Gaspard2 manage the parallelism using optimized sequential tasks.

CONCLUSION AND FURTHER WORK

In summary, we have presented the Gaspard2 framework that generates OpenMP code based on an MDE approach. All the transformations have been implemented as an Eclipse plugin. The results show that letting Gaspard2 manage the parallelism using optimized tasks on sequential part is the right way to use the framework. This method permits to generate code with performances closed to the optimized library. Programming at a high abstraction level allows to be language independent and to reuse models for different targeted language. This approach has also been tested on a conjugate gradient solving Maxwell equation and the acceleration is promising.

Several further works can be extracted from this approach. The first one is to optimize the generated OpenMP code to provide a better data locality. The use of private arrays instead of shared arrays should improve the use of cache and raise performance. As the multi-core processors are widely widespread, the second work which should be carried out is to optimized code for multi-core architecture which is not a classical SMP architecture since multi-core architectures share more resources as cache memory level. The last task is to target new hardware architectures such as the Graphical Processing Units which are a special kind of shared memory architecture, and the distributed memory architecture using MPI to manage communications.

REFERENCES

[1] Adhianto, L., Chapman, B., Lancaster, D., and Wolton, I. (2000). Tools for OpenMP Application Development: The POST Project. *Concurrency: Practice and Experience*, 12:1177–1191.

[2] Allen, E., Chase, D., Hallett, J., Luchangco, V., Maessn, J.-W., Ryu, S., Jr., G. L. S., and Tobin-Hochstadt, S. (2007). The Fortress Language Specification Version 1.0 Beta. Technical report, Sun Microsystems, Inc.

[3] Bastoul, C. (2004). Code generation in the polyhedral model is easier than you think. In *PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques*, pages 7–16, Juan-les-Pins.

[4] Ben Atitallah, R., Boulet, P., Cuccuru, A., Dekeyser, J.-L., Honoré, A., Labbani, O., Le Beux, S., Marquet, P., Piel, E., Taillard, J., and Yu, H. (2007). Gaspard2 uml profile documentation. Technical Report 0342, INRIA.

[5] Boulet, P. (2007). Array-OL revisited, multidimensional intensive signal processing specification. Research Report RR-6113, INRIA.

[6] Boulet, P., Marquet, P., Piel, E., and Taillard, J. (2007). Repetitive Allocation Modeling with MARTE. In *Forum on specification and design languages (FDL'07)*, Barcelona, Spain. Invited Paper.

[Chunky Loop Generator] Chunky Loop Generator. *CLooG home page*. `http://www.cloog.org`.

[7] eclipse.org (2005). Eclipse. `http://www.eclipse.org`.

[8] Favre, J.-M. (2005). Foundations of model (driven) (reverse) engineering : Models – episode i: Stories of the fidus papyrus and of the solarus. In Bezivin, J. and Heckel, R., editors, *Language Engineering for Model-Driven Software Development*, number 04101 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[9] Goto, K. and van de Geijn, R. (2002). On Reducing TLB Misses in Matrix Multiplication. Technical Report TR-2002-55, The University of Texas at Austin, Departement of Computer Sciences. FLAME Working Note #9.

[10] Ierotheou, C. S., Jin, H., Matthews, G., Johnson, S. P., and Hood, R. (2005). Generating OpenMP code using an interactive parallelization environment. *Parallel Computing*, 31(10-12):999–1012.

[11] Jin, H., Frumkin, M., and Yan, J. (2000). Automatic Generation of OpenMP Directives and Its Application to Computational Fluid Dynamics Codes. pages 440–456.

[12] Jutta Degener (1995). Ansi c yacc grammar. URL: `http://www.lysator.liu.se/c/ANSI-C-grammar-y.html`.

[13] Krawezik, G. and Capello, F. (2003). Performance comparison of MPI and three OpenMP programming styles on shared memory multiprocessors. In *SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 118–127, New York, NY, USA. ACM.

[14] Object Management Group, Inc., editor (2004). *UML 2 Superstructure (Available Specification)*. `http://www.omg.org/cgi-bin/doc?ptc/2004-10-02`.

[15] Object Management Group, Inc. (2005a). MOF Query / Views / Transformations. `http://www.omg.org/docs/ptc/05-11-01.pdf`. OMG paper.

[16] Object Management Group, Inc., editor (2005b). *UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) RFP*. `http://www.omg.org/cgi-bin/doc?realtime/2005-02-06`.

[17] OpenMP Architecture Review Board (2007). OpenMP Application Program Interface Draft 3.0.

[18] Planet MDE (2007). *Model Driven Engineering*. `http://planetmde.org/`.

[19] ProMarte partners (2007). UML Profile for MARTE, Beta 1. `http://www.omg.org/cgi-bin/doc?ptc/2007-08-04`.

[20] SmartQVT (2007). A QVT implementation.

[21] WEST Team LIFL, Lille, France (2005). Graphical Array Specification for Parallel and Distributed Computing (GASPARD-2). `http://www.lifl.fr/west/gaspard/`.

AUTHOR BIOGRAPHIES

JULIEN TAILLARD is a PhD student on computer science. His research interest are models and high performance computing.

FRÉDÉRIC GUYOMARC'H is assistant professor in University of Rennes and currently spends one year as researcher at the INRIA Lille - Nord Europe. He is interested in Linear Algebra and parallel algorithms.

JEAN-LUC DEKEYSER is professor in the Computer Science department at the University of Lille. He is also the scientific leader of the DaRT project at INRIA Lille - Nord Europe and the WEST team at LIFL (Laboratoire d'Informatique Fondamentale de Lille).

ANALYTICAL MATRIX INVERSION AND CODE GENERATION FOR LABELING FLOW NETWORK PROBLEMS

Michael Weitzel and Wolfgang Wiechert
Institute of Systems Engineering / Simulation Group
Mechanical / Electrical Engineering & Computer Science
University of Siegen, 57068 Siegen, Germany
E-mail: michael.weitzel@uni-siegen.de

KEYWORDS

Analytical Matrix Inversion, Sparse Systems of Linear Equations, Code Generation.

ABSTRACT

Symbolic simplification and algebraic differentiation are only some of the advantages symbolic computations offer over their numerical counterparts. With a focus on sparse systems of linear equations, this contribution presents an analytical approach to a matrix inversion problem occurring in the field of Metabolic Flux Analysis. It is illustrated how the inherent complexity of the approach can be handled and how symbolic solutions can be compiled into highly performant machine code. Finally, benchmark results demonstrate that compiled analytical solutions offer comparable or even better performance than state-of-the-art sparse matrix algorithms.

INTRODUCTION

The research context of this contribution is given by *Metabolic Flux Analysis* (MFA), a class of network flow problems found in Systems Biology. One especially successful approach to MFA is based on isotopic labeling where a specifically labeled substrate, e.g. ^{13}C labeled glucose, is fed to the cells of an (typically unicellular) organism. According to the intra-cellular reaction rates (*flux values*) the isotopic labeling distributes among the cell's metabolites (Wiechert 2001). In the following, only those details are introduced which are necessary to understand the underlying mathematical structures and the described algorithms.

The computational heart of MFA is the simulation of isotopic labeling enrichment by the solution of a cascaded system of linear equations (Wiechert and Wurzel 2001):

$$\begin{aligned} {}^0\mathbf{x} &= \mathbf{1} \\ \mathbf{0} &= {}^k\mathbf{A}(\mathbf{v})\,{}^k\mathbf{x} + {}^k\mathbf{b}\left(\mathbf{v}, {}^1\mathbf{x}, \ldots, {}^{k-1}\mathbf{x}, {}^k\mathbf{x}_{inp}\right) \qquad (1) \\ &\quad \text{for } k = 1, 2, \ldots \end{aligned}$$

These systems are obtained by serializing cascaded flow network graphs into balance equations. The unknowns ${}^k\mathbf{x}$ represent the labeling state of metabolites; the so called *cumulative isotopomer fractions*. The reaction rates $\mathbf{v}$ connecting the metabolites as well as the labeled input substrate ${}^k\mathbf{x}_{inp}$ are variables of the system. Moreover, vectors ${}^1\mathbf{x}, \ldots, {}^{k-1}\mathbf{x}$ serve as additional input for the system on level k and go into vector ${}^k\mathbf{b}$ in a nonlinear way. The described scheme is illustrated by the tiny example network in Fig. 1. The number of cascade levels corresponds to the maximum number of atoms (*labeling positions*) in a metabolite. The two connected flow network graphs are serialized into a cascaded system of balance equations. More details can be found in (Weitzel, Wiechert, and Nöh 2007).

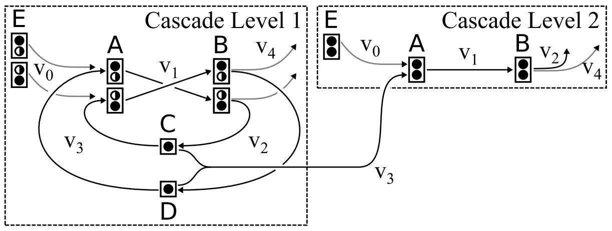


Figure 1: Cascaded flow network graphs

Cascade level 1:

$$ {}^1\mathbf{A}(\mathbf{v}) = \begin{pmatrix} -v_0 - v_3 & 0 & 0 & 0 & 0 & v_3 \\ 0 & -v_0 - v_3 & 0 & 0 & v_3 & 0 \\ 0 & v_1 & -v_1 & 0 & 0 & 0 \\ v_1 & 0 & 0 & -v_1 & 0 & 0 \\ 0 & 0 & v_2 & 0 & -v_2 & 0 \\ 0 & 0 & 0 & v_2 & 0 & -v_2 \end{pmatrix} $$

$$ {}^1\mathbf{x} = \begin{pmatrix} \text{A◐●} & \text{A●◐} & \text{B◐●} & \text{B●◐} & \text{C●} & \text{D●} \end{pmatrix}^T $$

$$ {}^1\mathbf{b}(\mathbf{v}, {}^1\mathbf{x}_{inp}) = \begin{pmatrix} v_0 \cdot (\text{E◐●}) & v_0 \cdot (\text{E●◐}) & 0 & 0 & 0 & 0 \end{pmatrix}^T $$

Cascade level 2:

$$ {}^2\mathbf{A}(\mathbf{v}) = \begin{pmatrix} -v_0 - v_3 & 0 \\ v_1 & -v_1 \end{pmatrix} \qquad {}^2\mathbf{x} = \begin{pmatrix} \text{A●●} & \text{B●●} \end{pmatrix}^T $$

$$ {}^2\mathbf{b}(\mathbf{v}, {}^1\mathbf{x}, {}^2\mathbf{x}_{inp}) = \begin{pmatrix} v_0 \cdot (\text{E●●}) + v_3 \cdot (\text{C●}) \cdot (\text{D●}) \\ 0 \end{pmatrix} $$

In practice, each level of these cascaded *compartmental systems* typically contains balance equations for a few dozens to several thousands of unknowns and possesses certain structural properties (Weitzel, Wiechert, and Nöh 2007):

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

1. Each edge of the flow network graph labeled with a flux value v_i subtracts v_i from the diagonal of ${}^k\mathbf{A}$ and adds the same v_i to an off-diagonal element. This property results in *weak diagonal dominance* of ${}^k\mathbf{A}$ since the diagonal elements sum-up a node's total influx with negative sign.

2. Metabolic networks are usually sparse graphs (Weitzel et al. 2007). The ratio of the number of edges to the number of nodes (i.e. the *connectivity*) ranges from 2 to 2.5. This means that the number of edges is in the order of the number of nodes. This property results in sparse matrices ${}^k\mathbf{A}$.

3. It can be shown that the connectivity of graphs decreases monotonously with increasing cascade level, thus matrices ${}^k\mathbf{A}$ are increasingly sparse.

4. Except for the diagonal, matrices ${}^k\mathbf{A}$ have the same non-zero pattern as the transposed adjacency matrix of the flow network graph. This property is invariant with respect to flux values $\mathbf{v}$ as long as $v_i \neq 0$.

5. The flow network graphs associated with ${}^k\mathbf{A}$ typically contain a wide range of isomorphic subgraphs (Weitzel et al. 2007).

6. Only a few variables $\mathbf{v}$ determine matrices ${}^k\mathbf{A}(\mathbf{v})$. Moreover, with increasing k, the variety of flux variables is cut back.

7. By using a certain permutation $\mathbf{P}_k$, matrices ${}^k\mathbf{A}$ can be permuted into a upper (lower) block-triangular form $\mathbf{P}_k^T({}^k\mathbf{A})\mathbf{P}_k$ where the blocks on the diagonal are compartmental matrices, again.

Strategies for Solving the Cascaded System

The straightforward approach for the solution of Eq. (1) would be to choose an efficient implementation of a standard LU algorithm, such as LAPACK's `DGETRF` / `DGESV` (Anderson et al. 1999). These standard implementations use partial pivoting in order to provide numerical robustness. For a reasonably large network, where $k = 1, \ldots, 11$ and $\max_k\{\dim({}^k\mathbf{A})\} \approx 1200$, this results in about 20 sec. for a single simulation run. However, by taking advantage of the special structure of Eq. (1), the running time can be reduced to about $20 \cdot 10^{-3}$ sec. for the same system (Weitzel, Wiechert, and Nöh 2007).

Diagonal dominance (property 1) can be used to speedup the classical LU algorithm by skipping the partial pivoting phase completely without risking numerical stability (Golub and van Loan 1996).

Sparsity of matrices ${}^k\mathbf{A}$ (property 2.) suggests to use a specialized sparse matrix algorithm. One possibly senseful way to use this property is to solve the *bandwidth-reduced* system $(\mathbf{Q}^T\mathbf{A}\mathbf{Q})(\mathbf{Q}^T\mathbf{x}) + \mathbf{Q}^T\mathbf{b} = \mathbf{0}$ by using a banded-LU algorithm. Property 4 ensures that the bandwidth-reducing permutation $\mathbf{Q}$ can be computed in advance, since ${}^k\mathbf{A}(\mathbf{v})$ does not change its non-zero pattern if $\mathbf{v}$ is varied. Because the algorithm for determining $\mathbf{Q}$ is a heuristic (George and Liu 1981) and significant bandwidth reduction is not always possible this reduces the computation time for the above example network from 20 sec. to about four sec.

The basis for more successful algorithms is the upper block-triangular form $\mathbf{P}_k^T({}^k\mathbf{A})\mathbf{P}_k$. In case only the diagonal blocks contain non-zero values each block corresponds to an independent subsystem. These subsystems do not influence each other and can be solved in any order. In the more general case, where diagonal blocks are *connected* by coefficients above the block diagonal, the subsystems given by the diagonal blocks are unilaterally dependent and thus, have to be solved in a specific order during a single back-substitution run (Davis 2006). The described algorithm greatly benefits from decreasing connectivity (property 3) which leads to a large number of small subsystems and highly sparse matrices ${}^k\mathbf{A}$. The use of this algorithm results in the above mentioned speedup of about factor 1000.

By property 5, subgraph isomorphism is an important feature of graphs ${}^kG$ and thus, classes of *equivalent* subsystems (equivalent except for permutation) can be found in ${}^k\mathbf{A}$. Even though all subsystems of such an equivalence class share the same LU decomposition, the beneficial use of subgraph isomorphism is difficult in practice and involves costly management which usually foils the additional theoretic speedup at least for smaller subsystems. Since, on the other hand, larger subsystems are sparse as well it might seem beneficial to switch to a banded-LU in order to solve them. In practice, the obtained speedup is minimal: the number of large subsystems is small and due to their origin, the subsystems are ill-suited for bandwidth reduction.

Property 6 is of no use for all direct numerical methods. This changes for the analytical solutions presented in the next section.

ANALYTICAL SOLUTIONS

The Idea behind Analytical Solutions

Although there exist fast solution algorithms for positive definite, banded, sparse, or other special systems the running time is always governed by the structure of the solution algorithm, but at most roughly by the structure of the underlying problem (matrix). In case of systems where the coefficients in $\mathbf{A}$ may change their value, but the sparsity pattern remains the same, e.g. during a parameter variation study, common algorithms suffer from their static setup and specialized algorithms obtained from analytical solutions might do a better job.

Even though sparsity and isomorphism give the clue that additional speedup might be within reach, the classical direct numerical sparse matrix algorithms do not accomplish to capitalize these properties exhaustively. Furthermore, the presorting and graph analysis steps found in these algorithms do not care about *how* a certain non-zero element is composed (i.e. the mathematical expression that lead to the non-zero value) since this information is usually out of reach and of no use for the algorithms found in conventional sparse matrix libraries.

The idea behind an analytical solution is simple. In general, analytical solutions can be obtained by carrying out an algorithm's operations symbolically: instead of executing arithmetic operators they are recorded symbolically in form of an expression tree. This works especially well if the underlying algorithm is branch-free, like the LU algorithm if partial pivoting is omitted.

Analytical solutions facilitate any type of *a posteriori* analysis. An interesting example is the ability to perform algebraic differentiation of Eq. (1) in order to provide highly accurate partial derivatives (*sensitivities*)

$$\frac{\partial \mathbf{x}(\mathbf{v})}{\partial \mathbf{v}} = -\frac{\partial}{\partial \mathbf{v}} \left({}^k\mathbf{A}(\mathbf{v})^{-1} \cdot {}^k\mathbf{b}(\mathbf{v}, \dots) \right)$$

used by various optimization algorithms during the parameter fitting step. Algebraic differentiation of explicit solutions is implemented as a set of simple expression tree transformation rules. Another example is the estimation of error propagation when a solution formula is actually evaluated using floating point arithmetic.

Analytical Matrix Inversion

Matrix inversion is usually not an option when solving a moderately sized system $\mathbf{Ax} + \mathbf{b} = \mathbf{0}$. Once a LU factorization of $\mathbf{A}$ is computed in $\Theta(n^3)$ FLOPS the solution of $\mathbf{Ax} + \mathbf{b} = \mathbf{0}$ requires additional $2n^2$ FLOPS (where $n = \dim \mathbf{x}$). In contrast, a matrix inversion based on a LU factorization requires $2n^3$ FLOPS, i.e. n times the effort of a solution. However, seen from an asymptotic standpoint, both direct numerical approaches require $\Theta(n^3)$ FLOPS (Golub and van Loan 1996).

Analytical computations require a change of perspective: there is no immediate relation between the time complexity of the algorithm *generating* the analytical solution and the number of FLOPS required to *evaluate* it; i.e. its size. While the underlying algorithm might not be able to handle sparsity efficiently, *structural* zeros in the input data cancel-out large parts of the analytical computations. In this context, it is helpful to adopt a classical idea from compiler design: high computational effort during the compilation phase and expensive optimizations are justified as long as the resulting code has improved efficiency and is executed frequently (Muchnick 1997).

For the original MFA problem, i.e. the solution of the cascaded system (1), the computation of an inverse $({}^k\mathbf{A})^{-1}$ becomes an option because only a small subset of the elements of $({}^k\mathbf{A})^{-1}$ is actually required to describe the MFA experiment:

- Measurements typically describe only a small subset of ${}^k\mathbf{x}$. A direct numerical approach for solving the cascade (1) cannot be trimmed to provide only the relevant subset of the solution vector ${}^k\mathbf{x}$. Clearly, having an analytical solution for $({}^k\mathbf{A})^{-1}$, this can be accomplished by extracting the relevant rows out of ${}^k\mathbf{x} = -({}^k\mathbf{A})^{-1}({}^k\mathbf{b})$.
- Vectors ${}^k\mathbf{b}$ are likewise sparse and a multiplication of type $({}^k\mathbf{A})^{-1}({}^k\mathbf{b})$ *selects* relatively few columns of $({}^k\mathbf{A})^{-1}$ for building the solutions ${}^k\mathbf{x}$.

Moreover, by the application of iterative improvement, an inverse $(\mathbf{A}(\mathbf{v}))^{-1}$ for flux vector $\mathbf{v}$ can be reused for some time as long as the parameter fitting procedure generates new flux vectors $\mathbf{w} = \mathbf{v} + \mathbf{d}$ by introducing slight changes $\mathbf{d}$:

$$\begin{aligned} \mathbf{x}_1 &\leftarrow -(\mathbf{A}(\mathbf{v}))^{-1}\mathbf{b}(\mathbf{w}) \\ \mathbf{x}_{i+1} &\leftarrow \mathbf{x}_i + (\mathbf{A}(\mathbf{v}))^{-1}(\mathbf{A}(\mathbf{w})\mathbf{x}_i - \mathbf{b}(\mathbf{w})). \end{aligned}$$

In practice the sequence $\mathbf{x}_1, \dots, \mathbf{x}_m \approx \mathbf{x}(\mathbf{w})$ converges to an acceptable result after few iterations. However, this is not applicable for larger $\mathbf{d}$. Furthermore, $m \ll n$ should hold, since each iteration requires $\Theta(n^2)$ FLOPS. See (Press et al. 2007) for details.

Other applications of the inverse include the implicit EULER iteration, where a non-stationary variant of Eq. (1) is considered.

Complexity of Analytical Solutions

Algebraically, the matrix inversion problem is closely related to the computation of the transitive, reflexive closure of a graph (Lehmann 1977). Each element of an inverse corresponds with a set of paths connecting a pair of nodes in the associated computational graph. Aiming to compute only a subset of ${}^k\mathbf{x}$ with the greatest possible efficiency this *path tracing* approach marked the starting point for our research. It was shown in (Isermann, Weitzel, and Wiechert 2004) how the approach can be applied to the solution of Eq. (1) using a generalized KLEENE algorithm. The resulting branch-free matrix inversion procedure is extremely simple and shall be used to illustrate the sketched scheme for building an analytical solution:

```
INVERSE $(\mathbf{E}_0 : \mathbb{R}^{n\times n}) : \mathbb{R}^{n\times n}$
1  $\mathbf{E} \leftarrow \mathbf{I}_n - \mathbf{E}_0$
2  for $k \leftarrow 1$ to $n$
3      do $\mathbf{E} \leftarrow \mathbf{E} + \frac{\mathbf{E}(:,k)\mathbf{E}(k,:)}{1 - \mathbf{E}(k,k)}$
4  return $\mathbf{E} \leftarrow \mathbf{I}_n + \mathbf{E}$
```

The following result can be proven by induction:

Proposition. *After executing k times line 3 the contents of $\mathbf{E}$ are*

$$\mathbf{E}_k = \begin{bmatrix} \mathbf{A}_k^{-1} - \mathbf{I}_k & -\mathbf{A}_k^{-1}\mathbf{B}_k \\ -\mathbf{C}_k\mathbf{A}_k^{-1} & \mathbf{I}_{n-k} - (\mathbf{D}_k - \mathbf{C}_k\mathbf{A}_k^{-1}\mathbf{B}_k) \end{bmatrix}$$

with the partition

$$\mathbf{E}_0 = \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{cc} k & n-k \\ \multicolumn{2}{c}{\begin{pmatrix} \mathbf{A}_k & \mathbf{B}_k \\ \mathbf{C}_k & \mathbf{D}_k \end{pmatrix}} \end{array}.$$

Moreover, after executing n times line 3, the contents of $\mathbf{E}$ *are* $\mathbf{A}_n^{-1} - \mathbf{I}_n = \mathbf{E}_0^{-1} - \mathbf{I}_n$. *Hence, the inverse* $\mathbf{E}_0^{-1}$ *is returned in line 4.*

Discarding the matrix notation, line 3 can be equivalently written as

```
3  $\mathbf{E}' \leftarrow \mathbf{E}$
3  for $i, j \leftarrow 1$ to $n$
3      do $\mathbf{E}(i,j) \leftarrow \mathbf{E}'(i,j)$
3              $+\mathbf{E}'(i,k)\mathbf{E}'(k,j)\frac{1}{1-\mathbf{E}'(k,k)}$.
```

The matrix operation in line 1, the three loops, and the addition of the identity matrix require $5n^3 + n^2 + n$ FLOPS in total, which is about two times slower than a matrix inversion based on the LU algorithm. Likewise, the required memory is in $\Theta(n^2)$. However, if *deferred evaluation* is used, i.e. the above code is used to obtain an expression tree for a symbolic rational expression, the memory requirements change dramatically.

The memory requirements $s(k)$ of a single entry of $\mathbf{E}_0$ are assumed to be in $\Theta(1)$. In step k of the outer loop the relaxation step in line 3 incorporates four times a value from the previous matrix $\mathbf{E}'$, introduces two $\Theta(1)$ nodes for the value "1" and another five $\Theta(1)$ nodes for operators. Hence, the recurrence $s(k) = \Theta(4s(k-1)+7)$ with $s(0) = \Theta(1)$ describes the memory consumption of the symbolic solution of a single element. The solution to this recurrence is

$$s(k) = \Theta\left(4^k + 7\sum_{i=0}^{k-1} 4^i\right) = \Theta((10\cdot 4^k - 7)/3). \quad (2)$$

For the complete matrix $\mathbf{E}_0^{-1}$ this results in expression trees having $\Theta(n^2 4^n)$ nodes in total (not counting the operations in lines 1 and 4). Clearly, this is intractable even for small problem sizes – for example, when assuming 12 bytes for a single expression tree node and $n = 10$ this results in 4000 megabytes for $\mathbf{E}_0^{-1}$.

Elimination of Common Subexpressions

At this point, analytical solutions for the matrix inversion problem seem to be completely impractical: even if the exponential memory requirements were no problem evaluation of the symbolic solution for each element of $\mathbf{E}_0^{-1}$ would require exponential time – just because the analytical solution has exponential size.

Fortunately, this problem can be handled by using the fact that the generated symbolic expressions share common subexpressions to great extend. A minimal example for $n = 2$ shall illustrate this. Just before the loop in line 2 starts the contents of $\mathbf{E}$ are

$$\mathbf{E} = \mathbf{I}_2 - \mathbf{E}_0 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

When the loop of line 2 exits the contents of $\mathbf{E}$ are

$$\mathbf{E} = \mathbf{E}_0^{-1} - \mathbf{I}_2 = \begin{pmatrix} e & f \\ g & h \end{pmatrix}.$$

For instance, the formula representing element h is

$$h = d + \frac{cb}{1-a} + \left(d + \frac{cb}{1-a}\right)\left(d + \frac{cb}{1-a}\right)S$$

using the term $S := 1/(1-(d+cb/(1-a)))$ which is a common subexpression of all solutions.

All expression trees for the elements of $\mathbf{E}_0^{-1}$ have the same size, namely 51 nodes (26 leaves, ten nodes for multiplication and five divisions, additions, and subtractions), thus 204 nodes in total. This conforms with (2), since $2^2 s(2) = 4 \cdot 51 = 204$. Collapsing all common subexpressions of the expression trees in $\mathbf{E}_0^{-1} - \mathbf{I}_2$ transforms the *forest* into a single directed acyclic graph (DAG) with only 33 nodes (5 leaves, 16 multiplications, two divisions and subtractions, and eight additions), i.e. only 16% of the nodes found in the forest.

A Divide & Conquer Matrix Inversion Algorithm

Different matrix inversion algorithms result in completely different analytical solutions. Although there is no immediate relation between size of the analytical solution and the running time of the underlying algorithm there is at least a tendency that a less efficient algorithm results in larger analytical solutions.

Another branch-free matrix inversion algorithm having its roots in the path tracing formalism is based on a recursive formulation of the matrix inversion problem (Conway 1971, Zhang 2005, Press et al. 2007):

$$\begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{T^{-1}} & -\mathbf{T^{-1}QS^{-1}} \\ -\mathbf{S^{-1}RT^{-1}} & \mathbf{S^{-1}(I + RT^{-1}QS^{-1})} \end{bmatrix}$$

with $\mathbf{T} \overset{\text{def}}{=} \mathbf{P} - \mathbf{QS^{-1}R}$ (the SCHUR complement of $\mathbf{S}$).

An efficient implementation of this formula results in a divide & conquer algorithm which requires two recursive calls for the inversion of $\mathbf{T}$ and $\mathbf{S}$ and inverts a matrix in about $2n^3 + 2n^2 - n/2$ FLOPS, which is slightly better than the LU algorithm without partial pivoting ($8n^3/3 - n^2/2 - n/6$).

More importantly, the DAGs containing the analytic solutions are about 20% smaller than those obtained from the KLEENE algorithm. Compared to the KLEENE algorithm an economical implementation of this algorithm is challenging and requires in-place matrix operations and careful handling of memory resources.

Creating DAGs and Handling Sparsity

Clearly, creating the analytical solutions in form of expression trees, first, and then collapsing them into DAGs in a second step would be an impractical strategy since this would require exponential memory. Instead, DAGs have to be created on-the-fly.

This is an especially elegant way because in each step of the matrix inversion algorithm a newly created operator node accesses only previously seen nodes. Using techniques like *common subexpression elimination*, *value* *numbering* (Muchnick 1997), as well as algebraic properties, like commutativity of operators, the DAG generation algorithm is able to decide whether an operator node needs to be created, or an existing node can be reused.

Certain applications of operators may lead to the value zero. In most of these cases the operator is a multiplication and one of the operands is a zero. This situation can be handled by replacing the operator node by a reference to the value zero. The remaining non-zero operand is kept in memory since it may be used by a later computation. Finally, when all elements of the inverse have been computed, the DAG is pruned of the unused nodes. This strategy efficiently handles sparsity and identifies matrix elements which contain a *structural zero*, i.e. a zero value which originates in the structure of the actual inversion problem.

Machine Code Generation

Once the DAG representing the inverse $\mathbf{A}^{-1}$ is in memory the goal is to evaluate it as fast as possible. Since the DAG is acyclic this can be established using a simple depth-first search starting at the individual root nodes and stopping either at the leaf nodes, or at nodes that have been visited before. Although this approach is far better than evaluating the original forest, there is some overhead for dereferencing pointers, switching the different operators and managing node labels.

This overhead can be eliminated by compiling the DAG into byte-code – either for an interpreter, or into machine code for direct execution on a CPU. Another option would be to emit a programming language source code, e.g. C code, which is compiled and optimized using an existing compiler. The drawback of this option is that common compilers are designed for small expressions and small basic blocks and compilation may take too long or require too much memory.

In the ongoing project the DAGs are compiled into efficient machine-code for the FPUs found in x86 family CPUs. These FPUs are organized as stack machines.

Clearly, the DAG has to be evaluated in a topological order starting at its leaves and stopping at the DAG's roots, i.e. result nodes representing the elements of the inverse matrix. Following (Bruno and Lassagne 1975), this special topological order, together with some management instructions, can be described as an *abstract stack machine* program. By design, this *program* forces an evaluation if either the stack load exceeds the maximum capacity or a root of the DAG (i.e. a result node) is reached. Furthermore, the abstract stack machine program cares about reusing previously computed intermediate results and saving them for later use.

Based on the information contained in the DAG and the abstract stack machine program x86 machine code is generated. Since complicated issues, like the allocation of the stack registers, and the correct handling of common subexpressions are already treated during the assembly of the abstract stack machine program a single pass is sufficient for generating the x86 FPU code.

In a second pass, a *peephole optimization* performs local optimization by sliding over the generated code and removing suboptimal constructs within a certain window. Finally, some prolog and epilog are generated, the assembler code is compiled into an object file and linked into a dynamic library. This library is dynamically loaded once at runtime and the matrix inversion code can be accessed via a function pointer.

RESULTS

In this section the numerical accuracy of the presented KLEENE algorithm and the divide & conquer algorithm are compared in order to illustrate the appropriateness for the cascaded systems in Eq. (1). Next, the performance of the compiled analytical solutions is compared to `LAPACK` (Anderson et al. 1999) and `CSparse` (Davis 2006), a state-of-the-art sparse matrix library.

Empirical Comparison of Numerical Accuracy

Figures 2 and 3 give empirical results for the numerical stability of the different matrix inversion algorithms. Each curve shows the mean, the minimum, and maximum of the residual $||\widehat{\mathbf{A}^{-1}}\mathbf{A} - \mathbf{I}||_\infty$ of 100 randomly generated matrices $\mathbf{A}$ and the numerically computed inverses $\widehat{\mathbf{A}^{-1}}$. These results demonstrate the growth of numerical errors for fully populated and sparse matrices when doubling the matrix dimension and are based on direct numerical implementations, i.e. do not use analytical solutions. Figure 2 shows the results for matrices fully pop-

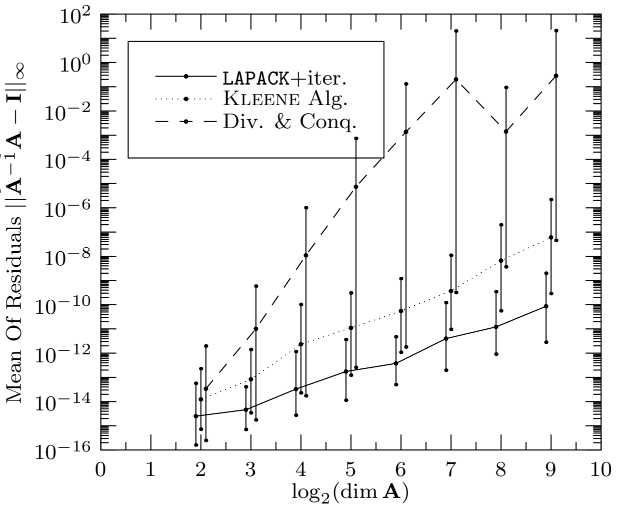


Figure 2: Numerical stability of different matrix inversion algorithms on fully populated random matrices. Shown are mean, min., and max. values of residuals.

ulated with $N(0, 1)$ random numbers. Matrices of this type are usually well-conditioned. The accuracy of the KLEENE algorithm is acceptable even for larger matrices. In contrast, the divide & conquer algorithm performs bad and the worst-case numerical accuracy is unacceptably low for larger matrices. The solid line corresponds to a matrix inversion based on a combination of the LAPACK routines `DGETRF` / `DGETRI` and an additional iterative improvement and is given for reference.

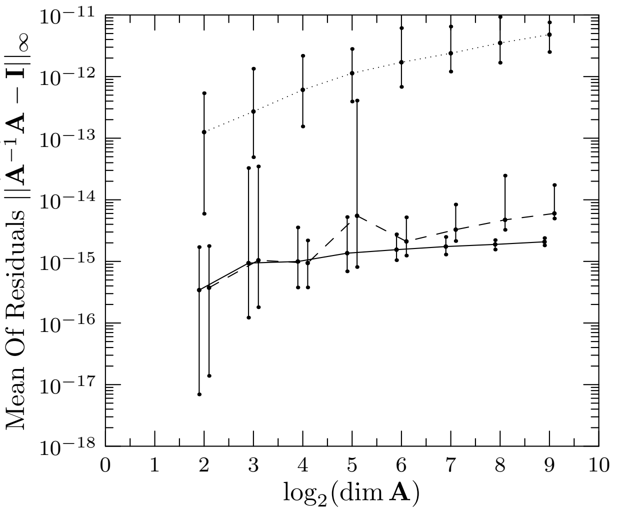


Figure 3: Numerical accuracy of the different matrix inversion algorithms on a set of 100 sparse random matrices obtained from randomly connected flow networks.

In Fig. 3 the same analysis was repeated for compartmental matrices obtained from randomly generated ERDÖS-REYNI flow network graphs (Bolobás 2001) assuming $\dim(\mathbf{A})$ in- and effluxes and $N(0,1)$ random flux values. The performance of all algorithms is very good. This time, amazingly, the divide & conquer algorithm returns the second best results.

Performance of Analytical Solutions

Figures 4, 5, and Fig. 6 compare the performance of the analytical solutions with LAPACK's `DGETRF` & `DGETRI` (Anderson et al. 1999) and the state-of-the-art sparse matrix library `CSparse` (Davis 2006). All benchmarks were prepared on a 2 GHz Pentium M machine with 2 GB of RAM. All analytical solutions are based on the presented divide & conquer algorithm.

Figure 4 shows the results for the set of fully populated random matrices. Because of the memory limitation during compile-time it was possible to compose analytical solutions for inverses of fully-populated matrices up to dimension 160.

Figure 5 gives the results for the set of random sparse matrices, again obtained from randomly connected flow networks. The maximum dimension achievable for sparse matrices ($\dim 2000$) depends on the degree of sparsity, which was adjusted to reflect a realistic network. The size of the machine code generated from sparse and fully populated matrices hitting the memory limit is comparable.

Finally, Fig. 6 shows the results for a realistic cascade obtained from a metabolic network representing the central metabolism of *E. Coli*. In contrast to the results pre-

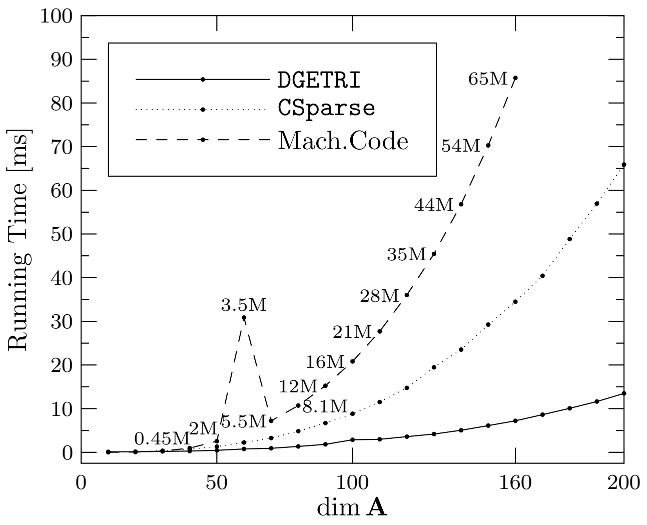


Figure 4: Running times of `LAPACK`, `CSparse`, and the generated machine code on fully populated matrices. Code sizes of the machine codes are given in megabytes.

sented in figures 4 and 5, the code generation algorithm worked on matrices ${}^k\mathbf{A}(\mathbf{v})$ containing symbolic expressions in its elements while `CSparse` worked on sparse matrices containing double precision floating point values.

CONCLUSIONS

Typical sparse matrix algorithms are based on the analysis of a system's non-zero pattern, or equivalently, the system's computational graph. The resulting techniques, like fill-in reducing permutations or system decomposition, are often based on elaborate graph-theoretical algorithms. Preconditioning and pivoting, on the other hand, analyze the matrix elements in order to provide numerical robustness. In contrast to the presented analytical matrix inversion, all these techniques may be characterized as *top-down* approaches.

Analytical solutions are constructed *bottom-up* using simple rules, which results in highly complex tailor-made solutions for a specific problem. Efficient handling of the inherent complexity and the choice of the underlying algorithm is crucial for the feasibility and the numerical stability of the approach. The divide & conquer algorithm qualifies in both aspects for the flow network problem.

Since the preparation of an analytical inverse may take from seconds to several minutes the effort has to be justified by the underlying problem (e.g. the use in a parameter variation study) and analytical computation is surely no option if an inverse is needed only once. While the generated code for the analytical inverse of fully populated matrices is slower than the LU algorithm used in LAPACK (Fig. 4), it is about three times faster than a state-of-the-art sparse matrix technique for randomly generated sparse matrices (Fig. 5) and up to five times faster for matrices obtained from a realistic network, where symbolic equations can be provided (Fig. 6).

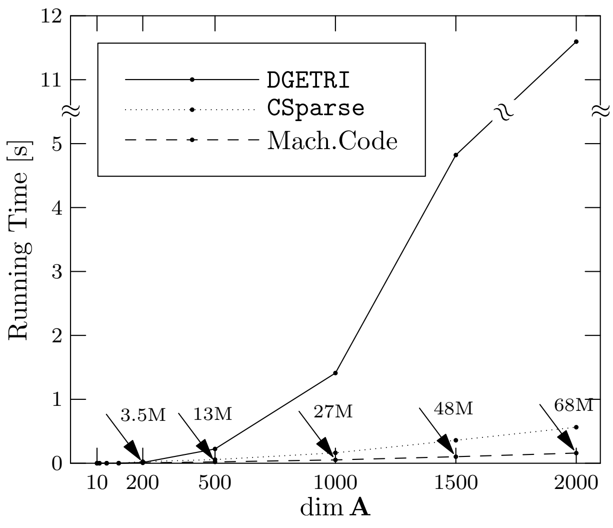


Figure 5: Running times of LAPACK, CSparse and the generated machine code on sparse matrices obtained from randomly connected flow networks with $N(0,1)$ randomly distributed flow values.

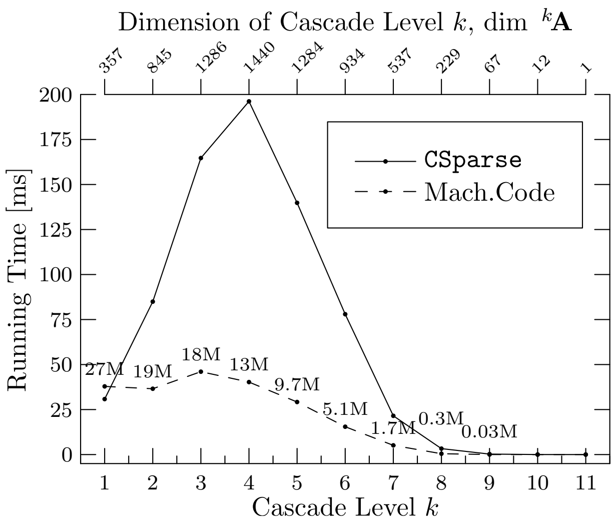


Figure 6: Running times on a realistic cascaded network representing the central metabolism of *E. Coli*. Sparsity increases monotonously with cascade level k and determines code size and running time.

FURTHER RESEARCH

The presented results are preliminary. Aiming to compute only a subset of ${}^k\mathbf{x}$ with the greatest possible efficiency the path tracing approach marked the starting point for our research. For this reason the variants of the classical Gauss algorithm, namely the branch-free variants of the LU and Gauss-Jordan algorithms, have not been ranked yet.

Analytical computations require a new way of thinking. There is no immediate connection between the performance of an analytical solution and the performance of the algorithm that was used to generate the solution. First results on algebraic differentiation show that the computation of derivatives is possible at little additional cost. This is due to the fact that derivatives can be build by reusing subexpressions of the original analytic solution.

Although the resulting machine code is rather small the preparation of the analytical solutions consumes much memory. First experiences with a garbage collection are promising and show that the compile-time memory requirements can be reduced drastically. Further research will focus on improving the quality of the generated code and other fields of application.

REFERENCES

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. 1999. *LAPACK Users' Guide*. Philadelphia: SIAM, third ed.

Bolobás, B. 2001. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2nd ed.

Bruno, J. L., and Lassagne, T. 1975. The generation of optimal code for stack machines. *J. ACM*, *22*(3), 382–396.

Conway, J. H. 1971. *Regular Algebra and Finite Machines*. Mathematics Series. Chapman and Hall.

Davis, T. A. 2006. *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. SIAM.

George, A., and Liu, J. W. H. 1981. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall Series in Computational Mathematics.

Golub, G. H., and van Loan, C. F. 1996. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press, third ed.

Isermann, N., Weitzel, M., and Wiechert, W. 2004. Kleene's theorem and the solution of metabolic carbon labeling systems. In *German Conference on Bioinformatics*, vol. 53 of *LNI*, (pp. 75–84). GI.

Lehmann, D. J. 1977. Algebraic structures for transitive closure. *Theoretical Computer Science*, *4*(1), 59–76.

Muchnick, S. 1997. *Advanced Compiler Design and Implementation*. Morgan Kaufmann.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 2007. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, third ed.

Weitzel, M., Wiechert, W., and Nöh, K. 2007. The topology of metabolic isotope labeling networks. *BMC Bioinf.*, *8*(315).

Wiechert, W. 2001. ^{13}C metabolite flux analysis. *Metababolic Engineering*, *3*, 195–206.

Wiechert, W., and Wurzel, M. 2001. Metabolic isotopomer labeling systems. Part I: Global dynamic behaviour. *Mathematical Biosciences*, *169*, 173–205.

Zhang, F. (Ed.) 2005. *The Schur Complement and Its Applications (Numerical Methods and Algorithms)*. Numerical Algorithms. Springer Science+Business Media Inc.

Workshop on Security and High Performance Computing Systems (SHPCS'08)

CORRELATION OF SYSTEM EVENTS: HIGH PERFORMANCE CLASSIFICATION OF SELINUX ACTIVITIES AND SCENARIOS

J. Rouzaud-Cornabas, P. Clemente, C. Toinard
Laboratoire d'Informatique Fondamentale d'Orléans
University of Orléans, Bâtiment IIIA, Rue Léonard de Vinci
45067 Orléans, France
Email: {jonathan.rouzaud-cornabas,patrice.clemente,christian.toinard}@univ-orleans.fr

KEYWORDS

SELinux sessions and scenarios, correlation, detection.

ABSTRACT

This paper presents an architecture for the characterization and the classification of activities occurring in a computer. These activities are considered from a system point of view, currently dealing with information coming from SELinux system logs.

Starting from system events, and following an incremental approach, this paper shows how to characterize high-level and macro activities occuring on the system and how to classify those activities. It gives the formal basics of the approach and presents our implementation. The results of experiments uses real samples taken from our honeypot. Correlation results are obtained using a *grid* computation. Our high performance architecture enables to compute a large amount of events captured during one year on a high interaction honeypot.

INTRODUCTION

The correlation of data generally aims at exploring heterogeneous information to highlight related data regarding particular aspects (e.g., chronology, semantics) among these different sources.

The core of the works presented here is the reconstruction of sessions composed of system activites, assuming that their elementary system operations (i.e., SELinux interactions or 'system calls') are all catched by the SELinux logs.

Following an incremental approach, our solution assembles system events together to represent high-level system activities (linux commands, execution of processes). We group them to represent macro activities including malicious ones, such as connections to hosts, island hopping between machines, massive activities, etc. The combination of these macro activities allow us to charaterize complete sessions on a SELinux system. Each macro activity can be classified. Complete sessions can also be reconstructed combining different types of macro-activities.

The paper gives the fundations of our approach. It defines events, high-level events, "meta-events", sessions, activities and gives the fundations for the algorithms that compute those data. The experimental results show the efficiency of the classification for SELinux activities. As far as we know, it is the first solution that enables to compute SELinux logs. It is interesting since SELinux provides a very secured kernel but produces large amout of events in the log file. Moreover, SELinux events are much more complicated than classical Unix traces.

STATE OF THE ART

The advantages of correlating information from multiple sources, are presented in (Valeur et al., 2004; Kruegel et al., 2005). Those authors propose a generic framework to correlate any kind of information coming from multiple network sources. But they only use data collected from networks and not at the operating system level. (Qin, 2005) combines several correlation algorithms. But again, the data come only from network tools and sensors.

(Chari and Cheng, 2003) studied system activities for specific system services. (Bowen et al., 2000) uses code analysis to find the authorized system calls. (Molina et al., 2007) uses strace to monitor system calls. All these approaches monitor partially the occuring system calls, so none of them is able to monitor all the system calls in order to reconstruct complete sessions. Moreover, complex sessions such as Island Hopping cannot be analyzed. In (Briffaut et al., 2006), scenarios are seen in terms of sequences of legal operations.

Finally, there is currently no solution at all supporting the reconstruction for SELinux sessions.

SOFTWARE ARCHITECTURE

The novelty of our approach deals with the reconstruction of system sessions. Sessions are sequences of system activites (i.e., SELinux, interactions, i.e. 'system calls', e.g. file/socket read/write). Those system activities enable to compute different types of macro activities called macro-events. Our architecture is designed to work with informations given by system loggers, host oriented sensors, network oriented sensors and also host IDS (HIDS) and network IDS (NIDS).

For the correlation process, each computer must have some tools to log all the events that are created on the computer or its local network (network connections).

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

The current implementation uses only system calls coming from the SELinux logs.

Adding Meta Knowledge to SELinux Events

In this first part, our correlation architecture analyzes each elementary operation reported by SELinux in order to highlight the similarity(ies) between them. Indeed, a lot of events are related to the same linux commands or process execution.

This first step adds meta-knowledges to raw events. That improves the effectiveness of the high-level classification modules.

Definitions

Event: Intuitively, an event is closed to the notion of elementary operation, or SELinux interaction or also 'system call'. Formally, a system event E is a set of attributes $ATTR$. Each attribute is associated with a label $l \in L$, $ATTR = \{attr_l; l \in L\}$

Table 1 gives a subset of L including the labels for the major attributes.

Label	***Description***
$class$	event/alarm classification
$scontext$	source security context
$tcontext$	target security context
sec_perms	security permission (e.g., r, w)
sec_class	security class (e.g., file, dir)
$hostname$	where the event appeared
pid	process number
$ppid$	parent process number
id	unique event number
$date$	date (in millisecond) of the event
$name$	name of the file involved
$inode$	inode number of the event
$priority$	priority level of the event/alarm
$idsession$	unique system session number
s_addr	source IP of the event
d_addr	target IP of the event
s_port	source $port$ of the event
d_port	target $port$ of the event
$count$	number of occurences of the event

Table 1: Event Attributes

Meta-event (*ME*): The notion of meta-event is also introduced. A meta-event ME is a set of attributes and/or collections of attributes. For example, the meta-event $ME = (\{scontext\}, tcontext, \{s\_addr\}, d\_addr, \{s\_port\}, d\_port)$ represents a meta-event with multiple source contexts, source IP and source ports (e.g. such as a meta-event that modelizes a DDoS attack).

Raw events (*RAW*): A RAW E^{raw} event is composed by the following attributes: id, $class$, $scontext$, $tcontext$, sec_perms, sec_class, $hostname$, pid, $ppid$, id, $date$, $name$, s_addr, d_addr, s_port, d_port.

Event filtering (*EF*): This module is a pre-processing for the correlation phase. If the events can not be normalized, they will not be used by the correlation processus, see figure 1.

Formally, an E^{ef} event has exactly the same attributes as a RAW event.

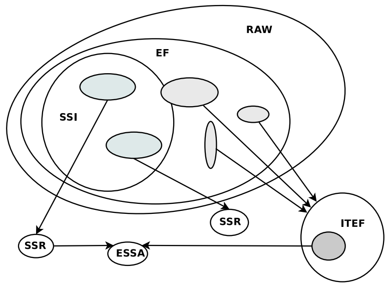


Figure 1: Overall Process of correlation

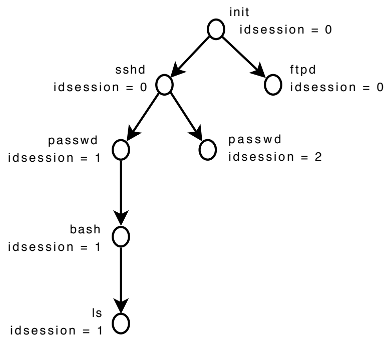


Figure 2: System Session Identification

For example, the EF module will exclude every events that have not been correctly generated by SELinux, i.e. missing or invalid attributes (e.g. no/incomplete date).

System Session Identification (*SSI*): SSI adds meta-knowledges to specific subsets of EF. It builds the tree of PID and PPID of each session.

Each new E^{ef} event of EF is added in real time to an existing tree that represents the (P)PID link or it creates a new tree where the event is added.

The main purpose of this module is to affect a unique attribute to each branch of the tree: the identification number of the session. Currently, only the events, that are associated to a user session, are taken into account. Ongoing works also allow to take into account sessions for services or system scripts.

Formally, a SSI E^{ssi} event is a EF event with an extra attribute $idsession$.

The initialization of the $idsession$ attribute uses an $entry\_point\_set$ that describes the interesting transitions between processes. For example, the transition from the 'sshd' context to the 'user' context enables to compute a new $idsession$ attribute.

Figure 2 shows that two branches below the '$sshd$' event have different $idsessions$.

InTeraction Event Factorizing (*ITEF*):

This module aggregates all the events corresponding to the same SELinux interaction (i.e., 'IT') inside a session. This module tackles the problem that an activity (execution of a single process) can produce a very large number of system calls. For example, the reading of a

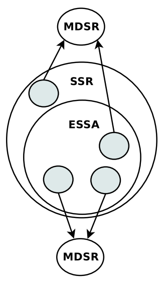


Figure 3: Migrating/Distributed Session Reconstruction

file can generate thousands of events for the same activity. So ITEF generates a single "meta-event' instead of thousands of events. This module is largely based on (Valeur et al., 2004; Kruegel et al., 2005; Qin, 2005).

An ITEF meta-event ME^{itef} is composed by single attributes: id, $class$, $scontext$, $tcontext$, sec_class, sec_perms, $hostname$, $dateBegin$, $dateEnd$ and $count$. Also, it has several sets of attributes for all included events: id, pid, $ppid$, $name$, s_port, d_port, s_addr, d_addr and optionally $idsession$.

For example, a "meta-event" factorizing all the read events of $/etc/rc.conf$ during the session 1142 on the host $selinux\_computer$ is in particular composed of: $scontext$ = $user\_u$: $user\_r$: $user\_t$, $tcontext$ = $system\_u$: $object\_r$: etc_t, $hostname$ = $selinux\_computer$, $idession$ = 1142, sec_class = $file$, sec_perms = $read$, $name$ = $rc.conf$

System Session Reconstruction (SSR)
As shown on the figure 1, the SSR module takes a subset of SSI events and builds a meta-event representing the session. The SSR meta-event ME^{ssr} contains the following attributes: id, $idsession$, $hostname$, $dateBegin$, $dateEnd$, $count$ and $class$.

For example, SSR merges all the events with the same identification session (e.g. idsession = '1051') and the same hostname (e.g. hostname = 'www-server'). It's worth mentioning that this module is an original proposition of this paper.

***Enhanced System Session Activities (*ESSA*)*: In contrast to (Valeur et al., 2004) (Kruegel et al., 2005), this module supports Inter Process Communication (like pipes, unix sockets). A ESSA meta-event ME^{essa} is composed by single: id, $idsession$, $hostname$, $dateBegin$, $dateEnd$, $count$ and $class$ and a set of attributes: $\{id\}$.

For example, ESSA produces a "meta-event" for a transmission between two processes through a Unix socket. Thus, a relationship is proposed between a $idsession$ and all the associated IPC events (using their id). A meta-event links a session SSR to all the events coming from the other sessions. The association between the sessions is achieved by the following module.

***Migrating/Distributed Session Reconstruction (*MDSR*)*: In contrast with (Valeur et al., 2004), MDSR authorizes multiple relationships. The MDSR meta-event ME^{mdsr}contains the following unique attributes: $dateBegin$, $dateEnd$, $exchange\_type$, $named\_socket$, and the following multiple attributes: $idsession_i$, $hostname_i$, s_addr_i, d_addr_i, $scontext_i$ and $tcontext_i$, with $i \in [1..n]$, where n equals the number of computers involved in the session.

Let a first information flow exist between session $s1$ and $s2$ and a second one between $s2$ and $s3$. Then MDSR produces a "meta-events" for the transitive information flow. The first information flow is associated to a ESSA "meta-event" $ME^{essa}_{s1,s2}$ between $s1$ and $s2$ events and the second information flow provides a ESSA meta-event $ME^{essa}_{s2,s3}$ between $s2$ and $s3$ events. The MDSR produces a first "meta-event" $ME^{mdsr}_{ME^{essa}_{s1,s2}}$ relating $s1$ and $s2$ and a second one $ME^{mdsr}_{ME^{essa}_{s2,s3}}$ relating $s2$ and $s3$. Finally, MDSR produces a meta-event $ME^{mdsr}_{ME^{mdsr}_{ME^{essa}_{s1,s2}},ME^{mdsr}_{ME^{essa}_{s2,s3}}}$ for the transitive flow relating $s1$ and $s3$.

Meta Events Recognition

***System Pattern Recognition (*SPR*)*: This module allows the correlation processus to classify SSR and ESSA.

Prerequisites and consequences (Ning et al., 2001) (Cuppens and Miège, 2002) do not feet for automatic learning of sessions. Prerequisites and consequences must be written by end users. It is a difficult task. Our approach is totally different. SPR creates an automaton that represents a session in order to compare it with System Patterns (SP) also represented as automata. Thoses SP are learnt by different processus (see the System Pattern Learning section). SPR generates a SPR meta-event that classifies the session with those patterns.

SPR generate a SPR "meta-event" with the following attributes: $idsession$, $hostname$, $id\_pattern$ and $level$.

For example, as seen on figure 4, SPR allows to compare a session represented as an automaton (on the left side) with a system pattern learnt before (on the right side). They are both real world sessions and patterns. More precisely, the session on the left side represents a SSH connection followed by: (1) the execution of a shell, (2) the transition from the SSHD context to the user one, (3) the opening of a virtual console, (4) some interactions, i.e. read and write, on this console. The pattern (on the right side) represents: (1) the detection of a connection through SSH (transition from SSHD to user context), (2) then the opening of a virtual console.

To allow a better recognition, this module compares the two automata using a similarity level (i.e. the number of event attributes that have to be equal between the two compared nodes).

For example, as seen on the figure 4, the lowest similarity level corresponds to the comparison between the session and the pattern session using only two predefined security context values. Different level of classification can be used with the same pattern. At the similarity level 4, the sub-session (in the rectangle) on the left cannot be recognized.

In terms of complexity, the SPR module compares each

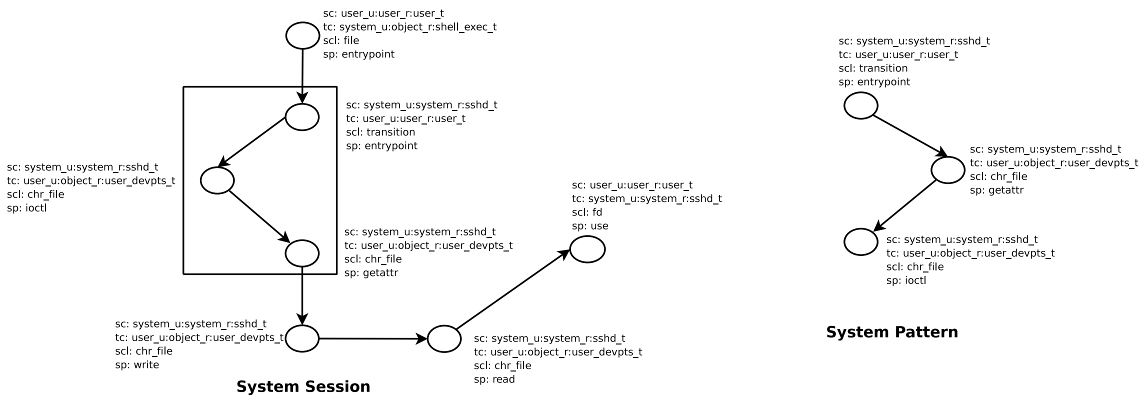

sc: user_u:user_r:user_t
tc: system_u:object_r:shell_exec_t
scl: file
sp: entrypoint
sc: system_u:system_r:sshd_t
tc: user_u:user_r:user_t
scl: transition
sp: entrypoint
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: ioctl
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: getattr
sc: user_u:user_r:user_t
tc: system_u:system_r:sshd_t
scl: fd
sp: use
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: write
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: read
System Session
sc: system_u:system_r:sshd_t
tc: user_u:user_r:user_t
scl: transition
sp: entrypoint
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: getattr
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: ioctl
System Pattern

Figure 4: System Pattern Recognition

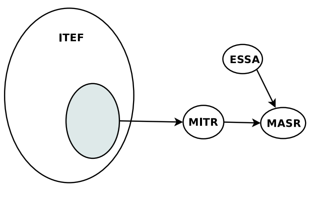


Figure 5: Massive Activity Detection

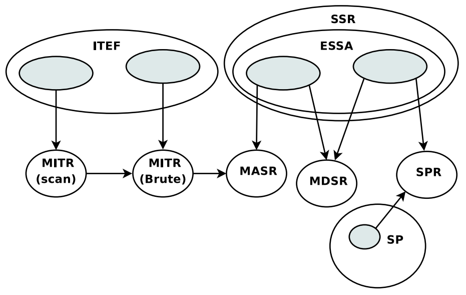


Figure 6: Complex Scenario Detection

unclassified (i.e. unrecognized) session with each System Pattern. Let n be the number of session to classify and p the number of patterns. The complexity $\mathcal{C}_{spr} = \Theta(np)$ in number of comparisons. Each comparison is actually quite complicated. Indeed, it adresses the problem of counting sub-graphs isomorphisms, which in general is at least in $\Omega(e!)$ where e is the number of event (nodes) of the graph (i.e. the System Pattern). Hopefully, recent works (Eppstein, 2000) have provided interesting theoritical results. They show that for a graph G that is simply a tree (i.e. a system session tree here), with a fixed number a of attributes per nodes (which is also the case here, where $2 < a < 7$), the counting of the sub-graphs, that are isomorph to another a fixed graph P, (i.e. a Session Pattern here) can be done linearly to e. With those results, the SPR complexity is reduced to $\mathcal{C}_{spr} = \Theta(np \times ke_G)$, where k is a multiplicative constant of the number of events e_G of G.

This is an important result as it can lead to a real-time correlation using the recognition of session patterns.

Massive Activity Reconstruction:

Massive IT *Reconstruction (*MITR*)*: In constrast to (Valeur et al., 2004) (Kruegel et al., 2005), this module classifies massive system activities. It merges and counts all the events (i.e. interactions or IT) that are the same (all attributes with an equal value) in a given time window to create a MITR meta-event. If they are too many above a given threshold, it creates a meta-event MITR. The definition of these thresholds remains outside the scope of this work. The meta-event contains relevant informations about the massive activity like its source(s) and destination(s). It is not limited to a one to one massive activity, it can also classify many to many and any other variations of a massive activity.

A MITR meta-event contains: $scontext_{i1}$, $tcontext_{i2}$, sec_class_{i3}, sec_perms_{i4}, $dateBegin$, $dateEnd$, $exchange\_type_{i5}$, $named\_socket_{i6}$, s_addr_{i7}, d_addr_{i8}, s_port_{i9}, d_port_{i10}, $count$ and id_mitr, where each ik equals 1 or n (with n the number of machines involved in the massive interaction).

*Massive Activity Session Reconstruction (*MASR*)*: This module is another original proposition. It ables to link a massive activity with sessions that created it or have been created by it. When a link is found, it creates a meta-event MASR (see figure 5).

This module generates a MASR meta-event with the following attributes: id_mitr, $idsession$, $hostname$ and $direction$.

For example, MASR links a massive activity, like a DDoS that has been launched from a monitored computer, with the session that have launched it (using meta-knowledges extracted from MITR meta-event and ESSA/SSR meta-event).

Complex Scenario Detection (CSD)

In other correlation approaches, a special language enables to express the links between "meta-events" (Cuppens and Miège, 2002; Ning et al., 2001; Valeur et al., 2004; Kruegel et al., 2005; Eckmann et al., 2002).

Our approach is different. CSD is not really a module like the other ones, but a combination of modules that represents the way to link the previously seen modules all together (with none, one or multiple implementations of each one) in order to detect a complex scenario. Actually, this combination is implemented by the connexion of automata representing each module.

The abstract figure 6 represents a “Complex Scenario” detection. It starts on the left with a scan of a monitored computer (i.e. a MITR meta-event) followed by a bruteforce attempt (i.e. a MITR meta-event. Then, the bruteforce intrusion attempt (i.e. the MITR meta-event) is linked (as an MASR) to the ESSA representing activities on the system caused by the bruteforce (cf. MASR).

On the right side, the SPR module classifies another ESSA using a system pattern (SP). This ESSA represents the system session caused by the successful intrusion following the bruteforce attempt.

This SPR is linked with the MASR meta-event as a new MDSR meta-event.

SYSTEM PATTERN LEARNING (SPL)

Currently, a learning module is proposed to compute automatically the System Patterns (used in SPD and CSD). For this purpose, each system automaton is compared with all the other system automata, according to a similarity level. The comparison is done on the nodes (i.e. filtered events or factorized filtered events). If at least two successive nodes are equals, a new pattern is created. This pattern contains only the equal nodes with the attributes corresponding to the similarity level used to create the pattern. If the pattern already exists, its frequency is increased.

The main advantage of this approach is that it is totally unsupervised. The algorithm compares all the sessions one by one in order to have correctness. In terms of theoritical complexity, as each single session is compared with all the others. Thus, the complexity is $\mathcal{C}_{spl} = \Theta((n-1)!)$ in number of comparisons between sessions, where n is the number of reconstructed sessions. Specific DNA sequence alignment algorithms, like the BLAST family (Altschul et al., 1990), enable to provide a lower complexity. Thus, this complexity problem can be addressed. Our first results provide automation for computation of the System Patterns (on the next section).

EXPERIMENTATION

Physical Architecture for the Correlation Process

The decentralized architecture presented in (Krugel et al., 2001), did not fit our needs because we wanted to limit the overhead on the monitored hosts due to security monitoring. Also, it seems too dangerous for us to let the correlation process take place on the same computers where scenarios could happen. In addition, we needed a scalable architecture for the various steps of the correlation process.

Thus, we proposed a *grid* architecture for the whole correlation process, supporting a large amout of RAW events, and also providing almost real-time recognition.

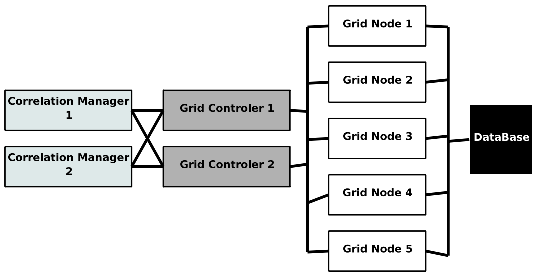


Figure 7: Correlation Architecture on Grid

The *grid* is composed of 3 computation nodes of 3Ghz and 512MB of RAM). We introduced a centralized database for the data retention but also to use a dedicated computing *grid* for our correlation process as shown in the figure 7. The dedicated computing *grid* also brings us the ability to easily implement our modular architecture: each module is implemented as an agent that can be instantiated on multiple nodes of the *grid* to compute various data. Thus, we have an easier scalable processus: we just need to add new computers on the fly, using PXE boot to increase the computing power.

Experiment Results

As said before, even if our conceptual architecture can correlate multiple informations coming from multiple sources and sensors, at this stage of our works, our solution has been experimented using SELinux events.

Those events come from a High-Interaction HoneyPot with 4 hosts using GNU/SELinux over Gentoo and Debian. One year of events are processed using our *grid* implementation. An IBM DB2 database is used to store those logs.

Event Filtering: One host of our honeypot generated around 164,000,000 of raw events i.e. SELinux system calls. The Event Filtering EF module is important because syslog makes many errors when it reports events. EF detects about 3 percents of wrong events but this rate can increase with an important activity of the SELinux host. During our experiments, EF produces 160,000,000 of valid events.

160,000,000 events cannot be processed using a classical database such as MySQL because of memory and CPU overhead. So, the IT Event Factorizing module is required in order to reduce the number of events that has to be stored in the database. Starting from 160,000,000 EF events, ITEF succeeded to produce only 8,000,000 meta-events (as shown on table 2).

System Session Identification: The System Session Identification module SSI was applied to the four SELinux hosts of our honeypot. The SSI module was set to use only SSH and FTP services as entry point (other services, such as HTTP SMTP IMAP, have not been

Without ITEF	With ITEF
160,000,000	8,000,000

Table 2: Events number for 1 Year on 1 SELinux Host

considered by those experiments). Table 3 shows the number of sessions for each of the four SELinux hosts. *Gentoo*1, *Gentoo*2 and *Debian* are connected directly to the Internet while *Gentoo*3 is reachable through one of those three gateways. Differences between *Gentoo* and *Debian* is due to the SElinux policies that are more precise on *Debian* than on *Gentoo*. Starting from a *Gentoo* host with 160,000,000 EF events, SSI identifies an average of 40,000 sessions.

	Gentoo 1	Gentoo 2	Gentoo 3	Debian
Sessions	58,163	30,825	79	139,859

Table 3: Number of Sessions Detected for Each Computer

System Session Reconstruction: The System Session Reconstruction module associates each session with all the system activities. Many sessions are almost empty. A typical example can be when an attacker only tests a password and disconnects (even when the password was the good one). Starting from 40,000 SSI sessions, SSR generates only 8,000 sessions where the activities continue after the login attempts. Table 4 shows that SSR consumes between 800 and 3,000 milliseconds to reconstruct a session according to the number of SELinux events included in that session. The computation duration includes a constant time of 700ms for launching the agent. Many sessions requires 800ms i.e. uses only 100ms for the algorithm run. The largest session takes, a longer time, up to 2300ms. This worst case was reached by a SSH session with a local bruteforce on the ftp server. In that case, the machine spent time for swapping because the required memory was big. This increases abnormally the computation time for this session.

	Small	Average	Large	Huge
Length	800	1500	3000	15700

Table 4: SSR Computation Time (in milliseconds)

For one year of experiments, the average time is 1500ms with our *grid* configuration. With 40,000 identified sessions, it took on average 190 hours to analyze one year of logs. So, it takes half an hour to analyze the logs of the day and 20ms for 1mn of logs. As one can see, it is possible to reconstruct sessions in real-time with our architecture.

Massive* IT *Reconstruction: The MITR module has been able to detect one bruteforce on the FTP service. Due to the policy configuration of SELinux, some relevant system calls were not audited. Those missing events prevented to detect SSH bruteforces.

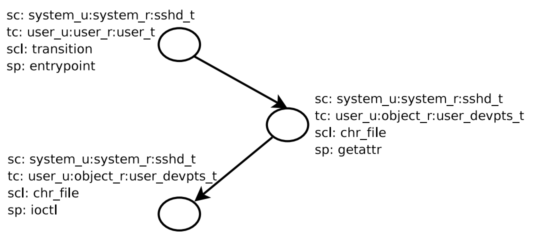

sc: system_u:system_r:sshd_t
tc: user_u:user_r:user_t
scl: transition
sp: entrypoint
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: getattr
sc: system_u:system_r:sshd_t
tc: user_u:object_r:user_devpts_t
scl: chr_file
sp: ioctl

Figure 8: Example of a Learned (Basic) System Pattern

System Pattern Learning: As we said in the System Pattern Learning section, the rough complexity of the construction of the system patterns is uncomputable. However, the study of the data from our honeypot showed us some directions we could exploit. In average a session includes about 600 nodes. The comparison between those sessions takes between 1 seconds for the lowest similarity level $a = 2$ and about 200 seconds for the highest similarity level $a = 7$. About $40,000$ sessions have been collected during one year. That means at least $\mathcal{O}(40000! \times 1) \simeq \infty$ seconds to compute all the sessions at the lowest similarity level. Of course, the comparison at higher similarity level are less numerous, and typically only few sessions are compared at the highest level. To minimize the computation, only big sessions have been considered, stating that those sessions contain also smaller ones. Among these 40000 sessions, only 72 had above the average number(i.e. 600 nodes per session). Our first experiments build 30 patterns after a computation of 3 weeks manually ended.

As previously said, we are currently investigating bioinformatics algorithms (like BLAST) and also unsupervised classification and learning technics to overcome this (sub-)graph clustering problem.

System Pattern Recognition: The SPR module used System Patterns created with the SPL module (see System Pattern Learning section).

The figure 8 shows a system pattern representing the connection of a user through SSH i.e. a migration from the SSH context to the user context, then the opening of a virtual console and finally an interaction between the user and the virtual console.

6 different levels of similarity have been considered for the recognition of 40,000 sessions (see the System Pattern Learning section). As seen in table 5, 13040 sessions contain this pattern for a similarity level 2 i.e. the comparaison is based only on security contexts (source and target). For level 3 (addition of one variable), 13000 sessions respect this pattern, and only 6680 sessions for level 4 (addition of one variable). Table 5 shows also that the recognition did not find any session respecting the pattern for higher levels of similarity. The maximum level of similarity can be increased using other patterns. Actually, the learning module is too slow to compute

enough and accurate patterns, that's why we are currently working on bioinformatics algorithms.

Similarity	**OK**	**Not OK**
2	13040	26960
3	13000	27000
4	6680	33320
5	0	0
6	0	0
7	0	0

Table 5: Classified System Patterns

Migrating/Distributed Session Reconstruction: MDSR recognized 4 IslandHopping (3 using network connections, 1 using a Unix socket). MASR was able to link 7 sessions that took part of a massive attack. It is worth saying that the limited numbers of complex sessions (such as MDSR, MITR and MASR) is due to the efficient protection provided by the SELinux kernel. This MAC protection limits really the possibilities of ordinary users and controls all the interactions between a process and the system ressources. So, it is really hard to conduct advanced attacks on such systems. However, one can see that possibilities to violate the security exist.

CONCLUSION

This paper presents several modules that cooperate to analyze complete sessions of SELinux system activities. Complex sessions can be reconstructed such as distributed, migrating sessions using several connections. Currenlty, it is the only solution able to analyze SELinux logs. The problem is complex due to the large amount of events generated by a complete system. The solution has been experimented during one year using several high-interaction honeypots. More than 160,000,000 events have been analyzed for each honeypot. Thus, 8,000 sessions, with relevant activities, have been recognized. Moreover several complex sessions have been completly reconstructed. Using a *grid* approach, a real-time classification of system sessions has been proposed. The solution uses System Patterns in order to analyze the logs. A learning module is proposed to compute automatically the relevant patterns starting from the reconstructed sessions. The paper shows that rough Pattern learning is a NP-Complete problem. However, heuristics have been proposed and real constructed patterns have been presented. Future works will address how to use the systems sessions to defined advanced security properties such as integrity models or to detect the violation of those properties. Applications will be proposed either for protecting a system of for detecting the intrusions. Moreover, several improvements will be proposed. First, newer parallel processing could reduce the required computation time. Second, newer heuristics will be defined to compute the Activity Patterns.

References

Altschul, S., Gish, W., Miller, Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410.

Bowen, R., Chee, D., Segal, M., Sekar, R., Uppuluri, P., and Shanbag, T. (2000). Building survivable systems: An integrated approach based on intrusion detection and confinement. In *DARPA Information Survivability Symposium*. IEEE Computer Society.

Briffaut, J., Lalande, J.-F., Toinard, C., and Blanc, M. (2006). Collaboration between mac policies and ids based on a meta-policy approach. In Smari, W. W. et McQuay, W., editor, *Workshop on Collaboration and Security (COLSEC'06)*, page 48–55, Las Vegas, USA. IEEE Computer Society.

Chari, S. N. and Cheng, P.-C. (2003). Bluebox: A policy-driven, host-based Intrusion Detection System. *ACM Transaction on Information and System Security*, 6(2).

Cuppens, F. and Miège, A. (2002). Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*, Oakland, USA. IEEE.

Eckmann, S., Vigna, G., and Kemmerer, R. (2002). STATL: An Attack Language for State-based Intrusion Detection. *Journal of Computer Security*, 10(1/2):71–104.

Eppstein, D. (2000). Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291.

Kruegel, C., Valeur, F., and Vigna, G. (2005). *Intrusion Detection and Correlation: Challenges and Solutions*. Springer.

Krugel, C., Toth, T., and Kerer, C. (2001). Decentralized event correlation for intrusion detection. In *Information Security and Cryptology*, pages 114–131.

Molina, J., Chorin, X., and Cukier, M. (2007). Filesystem activity following a ssh compromise: An empirical study of file sequences. In *ICISC*, pages 144–155.

Ning, P., Reeves, D., and Cui, Y. (2001). Correlating alerts using prerequisites of intrusions. Technical Report TR-2001-13, North Carolina State University.

Qin, X. (2005). *A Probabilistic-Based Framework for INFOSEC Alert Correlation*. PhD thesis, Georgia Institute of Technology.

Valeur, F., Vigna, G., Kruegel, C., and Kemmerer, R. A. (2004). A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on dependable and secure computing*, 1(3).

COOPERATIVE INTRUSION DETECTION SYSTEM (CIDS) IN GRID ENVIRONMENT ON UNLABELLED DATA

Abdul Samad Haji Ismail[1], Dahliyusmanto[2], Witcha Chimphlee[3], Abdul Hanan Abdullah[4], Kamalrulnizam Abu Bakar[5], Md Asri Ngadi[6]

[1,4,5,6]Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia, Tel: (607)-5532003, Fax: (607)-5565044

[2]Department of Electrical Engineering, Faculty of Engineering, University of Riau. Kampus Bina Widya Km 12.5 Simpang Baru 28293, Riau, Indonesia, , Tel: (62)-76163266, Fax: (62)-76163279

[3]Faculty of Science and Technology, Suan Dusit Rajabhat University, 295 Rajsima Road, Dusit, Bangkok, Thailand, Tel: (60)-2445224, Fax: (60)-6687136

[1]abdsamad@utm.my, [2] yoes_mantho@sigma-snt.com, [3]witcha_chi@dusit.ac.th, [4]hanan@utm.my,, [5]knizam@utm.my, [6]dr.asri@utm.my

KEYWORDS
Anomaly detection, unsupervised clustering, unlabeled data, Fuzzy Clustering, Grid Environment.

ABSTRACT

Intrusions pose a serious security risk in a network environment. The intrusion detection in computer networks is a complex research problem. Applying intrusion detection to the fast growing computational Grid environments improves the security and is considered to be the heart of this new field. Flexible cooperative distributed intrusion detection architecture is introduced that suits to and benefits from the underlying Grid environment. Intrusion detection techniques fall into two general categories: anomaly detection and signature recognition, with each one complements one other. Anomaly intrusion detection normally has high false alarm rates, and a high volume of false alarms will prevent system administrators from identifying the real attacks. This paper presents a clustering-based anomaly intrusion detection algorithm which trains on unlabeled data in order to detect new intrusions. This work does not make a strict hypothetical requirement with the percentage of attacks has to be less than a certain threshold (e.g.,~1.5%). It also does not label clusters by considering the sparse density is attacks. We propose a new labelling cluster algorithms, called NMF (Normal Membership Factor) that is capable of increasing normal detection which would be indicative of decrease false positive rate. Our method is able to detect many different types of intrusions, while maintaining a low false positive rate as verified over the Knowledge Discovery and Data Mining-KDD CUP 1999 dataset.

1. INTRODUCTION

Research and development efforts within the Grid community have produced protocols, services, and tools that address the challenges arising when seeking to build scalable virtual organization (VOs). A virtual organization is defined as a set of individuals and/or institutions sharing resources and service under a set of rules and policies governing the extent and conditions for that sharing. As stated in [1], "the sharing that Grid environments are concerned with is not primarily file exchange but rather direct access to computers, software, data and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering

This sharing is, necessarily, high controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. The technologies that have evolved from the Grid community include security solutions that support management of credentials and policies when computations span multiple institutions; resource management protocols and services that support secure remote access to computing and data resources and the co-allocation of multiple resources. The recent history of attacks against the information system and widespread vulnerabilities indicate that security threats have dramatically escalated in speed, impact and frequency. Grid Infrastructures like Globus [2], Legion [3] and Condor [4] are particularly vulnerable to intrusions and require and adequate level of security for users, data and resources. Grid are open environments, compromise of single resource may provide unauthorized access to data and services from other

Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

system. Grids are vulnerable to large-scale attacks that may cause disruption of the Grid services. Thus, it is essential for Grids to support prevention, detection and automatic response to intrusion attempts and work cooperatively.

2. RELATED WORK AND BACKGROUND

2.1 Grid Computing

Grid was proposed in the mid 1990 and has been widely used in many areas, such as bioinformatics, medicine, astronomy, chemistry, agriculture, business and engineering design, to solve large-scale and complex problems [5]. Today, many organizations such as Compaq, Sun Microsystems, Fujitsu, Hitachi and NEC fall into Grid research. They have adopted Globus Toolkit, developed by USC'S Information Science Institute (ISI) and Argonne National Laboratory, as their basic platform. Globus toolkit is an open-architecture, consisting of security, information infrastructure, resource management, data management and communication components. It facilitates creation of usable Grids, enabling high-speed coupling of people, computers, databases, and instruments [6].

In Grid environment, the security challenges face under three categories [7]: integration with existing security architectures and model across platforms and hosting environment, interoperability with different hosting environments (e.g., J2EE servers, .NET servers, Linux systems) at multiple level such protocol level, policy level, and identity level, and trust relationships among interacting hosting environments.

2.2 Intrusion Detection System (IDS)

Most traditional intrusion detection systems (IDSs) [8] take either a network-based or a host-based approach for recognizing and responding to attacks. These systems look either for attack signatures, specific patterns that indicate malicious or suspicious intent or deviation from a normal profile (anomaly) that indicate an attack. A network-based IDS looks for these patterns and anomalies in network traffic.

A host-based IDS looks for attack signatures and anomalies in system audit trails or application logs. In most cases intruder exploit vulnerabilities and misconfiguration in application server to break into a system. The application-level attacks can enter a system through the same open "door" in the perimeter defences used by legitimate users. Therefore, these attacks are difficult to detect. Current Network-based and Host-based IDSs work in isolation from access control for the application the systems aim to protect. The lack coordination and inter-operation between these components prevents detecting sophisticated attacks and responding to ongoing attacks in real time, before they cause damage. Another disadvantage of currently available IDSs is a large number of false positive (IDS reports an attack when none has occurred). Reports of attacks can trigger response actions (e.g., termination of the offending connections). Thus an inaccurate IDS decision (a false alarm) may result in disruption of service to legitimate users. Therefore, successful intrusion detection requires accurate and efficient models for analyzing a large amount of application, system and network audit data and real time response to the attacks.

3. THE PROPOSED GRID INTRUSION DETECTION ARCHITECTURE

In this section, the research proposes an architecture of Grid intrusion detection. This architecture was designed with the Grid characteristics in mind. The Grid intrusion detection architecture has two main parts as shown in Figure 1. The first is intrusion detection agent (represented by small black circle) that is responsible for gathering information. And the second part is the intrusion detection server (represented by big circle) that is responsible for analyzing the gathered information and cooperating with other IDSs to detect intrusions.

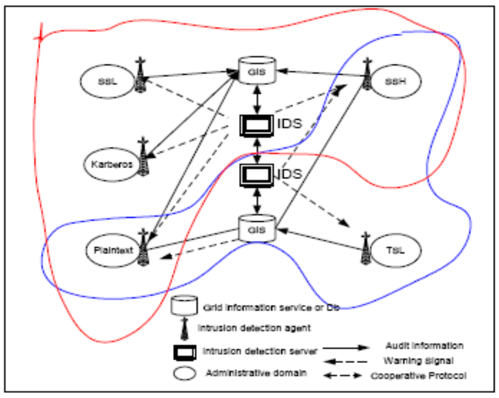
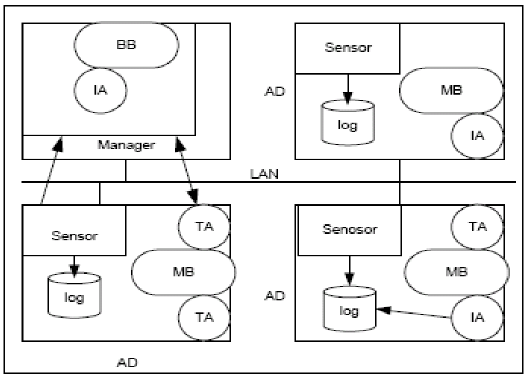


Figure 1: Proposed Grid Intrusion Detection Agent

3.1 Structure of Intrusion Detection Agent

In many conventional network intrusion detection systems, each target system transfers its system log to an intrusion detection server, and the server analyzes the entire log in search of intrusions. Methods of this kind fall under the client/server paradigm. In a large-scale network deploying an intrusion detection system, network traffic will be extremely high, since the volume of the system logs that are routinely transferred is very large, though most of it has no information related to intrusions. Therefore, this type of intrusion detection system on a large-scale network does not fulfil its function efficiently. To solve this problem, we adopted a mobile-agent paradigm in detecting intrusions. Mobile-agents autonomously migrate in host in this case is the administrative domain to collect only information related to intrusions, eliminating the need to transfer system logs to the server. We can deploy Intrusion Detection Agent on a local area network, the protocol of which is TCP/IP. Intrusion Detection Agent consists of sensors, message boards, tracing agents, and information-gathering agents. The system details are show in Figure 2.



AD : Administrative Domain
MB: Message Board
IA : Information_Gathering Agent
TA : Tracing Agent
BB : Bulletin Board

Figure 2: Structure of Intrusion Detection Agent

- **Manager**

The manager analyzes information gathered by information gathering agents and detects intrusions. It manages the mobile agents and bulletin boards and provides an interface between administrators and the system. The manager accumulates and weighs the information entered by the mobile agents on the bulletin board, and if the weights exceed a set threshold, the manager concludes that an intrusion has occurred. One manager resides on each network segment.

- **Sensor**

The sensors, present on each host, monitor system logs in search of suspicious activity. If a sensor finds that activity, it reports this finding to the manager.

- **Tracing Agent**

The intrusion-route tracing agent, called simply the tracing agent, traces the path of an intrusion and identities its point of origin; the place from which the user leaving a n activity remotely logged onto the target host. En route to finding the origin, a tracing agent can find any intermediate nodes that may be compromised. The manager, sensor, and tracing agent work together in the following way. First, the sensor detects a suspicious activity and reports it to the manager, and then the manager launches a tracing agent to the host system. The tracing agent migrates autonomously from machine to machine and traces the intrusion independently; without the manager. When many suspicious activities are found by a single host system in different sessions over a short period of time, many tracing agents corresponding to the suspicious activity are launched into the host system. A tracing agent makes no judgments about intrusions, and is not capable of deciding whether or not an intrusion has occurred. A tracing agent can migrate to any system in which Intrusion Detection Agent (IDA) is installed.

- **Information Gathering Agent**

An information-gathering agent, which is mobile, gleans information related to a suspicious activity from a target system. Each time a tracing agent in pursuit of an intruder is dispatched into a target system, it activates an information gathering agent in that system. Then the information gathering agent gleans information depending on the type of activities, returns to the manager and reports. If the tracing agent migrates to another target system, it will activate another information-gathering agent, which will gather information on the next host system. Many information gathering agents may be activated by many different tracing agents on the same target system. An information-gathering agent is not capable of deciding whether an intrusion has
occurred.

- **Bulletin Board and Message Board**

The bulletin board and the message board are common use area that can be accessed by tracing agents and information gathering agents, and means of information exchange. There is a message board on each target system, used by tracing agents for exchanging information; any tracing agent can know whether a track under its scrutiny has already been traced by other agents, and can use this information in deciding where to go. The bulletin board is on the manager machine and is used for recording information gathered from host systems by information-gathering agents, as well as for integrating the information gathered about every tracing route.

3.2 Action of Intrusion Detection Agent

The following is of intrusion detection agent works after a sensor detects a suspicious activity on an administrative domain. Intrusion detection agent accumulates the data required by intrusion-route tracing (i.e., about network connection, the various processes running on the system, etc.) on each target system in advance.

i. Each sensor on the administrative domain seeks a suspicious activity from the system log.
ii. If the sensor detects a suspicious activity, it reports it to the manager.
iii. The manager dispatches a tracing agent to the target system where the suspicious activity was detected.

iv. The tracing agent arrives at the administrative domain and activates an information-gathering agent.
v. The information-gathering agent collects information related to the suspicious activity on the administrative domain.
vi. After activating the information-gathering agent, the tracing agent investigates the point of origin of the suspicious activity in an effort to identify the user's remote site. The tracing agent can derive this from the accumulated data about network connection and processes running on the system.
vii. After collecting information, the information gathering agent, independent of the tracing agent, returns to the manager, and enters the information on the bulletin board.
viii. The tracing agent moves to the next administrative domain on the tracing route, and it activates a new information-gathering agent.
ix. If the tracing agent arrives at the origin of the route, or cannot move anywhere, or if other tracing agents have chased the route it could follow, it returns to the manager.

In cases where a sensor detects many suspicious activities on an administrative domain occurring over a short period of time, or if sensors detect suspicious activities on many administrative domain, intrusion detection agent works as described above for all suspicious activities detected. The SSL, Kerberos Plaintext, TLS and SSH are the administrative domains (resources) in a Grid environment. Each administrative domain will have an intrusion detection agent to collect data and the intrusion detection agent will register with one or more IDSs which will analyze the gathered data. Intrusion detection agent will be designed for each class of resources to handle heterogeneity.

3.3 Lab Environment and Software

This section gives an overview of the configuration of the software and hardware used in our lab. We built a very small scenario. It is the simplest Grid environment, intended to help illustrate the concepts and components behind the Grid and GT4.0.5. An Ethernet LAN, three Intel® Pentium IV machines, and one Laptop Intel® Centrino Duo machine were used. In Figure 3, we illustrate this environment with the host names and the functionality of each machine. The host names are T1, T2, T3 and T4. Also, an infrastructure server, called m0, was set. The machines should have a clock speed of 1 GHz, 512 MB of minimum memory, and hard drives totalling 40-120 GB.

If we have more than five machines available, we can build a bigger scenario for Proof-of-Concepts (PoC) proposals and/or demos. For that, you simply include more servers, such as T5, T6, and so on.

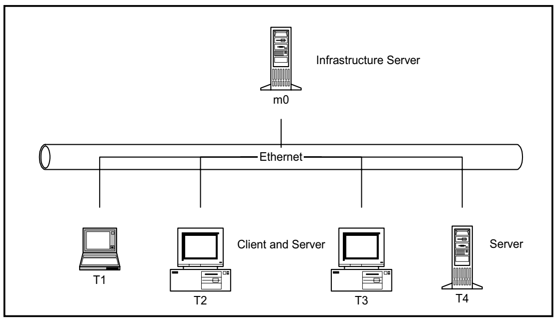


Figure 3: Hardware Environment and Software Functions of Each Machine

We give summarizes the names of the machines to be used in the Grid, their IP addresses, and the software to be installed on them as shown in Table 1.

Table 1: Host names and IP addressing

Hostname	**Internet Protocol (IP)**	**Description**
T1.grid.research.com	192.168.0.241	Globus server and Client machine
T2.grid.research.com	192.168.0.242	Globus server and Client machine
T3.grid.research.com	192.168.0.243	Globus server and Client machine
T4.grid.research.com	192.168.0.244	Globus server
M0.grid.research.com	192.168.0.10	Infrastructure server

Next, we define the users and groups that you want to use before implementation. Table 2 contains the list of user and group IDs used in our lab.

Table 2: IDs and Passwords

User ID	Group ID	password	Activities
root	root	<passwd>	Super user needs
globus	globus	<passwd>	Globus Toolkit environment. For installation and execution of the toolkit
logicacmg	logicacmg	<passwd>	End user environment. For job execution on the Grid.

GT4.0.5 needs several files, or tools, in order to complete the installation such as Java SDK, Apache Ant, Junit, Postgresql and Globus Toolkit. In this installation, we used Globus Toolkit bundle Version 4.0.5.

3.4 Implementation of Grid Intrusion Detection Architecture

This research uses two stages to test the proposed Grid intrusion detection architecture. The first stage simulates the intrusion detection agent and the Grid environment. Most of the available Grid simulation toolkits are designed for resource management and scheduling problems. For this reason this research uses the grid simulation toolkit based on GridSim [9] to satisfy the needs. The simulation environment simulates users with different behaviours, resources with associated intrusion detection agents, and intrusion detection agent registration with IDSs.

This allows us to perform the required experiments. Each experiment generate a dataset consisting of one or more log file. Figure 3 shows the simulation environment with dummy IDSs that only generate log files reflecting the data to be analyzed. The next stage implements the IDS modules and test them with the data generated from the simulation stage (Figure 4). In this initial implementation we use homogeneous IDSs for simplification. We believe that currently the best intrusion detection technique to use in this case is host-based anomaly intrusion detection [10].

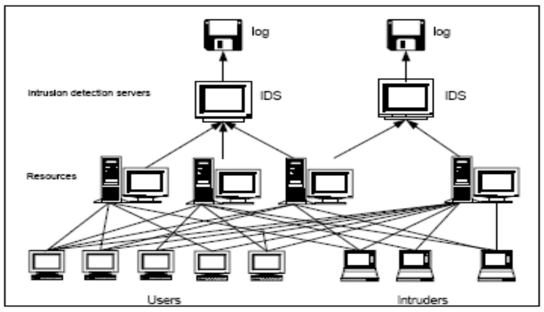


Figure 4: A Simulated Grid and Data Modules

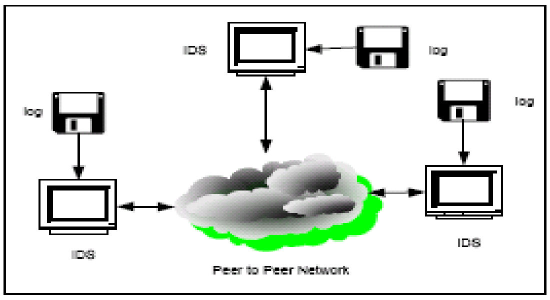


Figure 5. The implementation of IDS

4. NORMAL MEMBERSHIP FACTOR

The host in this case is the administrative domain with all its resources. The assigned intrusion detection agents will gather information about the user's interactions with this domain. The anomaly detection is implemented using *Normal Membership Factor* (NMF). The NMF is labelling clusters technique that identify number of instances in term of normal and attack. It is important to note that when labelling the clusters its relation to each of the clusters is taken into consideration. The results of labelling based on these factors will therefore include a degree of probability of the clusters belonging to normal group. As in Portnoy et al [11], they were determined the algorithm for cluster labelling follow their first assumption. Their first assumption about data is that normal instances constitute an overwhelmingly large portion (> 98%) of the training dataset. Under this assumption it is highly probable that cluster containing normal data will have a much larger number of instances associated with them then would clusters containing anomalies. Therefore, they labelled some percentage N of the clusters containing the largest number of instances associated with them as normal and the rest of the clusters are labelled as anomalous and are considered to contain attack.

In this paper, the portion of normal instances is not constituted to large or more than 98%. Only if the normal instance portion should more than 80 percent like Pornoy work. Then, it is not probable to identify only the group of the largest instances is normal type. Therefore, this work uses a new labelling algorithm to identify the other clusters that may be having normal pattern and then gather into normal group to reduce the false positive rate as well. For labelling the normal cluster, the two factors as described above are taken into account.

In this approach, to calculate the weight of clusters and its NMF of cluster are calculated as follows:
Weight of clusters (ci): the weight or size of the cluster is considered greatly reduces the affect of anomalies such as outliers. By multiplying the inverse distance to the cluster centres by the weight of the cluster, and dividing by the summarized weight of all the cluster centres as Equation (1).

$$WC(c_i) = \frac{1}{d(Normal, c_i)} \times \frac{number\ of\ instance\ in\ c_i}{number\ of\ all\ instance} \quad (1)$$

where $d(Normal, c_i)$ is the distance between the normal cluster and the other clusters, and i is the number of clusters. Normal Membership Factor (ci): In order to get the probability of clusters belonging to the normal cluster, the Normal Membership Factor is calculated as Equation (2).

$$NMF(c_i) = \frac{WC(c_i)}{\sum_{i=1}^{c} c_i} \qquad (2)$$

where $\sum_{i=1}^{c} c_i$ is the summarized of all the weight of cluster. If NMF values have greater than 40 percents then gather that clusters into normal group.

5. EXPERIMENTAL SETUP AND RESULT

There are numerous methods that discuss the evaluations of intrusion detection systems. Some methods emphasise the important of detection rate (DR) and false positive rates (FPR); while other look into the novel pattern detection rate. The performance of classifiers is evaluated with respect to their classification of unseen normal and intrusive patterns. The metrics embrace here are the generalisation abilities of the classifiers because they are the most important aspect of an anomaly detection scheme. Evaluation of the generalisation capability of any intrusion detection should consider the ability of the system to recognise new normal as well as intrusive behaviours.

The best performing instances of classifiers for each data set are chosen. Six major metrics are developed to quantify the performance of the classifiers in this thesis. These metrics are calculated based on the testing patterns according to following relations.

- **Normal Generalisation (NG):** the ratio of correctly classified intrusive vectors to the total number of intrusive vectors.
- **Intrusive Generalisation or Detection Rate (DR):** the ratio of correctly classified intrusive vectors to the total number of intrusive vectors.
- **Overall Generalisation (OG):** the ratio of correctly classified vectors to the total number of the vectors. It is important to mention that this metric is sensitive to the imbalanced numbers of the normal and abnormal testing patterns.
- **Discrimination Ability (DA):** the average of the normal generalisation and the intrusive generalisation. This metric is developed due to the problem of imbalanced number of the testing patterns of normal and abnormal behaviours which affect the overall generalisation metric. This metric is dependent on the *percentage* of the generalisation of both behaviours. It is not like the overall generalization metric which is dependent on the *number* of testing.
- **False Positive (FPR):** the ratio of incorrectly classified normal vectors to the total number of normal vectors.
- **False Negative (FNR):** the ratio of incorrectly classified intrusive vectors to the total number of intrusive vectors.

In choosing the best performance of algorithm is the greatest in first four metric (NG, DR, OG, DA) and lowest in two last metric (FPR, FNR). Thus, if values of first four metrics are high and two last metrics are low it means algorithm is good, on the other hand it means that algorithm is not good for detection. The results show in Table 2 with NG, DR, OG, DA, FPR and FNR in 92.29, 95.09, 92.05, 93.69, 7.71, and 4.91 respectively.

Tabel 3: The Six Metric Results

Metric	%
NG	92.29
DR	95.09
OG	92.05
DA	93.69
FPR	7.71
FNR	4.91

6. CONCLUSION

The effect of trust relationships between different resource owners and the use of heterogeneous IDSs will be further investigated. With Heterogeneous IDSs and trust relationships more complex algorithms will be needed for the cooperation module that will need further investigations. The application of the Grid in real problems will help in building a knowledge base of attack signatures that will enable the use of misuse intrusion detection with the Grid.

The experimental performance shows the outstanding result in all of the evaluation criteria. The results show that the high accuracy and low false positive rate. Intrusion detection model is a composition model that needs various theories and techniques. One or two models can hardly offer satisfying results. We plan to apply other theories and techniques to operate in a high accurate and low false alarm rate in intrusion detection in our future work.

ACKNOWLEDGEMENTS

This research was supported in part by grants from MOSTI, Malaysia. Project Number (UTM0000505).

REFERENCES

Foster. I., Kesselman. C., and Tuecke. S. 2001. “The Anatomy of the Grid: Enabling Scalable Virtual Organizations.” International Journal of Supercomputer Applications. 15(3).

Foster. I., and Kesselmen. C. 1997. “Globus: A Meta computing Infrastructure Toolkit.” International Journal of Supercomputer Applications.

Lewis. M., and Grimshaw. A. 1996. “The Core Legion Object Model”. In proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing.

Lizkow. M., Livny. M., and Mutka. M. 1998. “Condor – a hunter of idle workstations.” In proceedings of the 8th International Conference on Distributed Computing Systems.

Foster. I., and Kesselmen. C. 1999 (eds.). “The Grid: Blueprint for a New Computing Infrastructure.” Morgan Kaufmann.

Nagaratnam. N., Janson. P., Dayka. J., Nadalin. A., Siebenlist. F., Welch. V., Foster. I., and Tuecke. S. 2002. “The Security Architecture for Open Grid Services.” Open Grid Service Architecture Security Working Group (OGSA-SEC-WG). Global Grid Forum.

Bace. R., and Mell. P. 2001. “Intrusion Detection Systems.” National Institute of Standard and Technology (NIST) Special Publication on Intrusion Detection Systems.

M. Murshed, R. Buyya, and D. Abramson. 2002. "GridSim:A Grid Simulation Toolkit for Resource Management and Scheduling in Large-Scale Grid Computing Environments". 17th IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2002), April 15-19, Fort Lauderdale, FL, USA.

M. Tolba, M. Abdel-Wahab, I. Taha, and A. Al-Shishtawy. 2000. "Distributed Intrusion Detection System for Computational Grids". Second International Conference on Intelligent Computing and Information Systems, March.

Portnoy, L., Eskin, E. and Stofo, S.J. 2001. “Intrusion detection with unlabeled data using clustering.” in Proceedings of ACM CSS workshop on data mining applied to security (DMSA-2001).

BDUL SAMAD ISMAIL, He is a Deputy Dean (Development) and An a Associate Professor at Faculty of Computer Science & Information System, University Teknologi Malaysia. He received his B.Sc. in Mathematical./Computer Science from University of Wisconsin Superior, his M.Sc. in Computer Science received from Central Michigan Univeristy, he also graduated his Ph.D in similar field (Computer Science) from University of Swansea. His email is address is : abdsamad@utm.my.

DAHLIYUSMANTO was born in Pekanbaru, Riau - Indonesia and went to the University of Putra Indonesia Padang – West Sumatera, where he studied computer engineering and obtained his degree in 1996. He worked for Danamon Bank of Indonesia before moving in 2001 to the Universiti Teknologi Malaysia to continue his Master degree. He studied computer sains and completed his study in 2004. Now, he is a Ph.D student focus on Grid security field. At the same time, he was approved as Lecturer in Faculty of Engineering, University of Riau, Indonesia. His e-mail address is : `yoes_mantho@sigma-snt.com` and his Web-page can be found at `http://www.sigma-snt.com`.

ABDUL HANAN ABDULLAH, He is a Dean and Professor at Faculty of Computer Science & Information System, Universiti Teknologi Malaysia. He received his B.Sc and M.Sc in computer science from University of San Fransisco, California USA. He also graduated his Ph.D in the similar field (Computer Science) from Aston University, Birmingham. His email is : hanan@utm.my and and his Web-page can be found at http://www.csc.fsksm.utm.my/~hanan

An IDS for Web Applications

A. Biscotti, G. Capuzzi, E. Cardinale, L. Spalazzi
DIIGA — Università Politecnica delle Marche — I-60131 Ancona - Italy
e-mail: a.biscotti@univpm.it, {capuzzi, cardinale, spalazzi}@diiga.univpm.it

Abstract—**This work presents a WEB-IDS that combine both anomaly and misuse detection approach. This mixed solution is really interesting because merges the two complementary methods used to recognize attacks; we solved the usual conflicts presented by this choice and obtained an higher results accuracy. Our tool starts with the misuse-based module and its results are passed to the anomaly detection module: in this way the system has an high reactivity, less false negatives, it is simplier to solve conflicts between the two modules and the anomaly based module do not need to process dangerous events recognised by the first module. Our system does not need any specific setting, but only a training period. There are also different auto-setting tresholds for the different resources that reduce false alarms. The system is implemented as system service and tested with a real dataset by a services company.**

I. Introduction

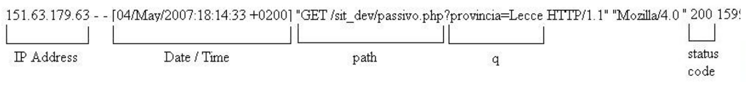
Web servers and web applications are often under attack by malicious software or intruders. The most part of web applications or web server-extentions are not structured with a secure criteria, so the number of attacks to them is increasing. Intrusion Detection Systems are provided with signatures to reveal attacks to this kind of applications; unfortunately is really difficult to mantaine updated the signatures because the high number of new vulnerability daily discovered. To avoid this problem, this kind of IDS shoud be complemented by anomaly detection systems that consent to discover attacks with unknow signature. In literature the possibility to combine misuse and anomaly detection was analyzed by E. Tombini, *et al.* [6], evidencing that in web application is better to have first the misuse module and then the anomaly one. The concept of anomaly detection was proposed for the first time by D.E. Denning [4], that presented an abstract model of a real-time intrusion detection system (IDES), based on the convinction that an use of the system different from the previous uses may be a signal of an improper use. A lot of other techniques were proposed to approach the anomaly intrusion detection by A.K. Ghosh, *et al.* [9] that used a neural network model to identify anomalous beaviors; L. Portnoy, *et al.* [12] proposed an unsupervisioned clustering algorithm to classify normal and abnormal activity; T. Lane and C.E. Brodley [13] presented a machine learning model based on IBL (instance-based learning) applyied to system applications. H. Feng, *et al.* [7] improved the static analysis for intrusion detection using PDA (push-down automaton). The limits of the state of the art is that all this works refers to system programs that are static: their beavior doesn't change to much working so is simply to identify anomalies in their working. In the case of web applications is different: we have dinamic entities and their behavior is strictly dependent from the interaction with the users; for this reason are interesting statistical and probability techniques proposed in their works about intrusion detection on web applications by C.Kruegel, *et al.* [15]. Cisco Systems [2] or Real Secure [11], implemented in their commercial systems solutions that combine anomaly and misuse detection, but they are principally misuse based. Therefore the target of this project is to improve web server and web applications security with many innovative algorithm modifications respect to common IPS, obtaining better performance in discovering attacks that exploit specific vulnerabilities for that applications, especially those developed in-house by service companies.With this paper we analyzed a possible combination of two different detection modules (misuse-based and anomaly-based) and our target was to determinate the capability and the efficiency of this combination to discover web-based attacks. Our attention is focused to obtain the less critical automatic solution of the conflicts between the two different evaluations on a same security event too, by developing an update engine which automatically adjust the sensitivity of the system to reduce the number of false alarms. Another objective was to study the possibility to add an efficient automatic prevention engine in order to decrease the quantity of critical data to manage for the system administrator. The basic problem of this solution is that web applications developed in-house by service companies does not provide a list of signatures, so it is necessary a certain time before the system can learn the attributes of applications and either a complete list of intrusive events or profiles of normal utilization of these applications have to be created, before the system achieves the best efficiency.

The paper is structured as follows: Section II provides an overview of the Detection Model; Section III, describes the Misuse detection Model; Section IV presents the evaluation tests of the system; Section V describes conclusions.

II. Overall Detection Model

Our system analyzes on-line a series of temporal subsequent HTTP requests as logged by most common web server (for example, Apache [8]). The analysis focuses on either GET or POST requests to HTML pages, server-side programs or active documents and consists of a serial combination of misuse and anomaly detection, with misuse detection first. More formally, the input to the process is a request R, extracted from web server access log file. A request R can be expressed in several ways depending of custom web server logging directives. In our work, we assumed that requests were logged in the Common Log Format (CLF) or Extended Common Log Format (ECLF). In these formats the most important features logged by web server are the IP source address, the date and the time of the request, the path to the desired web page

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)



```
151.63.179.63 - - [04/May/2007:18:14:33 +0200] "GET /sit_dev/passivo.php?provincia=Lecce HTTP/1.1" "Mozilla/4.0 " 200 159
```

IP Address
Date / Time
path
q
status code

Fig. 1. Sample web server access log entry

(*path*), and sometimes an optional query string (q). The query string is used to eventually pass parameters to the referenced script and it is identified by a leading ”?” character. A generic query string normally consists of an ordered list of n pairs of parameters with their corresponding values, so q can be expressed as $q = (p_1, v_1), (p_2, v_2), ..., (p_n, v_n)$ where $p_i \in P$, the set of all parameters and v_i is a string. Another important feature is the status code of the request. Figure 1 shows an example of a request logged by web server in ECLF format, in which elements used in the analysis are underlined. Let E be the event processed by the system (in this case the last line of the access log file). Additionally let M_i and M_u be the subsets of E that represent the sets of events declared respectively *intrusive* or *unknown* by the misuse detection component and let A_s and A_u be the subsets of E that represent the sets of events declared respectively *safe* or *unsafe* by the anomaly detection component. In this combination the misuse component tries to match E with a set of signatures and raises an alarm if the result of pattern matching is true (M_i). Events declared unknown in the first step (M_u) are then analyzed by an anomaly detection component. If the anomaly detector declares E safe, the event is assumed completely irrelevant (A_u) and is filtered out. On the contrary, an alarm is raised if E is declared intrusive (A_i). The detection process is shown in Figure 2. Events in the subset $M_u \cap A_u$ represent the set of events declared unsafe (potentially intrusive) by the anomaly detector and unknown (potentially safe) by the misuse detector. These events cannot be interpreted automatically because they should be considered as false positives issued by the anomaly detector or false negatives issued by misuse detector and the anomaly model (respectively the signatures database) must be updated. In this case the main task of system administrator is signaling what component gave wrong response, and an update engine automatically provides to update the anomaly detection component or add a new signature to the misuse detection component. At this time, in the first case the event tagged as normal is used to refine the anomaly model by computing a new anomaly threshold for the analyzed script, as shown more accurately in section about Anomaly detection Model. In the second case the automatic generation of a signature is made simply adding the tagged request at the end of the list of regular expressions. However in some cases, the system tries to automatically apply some prevention rules in order to block the communication with client who generated the anomaly event, with the help of a firewall. The prevention model is described in section Prevention Model.

III. Misuse Detection Models

Misuse detection Model: The misuse detection component is used to detect attacks embedded in URLs and report known malicious requests. For example, the presence in the request of the words ”SELECT” or ”WHERE” could be used to detect an attempted incorrect use of SQL commands to retrieve

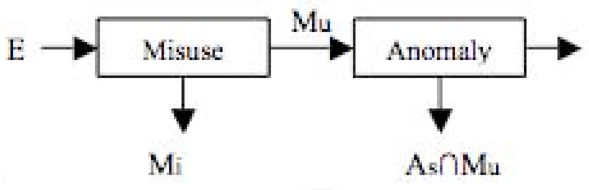


Fig. 2. System architecture

sensible data. The misuse detection process uses a list of regular expressions, which items represent manifestations of the most common attacks. These regular expression are used to match against the request analyzed and an alarm is raised when a match occurs. We used the Java regex package to implement the mechanism of pattern matching, which is in conformance with Level 1 of Unicode Technical Standard #18: Unicode Regular Expression Guidelines [3], plus RL2.1 Canonical Equivalents. Currently, the list contains about 70 regular expressions, but new attacks can be detected by adding new regular expressions. A new expression can be added manually without stopping system execution or it can be automatically generated and added by the system when a suspicious request is tagged as intrusive by system administrator. Regular expressions are compiled at runtime, before the matching process starts. The list of regular expressions is specified into a text file, in an XML-based format. An example is shown below.

```
<signatures>
  <expression name="Cross-site Scripting">.*(script)+.*
  </expression>
  <expression name="Directory Trasversal">.*(\.\./)+.*
  </expression>
<expression name "SQL Injection">.*
                (select|insert|update|delete|union|--)+.*
</expression>
[...]
</signatures>
```

At this time, the automatic generation of a signature is made simply adding the tagged request at the end of the list of regular expressions. *Anomaly detection Model*:the anomaly detection component provides a model of the ”normal behavior” of users and applications. The basic assumption is that, in case of intrusive actions, the behavior of users or applications differs substantially from normal behavior and this difference can be expressed quantitatively. The behavior of software entities in the web applications context is very dinamic and it is strictly connected with user interaction, which often governs the visualization process of data and informations through the choice of a set of values associated with some parameters. Additionally each web application is different from others in terms of number and type of parameters to elaborate. Our approach was initially based on the model proposed by C.Kruegel, *et al.* [15], which uses several different *evaluations* about requests, parameters and their relationship to detect anomalous entries. An evaluation is a set of statistical procedures used to evaluate a certain *feature* of a request. A feature can be related to a single parameter of a query string (e.g. the string length of a particular parameter value), to all parameters (e.g. the order of parameters in a query string), or to some relationship between a request and others related to a specific web page or script *r* (e.g. number of requests in a time slot). The model operates in two distinct steps: in a first step (*learning* or *tuning*) each evaluation is performed on a sufficiently large set of normal requests relating to the same web page or script (first 1000 in our implementation)and on their parameters in order to build a *profile* for each web page or script;for this purpose we used historical access log data files, gathered from the system administrator of a small Italian company . Afterward, a detection threshold is estabilished by evaluating requests separately. In the detection phase, the task of these evaluations is to assign a probability value to either a request as a whole or one of its parameters. This probability value reflects the probability of the occurrence of the given feature value with regards to an established profile. The assumption is that feature values with a sufficiently low probability can indicate a potential intrusive behavior. Based on the evaluation outputs, a request is either reported as a potential intrusive behavior or as normal. This decision is reached by calculating an *anomaly score*. A request is reported as anomalous if this anomaly score is above the corresponding detection threshold. The anomaly score value is calculated using a weighted sum as shown in Eq.(1). In this equation, w_e represents the weight associated with evaluation *e*, while p_e is its returned probability value. The probability p_e is subtracted from 1 because a value close to zero indicates a possible anomalous event. In this case it should yield a relative high anomaly score. The w_e values are estabilished a priori and they can be adjusted regarding the analyzed application, web page or script, after a brief analysis process on historical data of web server.

$$AS = \sum_{e \in Evaluations} w_e * (1 - p_e) \tag{1}$$

Our evaluation procedures are very similar to those proposed by Kruegel et al., but we used only six anomaly evaluations to perform detection process and we refined evaluation procedures in some points. The evaluations used are enumerated below:
Length of values associated with parameters;
Distribution of characters in values associated with parameters;
Presence of limited set of values associated with parameters;
Presence or absence of a parameter in a request;
Order of parameters in a request;
Access frequency to a web page or script;
When a certain request contains several parameters, we chose to consider the lower probability value returned by our anomaly evaluations, in order to avoid a malicious user to hide a single invalid input into a series of valid inputs. Additionally, when a script without parameters to elaborate is analyzed, only two evaluations are performed: the check of presence or absence of parameters in the request and the access frequency to the script.
Length of values associated with parameters:The length of a value associated with a parameter of a request can be used to detect anomalous requests, especially when these values are either fixed-size tokens or short strings derived from user input (such as fields in a form). For example, to overflow a buffer in a target application, it is necessary to send a large amount of data, depending on the length of the buffer. In the training phase the goal of this evaluation is to approximate the actual but unknown distribution of the parameter lengths, so in the detection phase it can detect instances that significantly deviate from the observed normal behavior. Clearly, the probability density function of the underlying real distribution often doesn't follow a smooth curve. Additionally the distribution has a large variance in several cases. Nevertheless, the evaluation is able to efficiently identify significant deviations.
LEARNING:in the learning phase, the length of values associated with a certain parameter in a request can be expressed as a random variable, and it is possible to calculate mean μ and variance σ^2 associated with it. The mean and the variance of the real parameter values length distribution are so approximated by calculating the sample mean and the sample variance for the lengths $l_1, l_2, \ldots, l_n$ of the values associated with the analyzed parameter and processed during the learning phase (assuming that *n* requests with that parameter were processed). The cost of this evaluation is proportional to the number *n* of queries analyzed during the learning phase, followed by a constant cost to compute the mean and the variance.
DETECTION:In the detection phase we already know the estimated parameter length distribution, in particular μ and σ^2. The task of this phase is to evaluate the anomaly of a value associated with the analyzed parameter with length *l*. We assume that a value with length less than μ, relating to a specific parameter, is associated with a probability value of 1. This is due the fact we consider this evaluation able to detect anomalous requests in which a large amount of data is injected in one or some values. To evaluate the anomaly of a string with length *l* higher than μ, we quantify the "distance" of the length l from the mean value μ with the help of the Cantelli inequality [5]. The Cantelli inequality is an efficient metric to model decreasing probabilities for strings with lengths that increasingly exceed the mean. It puts, for an arbitrary distribution with mean μ and variance σ^2, an upper bound on the probability that a generic length *x* is higher than *l*, as shown in Equation 2. When *l* is very distant from μ then the probability value associated with a string having a greater length than *l* should decrease. In this case an attacker cannot insert malicious input by padding the string and increasing its length, because an increase in length reduce the probability value associated with that string.

$$p(l) = \begin{cases} 1 & l \leq \mu \\ \frac{\sigma^2}{\sigma^2 + (l-\mu)^2} & l > \mu \end{cases} \tag{2}$$

Distribution of characters in values associated with parameters When we analyze a request, sometimes we are able to detect an anomalous value associated with a certain parameter by looking at its distribution of characters. This is due the fact that often user input have a regular structure, is mostly human-readable, and almost always contain only printable characters. Additionally, a large percentage of characters in regular values are drawn mainly from letters, numbers, and sometimes a few special characters. However, there are some cases in which a malicious user sends binary data (e.g., buffer overflow attacks), or inject a string with a noticeable strange character sequence (e.g., with many repetition of the dot character in directory traversal exploits). Anyway, when we analyze a set of several requests containing same parameters, similarities often can be observed between the character frequencies of values associated with a certain parameter. In fact, if we consider a string drawn from Italian or English language, we observe that there are words in which some characters are more frequent than others but often there is no one that is clearly more prevalent than others. So, if we sort in descending order the relative frequencies for all possible characters in a legitimate string, one can expect that the relative frequencies slowly decrease in value. In case of malicious input, instead, these frequencies often drop extremely fast. This can be the result of a certain padding character that is repeated many times in a buffer overflow attack or because of the many occurrences of the dot character in a directory traversal attempt. Therefore we assume that the character distribution of a value is represented by the array of its relative frequencies sorted in descending order.

LEARNING the goal of the learning phase is to build a profile of the normal character distribution of a generic value associated with a certain parameter, for each parameter of a script. For this purpose, the character distribution for each observed parameter is stored. Then an "idealized" character distribution is approximated by setting, for each index *i* of the relative frequencies array, the average of all *n* values of stored character distributions that are in the position *i*, assuming that *n* requests with that parameter are analyzed. Because all individual character distributions sum up to unity, their average will do so as well, and the idealized character distribution is well-defined. The cost of building this model is linear in the number of parameters that are analyzed during this phase. For each parameter, the character distribution has to be determined, an operation which has a cost that is proportional to the length of its value string.

DETECTION: the task of the detection phase is to determine the probability that the character distribution of a certain value is conforming with the modelized legitimate character distribution associated with its reference parameter. In this case, more precisely, we want to check if this array is a sample drawn from a population with a certain distribution, that can be represented by the idealized character distribution of its reference parameter. This probability, or more precisely, the confidence in this hypothesis is calculated by the Pearson χ^2 statistical test[5]. We chose to group the values of the array in six intervals defined as follows:[0], [1, 3], [4, 6], [7, 11], [12, 15], [16, 255], assuming that possible characters are drawn from a subset of 256 character (ASCII - 8 bit) and reflecting the fact that the relative frequencies are sorted in descending order, so values in position *i* are higher when *i* is small. Additionally, the value representative of an interval is the average of values between the considered indexes. When a new parameter is analyzed, the number of occurrences of each character in the string value is determined. Afterward, the values are sorted in descending order and combined by aggregating values that belong to the same interval *i*. The resultant value is indicated as O_i. The χ^2 value is so determined as shown in Equation 3. In this equation E_i represents the expected value and it is calculated by multiplying the value of the interval i for the length of the string analyzed.

$$\chi^2 = \sum_{i=0}^{i<6} \frac{O_i - E_i}{E_i} \tag{3}$$

The χ^2 test is then used to calculate the probability that the given sample has been drawn from the legitimate character distribution. The actual probability p is read from a predefined table (Table I) using the χ^2 value and the degrees of freedom as index, observing that in this test the degrees of freedom are calculated as $(numberofintervals - 1)$. The derived value p is used as the return value for this model. When the probability that the sample is drawn from the legitimate character distribution increases, p increases as well. The cost of evaluate an input string using this method consists of the calculation of the χ^2 value[].

TABLE I
PROBABILITY VALUES FOR 5 DEGREES OF FREEDOM

χ^2	p	χ^2	p
0.41	0.995	9.24	0.1
0.55	0.99	11.07	0.05
0.83	0.975	12.83	0.025
1.15	0.95	15.09	0.01
1.61	0.9	16.75	0.005

Presence of limited set of values associated with parameters:the purpose of this evaluation is to determine whether the values of a certain parameter are drawn from a limited set of possible alternatives (i.e., "checkbox", "radio" or "select" HTML input types). When a malicious user attempts to use these parameters to pass to the analyzed script illegal values or values that are not in the trusted set, the intrusion attempt can be detected. When a set of possible alternatives can be identified for a certain parameter, it is assumed that the parameter values are random and no attacks can be detected by this evaluation.

LEARNING:the goal of this phase is to model when the values associated with a certain parameter are random or part of an enumeration. When a set of different occurrences of the analyzed parameter values are observed, one can see that in the case of an enumeration, the number of different values encountered does not exceed a certain unknown threshold *t*. Instead, when the number of different parameter values grows proportional to the total number of analyzed instances, the use of random values is indicated. It is consequently necessary to estabilish the correlation between the total number of analyzed values and the total number of different values encountered during this phase. More formally, to decide if a parameter is associated with an enumeration, we calculate the statistical correlation ρ between the values of the functions f and g, defined $\forall x \in \{1, 2, ..., n\}$ as follows on N_0, where n is the total number of values encountered during this phase, and $S^{(x)}$ represents the set of values encountered at the time the x^{th}

value is analyzed:

$$f(x) = x \tag{4}$$

$$g(x) = \begin{cases} g(x-1)+1 & x^{th} value \in S^{(x)} \\ g(x-1)-1 & x^{th} value \notin S^{(x)} \\ 0 & x = 0 \end{cases} \tag{5}$$

The correlation parameter ρ is derived after the training data has been processed. It is calculated from *f* and g with their respective variances Var(*f*), Var(*g*) and the covariance Covar(*f*,*g*) as shown below:

$$\rho = \frac{Covar(f,g)}{\sqrt{Var(f) * Var(g)}} \tag{6}$$

If ρ is less than 0, then *f* and *g* are negatively correlated and an enumeration is assumed, reflecting the fact that increasing the value of observed values, the number of different occurrences has not shown a proportional increase too. This means that some values was encountered several times during the training phase. In the opposite case, where ρ is greater than 0, the values observed have shown sufficient variation to support the hypothesis that they are not drawn from a small set of predefined values. Naturally, when an enumeration is assumed, the complete set of values encountered is stored for use in the detection phase. The cost of building this model is strictly connected to the calculation of the covariance between these two simple functions. This cost depends on the number of analyzed requests.

DETECTION:once it has been determined that the values of a certain parameter of a script are tokens drawn from an enumeration, any new value *v* is expected to appear in the set *S* of known values. When this happens, a probability value of 1 is returned. If the value is not in the established set of values, a probability value of 0 is returned, as shown in Equation 7. If it has been determined that the parameter values are random, the model always returns 1. In order to increase the efficiency of the detection we use the Java HashMap data type to store and retrieve values.

$$p(v) = \begin{cases} 1 & v \in S \\ 0 & v \notin S \end{cases} \tag{7}$$

Presence or absence of a parameter in a request: Most of the time, server-side programs or scripts are not directly invoked by users typing the input parameters into the URIs themselves. Instead, client-side programs or scripts pre-process the data and transform it into a suitable request. This usually results in a high regularity in the number, name, and order of parameters. Another interesting situation is when developers with little expertise use *"hidden"* type forms in their HTML pages and then process the data in a separate script. In this case this type of parameters does not appear in requests logged by the web server, so its presence can indicate the attempt of an intrusion. The analysis performed by this evaluation takes advantage of these facts and tries to detect requests that deviate from the estabilished profile, built in the training phase. This evaluation described in this section, deals with the presence and absence of parameters p_i in a query string q and consider a query as a whole, differing from previous ones, which focus on features of individual query parameters. This approach assumes that the absence or abnormal presence of one or more parameters in a query might indicate malicious behavior. This allows for the detection of intrusion attempts where server-side applications or scripts are probed or exploited by sending malformed requests.

LEARNING: the goal of this phase is to estabilish when a script is associated with some parameters and, in the positive case, to create a model of acceptable subsets of parameters that appear simultaneously in a query string. This is done simply by storing each distinct subset $S_q = \{p_i, ..., p_k\}$ of parameters seen during the training phase.

DETECTION: the detection phase have to deal with three different situations: *Scripts without visible parameters* - In this case the evaluation is performed observing if the analyzed request has one or more parameters in its URI. In this case a probability value of 0 is returned, 1 otherwise; *Scripts with visible parameters* - In this case, for each request analyzed, the current parameter set is extracted. When the observed set of parameters has been encountered during the training phase, 1 is returned, otherwise 0. *Generic scripts* - This situation deals with scripts that can appear with or without parameters. In this case the detection process is performed with the method described in the situation 2, when a set of parameters can be extracted from the analyzed request. Otherwise a probability value of 1 is returned. The current script type is determined by reading a code of two boolean values, set in the training phase, where *11* is a generic script, *10* is a script without visible parameters, *01* is a script with visible parameters and *00* is for the Unknown.

Order of parameters in a request:As discussed in the previous section, legitimate requests often contain the same parameters in the same order. This is usually not the case for hand-crafted requests, as the order chosen by a malicious user can be arbitrary and has no influence on the execution of the program. The goal of this evaluation is to determine whether the given order of parameters is consistent with a profile built during the learning phase.

LEARNING The order constraints between all *k* parameters of a legitimate query string are determined during this phase. It is consequently necessary to estabilish an order relationship between parameters. So we assume that a parameter p_t of a script precedes another parameter p_s when p_t and p_s appear together in the parameter subset of at least one query string and p_t comes before p_s in the ordered list of parameters of all queries where they appear together. In order to store these relationships between parameters we use a matrix *M* of (*s* x *s*) elements, where *s* is the total number of parameters associated with the analyzed script and $M_{ij} = 1$ if p_i precedes p_j, 0 otherwise. So, for every query string q_i, with *i* = 1,. . . ,*n*, that is analyzed during the training period, the ordered list of its parameters $p_1, p_2, ..., p_i$ is processed. For each attribute pair (p_t, p_s) in this list, with $t \neq s$, the M_{ts} value is set to 1. In this phase all analyzed requests are assumed to be normal, so the final result is that $M_{ts} \neq M_{st}$ when p_t precedes p_s, assuming that parameters can only appear in the same order. If parameters p_t and p_s are admitted in both different orders, then $M_{ts} = M_{st}$. The cost of building this profile is not high, because the total number of parameters processed by a generic script is usually relatively small.

DETECTION The detection process checks whether the parameters of a query string satisfy the order constraints determined during the learning phase. Given a query string with parameters $p_1, p_2, ..., p_i$ and the matrix *M*, all the parameter pairs (p_j, p_k), with $j \neq k$, are analyzed to detect potential violations. A violation occurs when for any single pair (p_j, p_k) of the current query string, where p_j precedes p_k, the corrisponding M_{jk} value is 0. In this case the evaluation returns a probability value of 0, otherwise it returns 1.

Access frequency to a web page or script:different server-side applications or web scripts normally are invoked with different frequencies. However, after monitoring a specific web page or script in a sufficiently long time interval, one can often observe that the general access patterns remain relatively constant. It is possible to distinguish between two types of access frequencies for each web page or script. One is the frequency of the application being accessed from a certain client (based on the IP address), the other is the total frequency of all accesses. When a malicious user attempts a DoS (Denial of Service) exploit for a certain script, the number of accesses observed in a small time interval from that client can increase drastically. Otherwise the frequency of all accesses can grow extremely in case of DDos (Distributed DoS) attempts. Changes in access patterns can indicate intrusion attempts (e.g., when an application is usually accessed infrequently but is suddenly exposed to a burst of invocations). This increase could be also the result of an attacker probing for vulnerabilities or trying to guess parameter values. A single determined attacker can evade detection by executing his actions slowly, but often most intruders use tools that execute brute force attacks, raising the total access frequency to a suspicious level.

LEARNING:the objective of this phase is to build a model of normal access frequency pattern for a web page or script. To determine the expected normal access frequencies, the time period between the first and the last request in the training data set is divided into consecutive time intervals of a fixed size (60 seconds in our implementation). Then, the total number of requests and the numbers of requests from distinct clients (distinct IP addresses) are counted in each of these intervals. The counts for the total accesses and the counts for the accesses from distinct clients can be considered as two random variables, which respective means and variances can be evaluated. These values can represent the normal web page or script behavior in terms of number of requests made to it. The cost of building this profile is proportional to the number of requests that are analyzed during the training period.

DETECTION:this evaluation focuses on whole sequences of queries, so it is necessary to maintain data of recent accesses to the analyzed script for a certain time interval. The main goal is to be able to detect vulnerability probing, parameter value guessing and DoS attempts. So during detection, time is divided into intervals of the same fixed size that was used during the learning phase. When a request is evaluated, the number of total requests n_1 and the number of requests from this client n_2, both in the current time interval, are determined. Similar to the attribute length evaluation, the Cantelli inequality is used to calculate the probability of n_1, given the mean and the variance of the total access frequencies, and the probability of n_2, given the mean and the variance of access frequencies from distinct clients. This two probabilities are then combined in a weighted sum, as shown below, and returned by this evaluation.

$$p(a) = \sum_{i=1}^{2} w_i * p(n_i) \tag{8}$$

The w_i values are initially set to $1/i$, but they can adjusted by the system administrator taking in account the specific web page or script type. The detection cost is proportional to the number of requests analyzed during the current detection interval. This module can be linked to a prevention module: we did it, but we do not talk about it in this paper.

IV. EVALUATION

This section describes the approach used in order to evaluate the intrusion detection system proposed. The evaluation was performed by gathering on-line (in two different experiment) real data from the main web server of an Italian company, SIMobile s.r.l.(www.simobile.it). The IDS was configured and integrated with the web architecture (e.g., operating system, web server, DBMS), by installing it as a service.

TUNING PHASE: the system tuning phase was performed off-line, analyzing a stored database of historical legitimate HTTP/HTTPS requests. So we usedthem to model the normal system behavior. Table II, shows relevant informations about the learning set, like data gathering time interval, total number of analyzed requests and total number of pages or scripts modeled.

DETECTION:there are several indexes to evaluate the effectiveness of an intrusion detection system. In our work, we used a set of typical indexes often used in diagnostic tests, as shown below.

TABLE II
LEARNING PHASE INFORMATIONS

Time interval	Log size	# of requests	# of modeled scripts
165 days	16 MByte	87284	260

Sensitivity (Sen)= $\frac{TP}{TP+FN}$,
Specificity (Spe)= $\frac{TN}{TN+FP}$,
Positive predictive value (PPV) = $\frac{TP}{TP+FP}$,
Negative predictive value (NPV) = $\frac{TN}{TN+FN}$,
Prevalence (Prev)= $\frac{TP+FN}{TP+FN+FP+TN}$,

Where TP,FN,FP,FN are respectively: true positive, false negative,false positive and false negative. The test has an high degree of effectiveness when sensitivity and specificity values are close to 1. Other two fundamental indexes are related to false alarms, as shown below.

False Positive Rate (FPR) = $\frac{FP}{TN+FP} = 1 - \textit{specificity}$,
False Negative Rate (FNR) = $\frac{FN}{FN+TP} = 1 - \textit{sensitivity}$.

Obviously, the test has an high degree of effectiveness when the false positive/negative rate is close to 0. The system has been evaluated by analyzing on-line the HTTP/HTTPS traffic towards the monitored web server. In our experiment the weight w_e of Equation (1) has been set to $1/6 = 0,167$ and the weight w_i of Equation (8) has been set to $1/2 = 0,5$. Table III reports the results of one of the two experiments.

TABLE III
EVALUATIONS RESULTS

Monitoring days	55	alerts	28	Sen	1
Request logged	18894	TP	17	Spe	0,997
Suspicious events	28	TN	3655	FPR	0,003
Intrusive events	17	FP	11	FNR	0
		FN	0	PPV	0,607
		Prev	0,0046	NPV	1

We have a false negative when the system fails to identify a potentially intrusive behavior; therefore, we can not compute the number of false negatives with an automatic procedure. The number reported in Table III has been manually computed by the system administrator day by day. His task consisted in checking daily requests (around 343 requests per day) and identifying potential intrusions not signaled by the system.
The reliability of the evaluation depends on two factors: the total number of analyzed requests and the comparison with the related works. The total number of analyzed request is related to the average server traffic load of the monitored company, but however the system shows that results are comparable with the main reference work [15]. Anyway we expect a sensible improvement in reducing false positives when automatically updating the misuse engine with new specific signatures and the anomaly engine with adjusted thresholds after a longer monitoring period.

V. Conclusions

In this work we considered the security problem of web-applications and the application of Intrusion Detection Systems to this kind of systems.We proposed an intrusion detection model that improves the previous model proposed by [15]. I this model we combined an anomaly detection approach with a misuse detection approach. Indeed, the best way to reveal web application attacks is to use the precision of signature based systems with the flexibility of anomaly detection systems and to solve problems coming from the combination of two approaches. About anomaly detection we obtained a great advantage combining different evaluation systems to cover the great number of attack typologies. The model proposed doesn't need any specific configuration, but only a training period. We implemented this model in a IDS that we experimented in a real context.

TABLE IV
IPS Comparison

Sistem	Sensitivity	FP/ # logs	DATA
Our IDS	100%	0,02% - 0,06 %	Real Data
[15]	100%	0,002% - 1,45%	Simulated
[14]	-	0,069%	Real Data

Table IV shows that our results are comparable with the main reference work [15] in terms of false positive , with the difference that while their results came from a simulation, we applied our IDS to a real company network. Furthermore, the Positive Predictive Value is about 60 %, that is a tipical value of PPV for IDS/IPS, as described in [10] and [1]. False positives in particular, occured in the first days of monitoring and then decreased, as consequence of adjusting detection thresholds.

References

[1] J.S. Baras. A.Cardenas and K. Seamon. A framework for the evaluation of intrusion detection systems. In *IEEE Symposium on Security and Privacy*.

[2] Cisco. Cisco intrusion prevention system. Technical report, http://www.cisco.com/en/US/products/sw/secursw/ps2113/index.html.

[3] Mark Davis. Unicode technical standard 18, unicode regular expressions. Technical report, http://www.unicode.org/unicode/reports/tr18.

[4] D.E. Denning. An intrusion detection model. volume 2, pages 222–232, 1987.

[5] Luc Devroye. In *Non-Uniform Random Variate Generation*, 1986. Springer-Verlag, New York.

[6] L. Me E. Tombini, H. Debar and M. Ducasse. A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic. December 2004.

[7] H. Feng, J. Giffin, Y. Huang, S. Jha, W. Lee, and B. Miller. Formalizing sensitivity in static analysis for intrusion detection. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA,*, 2004.

[8] Apache Software Foundations. Apache http server log files. Technical report, http://httpd.apache.org/docs/2.2/logs.html.

[9] A.K. Ghosh, J. Wanken, and F. Charron. Detecting anomalous and unknown intrusions against programs. volume AZ, pages 259–267, December 1998.

[10] David Dagon Wanke Lee Guofei Gu, Prahlad Fogla. In *Measuring Intrusion Detection Capability: An Informmation-Theoretic Approac.*, March 2006. In Proceedings of ACM Symposium of InformAction, Computer and Communications Security(ASIACCS06).

[11] IBM. Ibm internet security sytems proventia network intrusion preventionsystem. Technical report, http://www.iss.net/products/product_ sections/Intrusion_ Prevention.html.

[12] S. Stolfo L. Portnoy, E. Eskin. Intrusion detection with unlabeled data using clustering. Novembre 2001.

[13] T. Lane and C.E. Brodley. Temporal sequence learning and data reduction foranomaly detection. pages 150–158. ACM Press, 1998.

[14] M. Dacier M. Almgren, H. Debar. A lightweight tool for detecting web server attacks. In *ISOC Symposium on Network and Distributed Systems Security*.

[15] W. Robertson, G. Vigna, C. Kruegel, and R. Kemmerer. Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks. In *Proceeding of the Network and Distributed System Security (NDSS) Symposium San Diego, CA*, 2006.

Efficient Virus Detection Using Dynamic Instruction Sequences

Jianyong Dai, Ratan Guha and Joohan Lee
School of Electrical Engineering and Computer Science
University of Central Florida
4000 Central Florida Blvd, Orlando, Florida, 32816
E-mail: {daijy, guha, jlee} @cs.ucf.edu

KEYWORDS
Virus Detection, Data Mining

ABSTRACT

In this paper, we present a novel approach to detect unknown virus using dynamic instruction sequences mining techniques. We collect runtime instruction sequences from unknown executables and organize instruction sequences into basic blocks. We extract instruction sequence patterns based on three types of instruction associations within derived basic blocks. Following a data mining process, we perform feature extraction, feature selection and then build a classification model to learn instruction association patterns from both benign and malicious dataset automatically. By applying this classification model, we can predict the nature of an unknown program. Our result shows that our approach is accurate, reliable and efficient.

INTRODUCTION

Malicious software is becoming a major threat to the computer world. The general availability of the malicious software programming skill and malicious code authoring tools makes it easier to build new malicious codes. Recent statistics for Windows Malicious Software Removal Tool (MSRT) by Microsoft shows that about 0.46% of computers are infected by one or more malicious codes and this number is keep increasing [1]. Moreover, the advent of more sophisticated virus writing techniques such as polymorphism [2] and metamorphism [3] makes it even harder to detect a virus.

The prevailing technique in the antivirus industry is based on signature matching. The detection mechanism searches for a signature pattern that identifies a particular virus or strain of viruses. Though accurate in detecting known viruses, the technique falls short for detecting new or unknown viruses for which no identifying pattern is present. Whenever a new virus comes into the wild, virus experts extract identifying byte sequences of that virus either manually or automatically [4], then deliver the fingerprint of the new virus through an automatic update process. The end user will finally get the fingerprint and be able to scan for the new viruses.

However, zero-day attacks are not uncommon these days [34]. These zero-day viruses propagate really fast and cause catastrophic damage to the computers before the new identifying fingerprint is distributed [5].

Several approaches have been proposed to detect unknown virus without signatures. These approaches can be further divided into two categories: static approaches and dynamic approaches. Static approaches check executable binary or assembly code derived from the executables without executing it. Detecting virus by binary code is semantic unaware and may not capture the nature of virus code. Static approaches based on assembly code seems to be promising, however, deriving assembly code from an executable itself is a hard problem. We find that approximately 90% of virus binary code cannot be fully disassembled by state of the art disassembler. Dynamic approaches run the executables inside an isolated environment and capture the runtime behavior. Most existing dynamic approaches are based on system calls made by the unknown executable at runtime. The idea behind is that viral behavior of a malicious code is revealed by system calls. However, some malicious code will not reveal itself by making such system calls in every invocation of the virus code. On the other hand, some malicious behaviors such as self-modifying are not revealed through system calls. Based on these observations, we propose to use dynamic instruction sequences instead of system calls to detect virus dynamically.

Instead of manually analyzing captured runtime trace of every unknown executable, some people designed some automatic mechanisms. The obvious approach is to derive heuristic rules based on expert knowledge. However, this approach is time consuming and easier to be evaded by the virus writer. The other approach is data mining. Here data mining refers to a classification problem to determine whether a program can be classified into either malicious or benign.

The key problem for this classification problem is how to extract features from captured runtime instruction sequences. We believe the way how instructions group together capture the nature of malicious behavior. To this end, we devise a notion "instruction association".

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

In the first step, we organize instructions into logic assembly. Logic assembly is a reconstructed program assembly using available runtime instruction sequences. It may have incomplete code coverage, but logic assembly will keep the structure of the executable code as much as possible. Another merit of logic assembly is that we can deal with self-modifying code during the process of logic assembly construction.

The second step is to extract frequent instruction group inside basic block inside logic assembly. We call these instruction groups "instruction associations". We use three variations of instruction associations. First, we consider the exact consecutive order of instructions in a block. Second, we consider the order of the instructions in a block but not necessarily consecutive. The third is the instruction association that observes which instructions appear together in a block but does not consider the order.

We use the frequency of instruction association as features of our dataset. We then build classification models based on the dataset.

While accuracy is the main focus for virus detection, efficiency is another concern. No matter how accurate the detection mechanism is, if it takes long time to determine if an executable is a virus or not, it is not useful in practice as well. Our analysis shows that compare to system calls, our approach takes less time to collect enough data for the classification model, and the processing time is affordable.

RELATED RESEARCH

Although the problem of determining whether unknown program is malicious or not has been proven to be generally undecidable [6], detecting viruses with an acceptable detecting rate is still possible. A number of approaches have been proposed to detect unknown viruses.

Static approaches check executable binaries or assembly code without actually executing the unknown program. The work of Arnold et al [7] uses binary trigram as their detecting criteria. They use neural network as their classifier and reported a good result in detecting boot sector viruses for a small sample size.

InSeon et al [8] also use binary sequences as features. However, they construct a self organizing map on top of these features. Self organizing map converts binary sequences into a two dimensional map. They claim that malicious viruses from the same virus family demonstrate same characteristic in the resulting graph. But they do not give a quantitative way to differentiate a virus from benign code.

Schultz et al [9] use comprehensive features in their classifiers. They use three groups of features. The first group is the list of DLLs and DLL function calls used by the binary. The second group is string information acquired from GNU strings utility. The third group is a simple binary sequence feature. They conduct experimentation using numerous classifiers such as RIPPER, Naïve Bayes, Multi-Naïve Bayes.

In recent years, researchers start to explore the possibility to use N-Gram in detecting computer viruses [10, 11, 12]. Here N-Gram refers to consecutive binary sequences of fixed size inside binary code.

Kolter et al [12] extract all N-Gram from training set and then perform a feature selection process based on information gain. Top 500 N-Gram features are selected. Then, they mark the presence of each N-Gram in the training dataset. These binary tabular data are used as the input data for numerous classifiers. They experimented with Instance-based Learner, TFIDF classifier, Naïve Bayes, Support Vector Machines, Decision Trees and Boosted Classifiers. Instead of accuracy, they only reported AUC (Areas Under Curves). The best result is achieved by boosted J48 at AUC, 0.996.

Although above approaches show satisfactory results, these detection techniques are limited in that they do not distinguish the instructions from data and are blind to the structure of the program which carries important information to understand its behavior. We redo the expe-riment mentioned in [12] and we find that the key contributors that lead to the classifications are not from bytes which representing virus code, rather, they are from structural binary or string constants. Since structural binary and string constants are not essential components to a virus, this suggests that those detection mechanisms can be evaded easily.

Another area of current researches focuses on higher level features based on assembly code.

Sung A.H.et al [13] proposes to collect API call sequences from assembly code and compare these sequences to known malicious API call sequences.

Mihai et al [14] uses template matching against assembly code to detect known malicious behavior such as self-modification.

In [15], the author proposes to use control graph extracted from assembly code and then use graph comparing algorithm to match it against known virus control graphs.

These approaches seem to be promising. The problem is that disassembling executable code itself is a hard problem [16, 17, 18].

Besides static analysis, runtime features have also been used in virus research. Most of current approaches are based on system calls collected from runtime trace.

TTAnalyze [19] is a tool to executing an unknown executable inside a virtual machine, capture all system calls and parameters of each system call to a log file. An expert can check the log file to find any malicious behavior ma-nually.

Steven A. Hofmeyr et al [20] proposes one of the very first data mining approaches using dynamic system call. They build an N-Gram system call sequences database for benign programs. For every unknown executable, they obtain system call sequences N-Grams and compare it with the database, if they cannot find a similar N-Gram, then the detection system triggers alert. In [21], the author proposes several data mining approaches based on system calls N-Gram as well. They try to build a benign and malicious system call N-Gram database, and obtain rules from this database. For unknown system call trace, they calculate a score based on the count of benign N-Gram and malicious N-Gram. In the same paper, they also propose an approach to use first (n-1) system calls inside N-Gram as features to predict the nth system call. The average violation score determines the nature of the unknown executable.

In [22], the author compares three approaches based on simple N-Gram frequency, data mining and hidden Markov model (HMM) approach, and conclude that though HMM is slow, it usually leads to the most accuracy model.

In [23], the author runs viruses executables inside a virtual machine, collecting operating system call sequences of that program. The author intends to cluster viruses into groups. The author uses k-medoid clustering algorithm to group the viruses, and uses Levenshtein distance to calculate the distance between operating system call sequences of different runtime traces.

LOGIC ASSEMBLY

In this paper, we propose to use instruction sequences captured at runtime as our source to build classification models.

In order to capture runtime instruction sequences, we execute binary code inside OllyDbg [24]. OllyDbg has the functionality to log each instruction along with its virtual memory address when executing. OllyDbg logs in-struction in the form of assembly code. Because virus codes are destructive, we execute virus code and OllyDbg inside a virtual machine. Every time we finish running a virus code, we reset the disk image of the virtual machine.

OllyDbg captures execution log at a rate around 6,000 instructions per second in our computer. For some executable requires interaction, we use the most straightforward way, such as typing “enter” key in a command line application or press “Ok” button in a GUI application to respond.

In a conventional disassembler, assembly instructions are organized into basic blocks. A basic block is a sequence of instructions without any jump targets in the middle. Usually disassembler will generate a label for each basic block automatically. However, execution log generated by OllyDbg is simply a chronological record of all instructions executed. The instructions do not group into basic blocks and there is no labels. We believe that basic block capture the structure of instruction sequences and thus we process the instruction traces and organize them into basic blocks. We call the resulting assembly code “logic assembly”.
Compared with static disassembler, dynamic captured instruction sequences may have incomplete code coverage. This fact implies the following consequences about logic assembly code:

1. Some basic blocks may be completely missing

2. Some basic blocks may contains less instructions

3. Some jump targets may be missing, that makes two basic blocks merge together

Despite these differences, logic assembly carries as much structural information of a program as possible.
We design the algorithm to construct logic assembly from runtime instructions trace. The algorithm consists of three steps and we describe below:

1. Sort all instructions in the execution log on their virtual addresses. Repeated code fragments are ignored.

2. Scan all jump instructions. If it is a control flow transfer instruction (conditional or unconditional), we mark it as the beginning of a new basic block.

3. Output all instruction sequences in order along with labels

Each assembly instruction usually consists of opera-tion code (opcode) and operands. Some instruction set such as 80x86 also have instruction prefix to represent repetition, condition, etc. We pay attention to the opcode and ignore the operands and prefix since the opcode represents the behavior of the program. The resulting assembly code is called abstract assembly [25].
Figure 1 shows an example of logic assembly and abstract assembly construction. Figure 1.a is the original instruction sequences captured by OllyDbg. We remove duplicated code from line 7 to line 14, and generate label for jump destination line 3. Figure 1.b is the logic

assembly we generated. We further omit the operands and keep opcode, and we finally get abstract assembly Figure 1.c.

```
1. 01002157    pop ecx
2. 01002158    lea ecx, ds:[eax+1]
3. 0100215b    mov dl, ds:[eax]
4. 0100215d    inc eax
5. 0100215e    test dl,dl
6. 01002160    jnz short 0100215b
7. 0100215b    mov dl, ds:[eax]    ┐
8.  0100215d   inc eax             ├ Repetition
9.  0100215e   test dl,dl          │
10.01002160    jnz short 0100215b  ┘
11.0100215b    mov dl, ds:[eax]    ┐
12.0100215d    inc eax             ├ Repetition
13.0100215e    test dl,dl          │
14.01002160    jnz short 0100215b  ┘
```

a. Original Log

```
01002157    loc1    pop ecx
01002158            lea ecx,dword ptr ds:[eax+1]
0100215b    loc2    mov dl,byte ptr ds:[eax]
0100215d            inc eax
0100215e            test dl,dl
01002160            inz short 0100215b
```

b. Logic Assembly

```
loc1        pop lea
loc2        mov inc test jnz
```

c. Abstract Assembly

Figure 1 Logic Assembly and Abstract Assembly

One merit of dynamic instruction sequences over assembly is that dynamic instruction sequences expose some type of self-modifying behavior. If a program modifies its code at runtime, we may observe two different instructions at the same virtual address in runtime trace. A program may modify its own code more than once. We devise a mechanism to capture this behavior while constructing logic assembly.

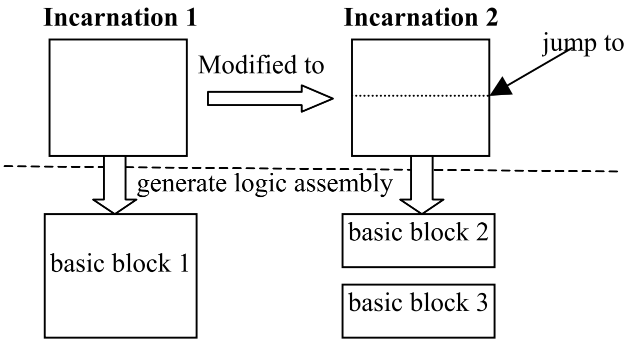


Figure 2 Different Incarnations

We associate an incarnation number with each virtual address we have seen in the dynamic instruction sequences. Initial incarnation number is 1. Each time we met an instruction at the same virtual address, we compare this assembly instruction with the one we have seen before at that virtual address, if the instruction changes, we increate the incarnation number. Subsequent jump instruction will mark the beginning of a basic block on the newest incarnation. We treat instructions of different incarnation as different code segment, and generate basic blocks separately. Figure 2 illustrate this process.

In this way we keep the behavior of any historical invocations even the code is later overwrote by newly generated code.

INSTRUCTION ASSOCIATIONS

Once we get abstract assembly, we are interested in finding relationship among instructions within each basic block. We believe the way instruction sequences groups together within each block carries the information of the behavior of an executable.

The instruction sequences we are interested in are not limited to consecutive and ordered sequences. Virus writers frequently change the order of instructions and insert irrelevant instructions manually to create a new virus variation. Further, metamorphism viruses [3] make this process automatic. The resulting virus variation still carries the malicious behavior. However, any detection mechanism based on consecutive and ordered sequences such as N-Gram could be fooled.

We have two considerations to obtain the relationship among instructions. First, whether the order of instructions matters or not; Second, whether the instructions should be consecutive or not. Based on these two criteria, we use three methods to collect features.

1. The order of the sequences is not considered and there could be instructions in between.

2. The order of instructions is considered, however, it is not necessary for instruction sequences to be consecutive.

3. The instructions are both ordered and consecutive.

We call these “Type 1”, “Type 2” and “Type 3” instruction associations. “Type 3” instruction association is similar to N-Gram. “Type 2” instruction association can deal with garbage insertion. “Type 1” instruction can deal with both garbage insertion and code reorder.
Figure 3 illustrates different type of instruction associations of length 2 we have obtained on an instruction sequence consisting of 4 instructions.

Instruction Sequences: sub push mov sub

Type 1
push sub
mov sub
mov push

Type 2
sub push
sub mov
sub sub
push mov
push sub
mov sub

Type 3
sub push
push mov
mov sub

Figure 3 Instruction Associations of Length 2

DATA MINING PROCESS

The overall data mining process can be divided into 7 steps. They are:

1. Run executable inside a virtual machine, obtain instruction sequences from Ollydbg
2. Construct logic assembly
3. Generate abstract assembly
4. Select instruction associations features
5. Extract frequency of instruction associations features in the training dataset and testing dataset
6. Build classification models
7. Apply classification models on testing dataset

This process is illustrated in figure 4.

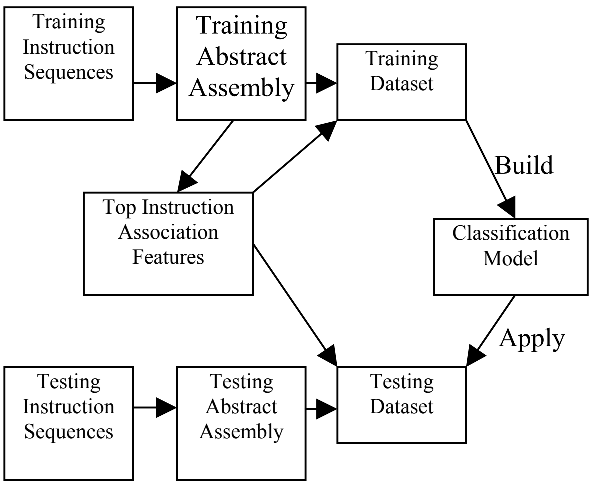

Training Instruction Sequences
Training Abstract Assembly
Training Dataset
Build
Top Instruction Association Features
Classification Model
Apply
Testing Instruction Sequences
Testing Abstract Assembly
Testing Dataset

Figure 4 Data Mining Process

Here we describe step 4 in detail. The features for our classifier are selected instruction associations. To select appropriate features, we use the following two criteria:

1. The instruction associations should be not too rare in the training dataset consisting of both benign and malicious executables. If it occurs very rare, we would rather consider this instruction association is a noise and not use it as our feature

2. The instruction associations should be an indicator of benign or malicious code; In other words, it should be abundant in benign code and rare in malicious code, or vice versa.

To satisfy the first criteria, we extract frequent instruction associations from training dataset. Only frequent instruction associations can be considered as our feature. We use a variation of Apriori algorithm [26] to generate all three types of frequent instruction associations from abstract assembly. Although there exists algorithms to optimize Apriori algorithm [30], the optimization only applicable to type 1 instruction association, besides, this step only occurs at training time. We believe optimize applying process is more critical because it will run on each computer under protection. Training, however, only need to be done on a specific hardware.

One parameter of Apriori algorithm is “minimum support”. It is the minimal frequency of frequent associations among all transactions. More specifically, it is the minimum percentage of basic blocks that contains the instruction sequences in our case. We do experiments on different support level as described in out experimental result.

To satisfy the second criteria, we define the term

$$Contrast\ (Fi) = \begin{cases} \dfrac{countB\ (Fi) + \varepsilon}{countM\ (Fi) + \varepsilon} & countB\ (Fi) \geq countM\ (Fi) \\ \dfrac{countM\ (Fi) + \varepsilon}{countB\ (Fi) + \varepsilon} & countB\ (Fi) < countM\ (Fi) \end{cases}$$

contrast

CountB (F_i)	normalized count of F_i in benign instruction file
CountM (F_i)	normalized count of F_i in malicious instruction file
ε	a small constant to avoid error when the dominant is 0

In this formula definition, normalized count is the frequency of that instruction sequence divided by the total number of basic blocks in abstract assembly. We use a larger benign code dataset than malicious code dataset. The use of normalization will factor out the effect of unequal dataset size.

We select top L features as our feature set. For one executable in training dataset, we count the number of basic blocks containing the feature, normalized by the number of basic blocks of that executable. We process every executable in our training dataset, and eventually we generate the input for our classifier.

We use two classifiers in our experiment: C4.5 decision tree [27] and libSVM [28] Support Vector Machine.

C4.5 decision tree is a classification algorithm that is constructed by recursively splitting the dataset into parts. Each such split is determined by the result of the entropy gain of all possible splits among all attributes inside the tree node. The decision tree keeps growing as more splits are performed until a specific stop rule is satisfied. During postpruning, some splits are removed to relieve overfitting problem. When a record of an unknown class comes in, it is classified through a sequence of nodes from the tree root down to the leaf node. Then, it is labeled by the class the leaf node represents.

Support Vector Machine (SVM) [35] is essentially a ma-thematical optimization problem which is originated from the linear discriminant problem. However, if two classes are inseparable in two dimensions, SVM can use a mapping, which is called kernel function, to map two dimension data into a higher dimension. The two classes may be separable in higher dimension. libSVM is a popular C implementation of SVM on Unix.

We also tested some other classifiers such as random forest [33]. We do not detect any classifier has clear advantage over others in the measure of accuracy. However, one reason drives us to use C4.5 and SVM in our experiment is that both classifiers are efficient to make decision. The performance of decision making process is the key to the system performance (See performance analysis).

EXPERIMENTAL RESULTS

Dataset

Due to the prevailing dominance of Win32 viruses to-day, we only use Win32 viruses as our virus dataset. We collect 267 Win32 viruses from VX heaven [17].

We also choose 368 benign executables which consist of Windows system executables, commercial executables and open source executables. These executables have the similar average size and variation as the malicious dataset.

For both malicious and benign codes, we randomly choose 70% of them as a training dataset and the remain-ing 30% as a testing dataset.

Criteria

In out experiment, we use accuracy on testing dataset as our main criteria to evaluation the performance of classification models. However, we also calculate false positive rate and false negative rate. False positive rate is the proportion of benign executables that were erroneously reported as being malicious. On contrary, false negative rate is the proportion of malicious executables that were erroneously identified as benign. We believe in a virus detection mechanism, low false negative rate is more vital than low false positive rate. It is wise to be more cautious against those suspicious un-known executables. High false positive certainly make things inconvenient for the user, but high false negative will destroy user's computer, which is more harmful.

Parameter Selection

There are five primary parameters in our classifier, they are:

1. Instruction association type IA (type 1, 2 or 3)

2. Support level of frequent instruction association (S). We experiment 0.003, 0.005, 0.01

3. Number of features (L), we try 10, 20, 50, 100, 200, 300. At some support level, some instruction association type generates relatively fewer number of available features. For example, at support lever 0.01, only 23 type 1 instruction associations are frequent. In that case, we use up to the maximum available features

4. Type of classifier (C), we compare C4.5 decision tree and SVM (Support Vector Machine)

5. Number of instruction captured (N). We try 1000, 2000, 4000, 6000, 8000

IA	S	L	C	N	Accuracy
2	0.01	300	SVM	1000	0.962/0.930
1	0.01	200	C45	1000	0.919/0.923
1	0.01	200	C45	6000	0.943/0.923
2	0.01	300	SVM	8000	0.950/0.920
1	0.01	200	SVM	2000	0.924/0.919
2	0.01	200	C45	8000	0.960/0.918
1	0.01	300	C45	8000	0.945/0.918
1	0.01	200	C45	8000	0.941/0.918
1	0.01	300	C45	4000	0.919/0.918
2	0.01	300	SVM	4000	0.955/0.914

Table 1 Top 10 Configurations

Table 1 lists top 10 configurations we get along with accuracy on both training dataset and testing dataset.

The result shows that support level 0.01 is clearly superior to others. It shows that frequent patterns are more important than infrequent patterns.

Instruction association type 1 and 2 outperform type 3. That is an interesting result which could serve to justify our approach in that traditional N-Gram based approach checks type 3 instruction association only.

The effect of number of instructions captured N is not quite clear yet. We further calculate average accuracy at different n in figure 5. We see that in general accuracy increase when we use a large N. However, the difference becomes very small when N>2000. That justify that when we use the first 4000 instructions, we can capture the behavior of the unknown executable. One interesting phenomenon is when N=1000, we get some really good result. Our top 2 classifiers all have the setting N=1000. That means in some settings, first 1000 instructions already capture the character of the executable, further instructions might only give noises.

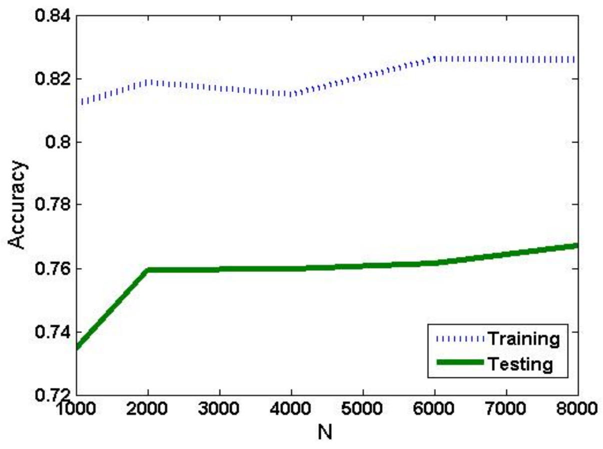


Figure 5 Effect of N

Model Selection

One problem in our best performed classifier is that it uses 300 features. The number of features affects the per-formance of our detector (See performance analysis). To this end, we choose the second best setting. The false positive rate for this classifier is 0.114, and the false negative rate is 0.013. We don't have space to show more data for false positive rate and false negative rate. In general, false positive rate is much higher than false negative rate in our experiments. That is exciting because we expect a lower false negative rate.

PERFORMANCE ANALYSIS

In this section, we focus on performance when applying the classification model on the end user computer. The performance to process one unknown executable is determined by the following factors: Capturing instruction sequences; Generating logic assembly; Counting the occurrence of instruction associations in feature set to generate testing features; Applying classification model.

Unlike system call, instruction sequences generate fast and at a stable rate. On our test computer, we generate around 6,000 instruction sequences in 1 second under Ollydbg. That is enough for the input for our classifier. This is the one major advantage over system call approach, which takes time to get enough system call traces.

Generating logic assembly consists of three phases. In the first phase, we need to sort the instruction sequences according to their virtual address. This could take up to O(nlogn) to finish. In the second phase, we mark jump destination using one linear scan of all instructions, which takes O(n). Maintaining different incarnations requires a memory map to remember the instruction and incarnation of each virtual address. Every instruction takes linear time to check this memory map, so this additional task will not increase the order of the overall processing time. Finally, we traverse the sorted instruction list to output basic blocks, which takes O(n). So the overall time complexity in logic assembly generation is O(nlogn).

Generating testing features requires counting the fre-quency of L features. Suppose average basic block contains k instructions, thus we have average n/k basic blocks. For every basic block, we will do a search for each one of L features.

Different types of instruction association use different approach to search inside a basic block. For type 1 instruction association, we use an occurrence bit for every instruction in the association, if all bit is on, then the basic block contains that instruction association. For type 2, we construct a finite state machine (FSM), and scan the basic block from the beginning. If we encounter an instruction matching the state in FSM, we advance the state of FSM, and begin matching the next instruction. For type 3, it is similar to a substring search. All these three types of search requires only one linear scan of the basic block, makes the bound of O(k).

We can calculate the processing time of testing feature generation as the multiply of the above factors. So this step takes (search time per feature per block)* (feature number) * (basic block number) = O(n/k*k*L) = O(nL).

The time complexity to apply a classification model is a property of specific classification model. For C4.5 decision tree, the applying time complexity is proportional to the depth of the tree [27], which is a con-stant at the applying time. SVM takes O(L) to apply the model on a specific sample [31].

Based on the discussion above, we conclude that the time complexity to process an unknown executable is bounded by max (O(nlogn), O(nL)), in which n is the number of instructions captured, L is the number of features.

In our experiment, processing instructions captured in 1 second, for which $n \approx 6000$, the calculation time is usually less than 3 seconds. This suggests that this approach can be used in practice.

CONCLUSION

In this paper, we have proposed a novel malicious code detection approach by mining dynamic instruction sequences and described experiments conducted against recent Win32 viruses.

Experimental results indicate that the proposed data mining approaches can detect malicious codes reliably even for the unknown computer viruses. The best classification rate on testing dataset is 93.0%. The performance in measure of time is acceptable in practical usage.

Compared with other approaches, instruction association deal with the virus code directly and is robust to me-tamorphism.

We also plan to build an end user simulator based on the best data mining model. The simulator will run the unknown executable inside a controlled environment, capture initial dynamic instruction sequences and make decision based on them.

REFERENCES

[1] Microsoft Antimalware Team , "Microsoft Security Intelligence Report", Volume 3, 2007, http://www.microsoft.com/security/portal/sir.aspx

[2] C. Nachenberg, "Computer virus-antivirus coevolution", Communications of the ACM, Volume 40 , Issue 1, pp:46–51, 1997

[3] P. Sz¨or and P. Ferrie, "Hunting for metamorphic", 11th International Virus Bulletin Conference, Prague, Czech Republic, 2001

[4] Jeffrey O. Kephart, William C. Arnold, "Automatic Extraction of Computer Virus Signatures", 4th International Virus Bulletin Conference, Jersey, Channel Islands, 1994.

[5] Stuart Staniford, Vern Paxson, Nicholas Weaver, "How to 0wn the Internet in Your Spare Time", 11th Usenix Security Symposium, San Francisco, USA, 2002

[6] F. Cohen, "Computational Aspects of Computer Viruses", Computers & Security, volume 8, pp:325-344, 1989

[7] William Arnold, Gerald Tesauro, "Automatically generated Win32 heuristic virus detection", 10th International Virus Bulletin conference, Orlando, FL, USA, 2000

[8] InSeon Yoo, "Visualizing windows executable viruses using self-organizing maps", Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, Fairfax, VA, USA, 2004

[9] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables", Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2001

[10] Abou-Assaleh, Nick Cercone, Vlado Keselj, and Ray Sweidan, "Detection of New Malicious Code Using N-grams Signatures", Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST'04), pp: 193-196, Fredericton, New Brunswick, Canada, 2004

[11] Abou-Assaleh, Nick Cercone, Vlado Keselj, and Ray Sweidan, "N-Gram-based Detection of New Malicious Code", Proceeding of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Hong Kong, China, 2004

[12] Kolter, J.Z., & Maloof, M.A., "Learning to detect malicious executables in the wild", In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp:470-478. New York, NY, 2004.

[13] Sung, A.H et al "Static analyzer of vicious executables (SAVE)", 20th Annual Computer Security Applications Conference, Tucson, AZ, USA, 2004

[14] Mihai Christodorescu, Somesh Jha, Sanjit A. Seshia, Dawn Song, Randal E. Bryant, "Semantics-Aware Malware Detection", IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2005

[15] Mihai Christodorescu, Somesh Jha, "Static Analysis of Executables to Detect Malicious Patterns", 12th USENIX Security Symposium, Washington DC, USA, 2003

[16] Christopher Kruegel et al, "Static Disassembly of Obfuscated Binaries", Proceedings of the 13th conference on USENIX Security Symposium, San Diego, CA, USA, 2004

[17] Cullen Linn et al, "Obfuscation of Executable Code to Improve Resistance to Static Disassembly", Proceedings of the 10th ACM conference on Computer and communications security, Washington D.C., USA, 2003

[18] B. Schwarz, S. Debray, G. Andrews, "Disassembly of Executable Code Revisited," wcre, p. 0045, 9th Working Conference on Reverse Engineering, Richmond, Virginia, USA, 2002

[19] Bayer, U., Kruegel, C., Kirda, E, "TTAnalyze: A Tool for Analyzing Malware", 15th Annual Conference of the European Institute for Computer Antivirus Research, Hamburg, Germany, 2006

[20] Steven A. Hofmeyr et al, "Intrusion detection using sequences of system calls", Journal of Computer Security, Volume 6 , Issue 3, pp:151-180, 1998

[21] Wenke Lee and Salvatore J. Stolfo, "Data Mining Approaches for Intrusion Detection", 7th USENIX Security Symposium, San Antonio, Texas, USA, 1998

[22] Warrender, C et al, "Detecting intrusions using system calls: alternative data models", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 1999

[23] Tony Lee, Jigar J. Mody, "Behavior Classification", 2006, http://blogs.technet.com/antimalware/archive/2006/05/16/428749.aspx

[24] http://www.ollydbg.de/

[25] Md. Enamul. Karim et al, "Malware Phylogeny Generation using Permutations of Code", Journal in Computer Virology, Volume 1, Numbers 1-2, 2005

[26] Rakesh Agrawal, Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules", Proc. 20th International Conference of Very Large Data Bases, VLDB, Santiago de Chile, Chile, 1994

[27] J.R.Quinlan, "C4.5:Programs for Machine Learning", Morgan Kaufmann Publishers Inc, 1993

[28] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[29] http://vx.netlux.org

[30] Jiawei Han , Jian Pei , Yiwen Yin, "Mining frequent patterns without candidate generation", Proceedings of the ACM SIGMOD international conference on Management of data, Dallas, Texas, USA, 2000

[31] Vladimir Vapnik, "Statistical Learning Theory", John Wiley & Sons, 1998

[32] http://research.microsoft.com/sn/detours/

[33] Breiman L, "Random Forests", Machine Learning Volume 45, pp:5-32, Kluwer Academic Publishers, 2001

[34] http://www.isotf.org/zert/

[35] John Shawe-Taylor & Nello Cristianini, "Support Vector Machines", Cambridge University Press, 2000

MIAT-WM5: FORENSIC ACQUISITION FOR WINDOWS MOBILE POCKETPC

Fabio Dellutri, Vittorio Ottaviani, Gianluigi Me

Dip. di Informatica, Sistemi e Produzione - Università degli Studi di Roma "Tor Vergata"
Via del Politecnico 1, 00133 Rome, Italy
Email: {dellutri,ottaviani,me}@disp.uniroma2.it

KEYWORDS

Mobile Forensics, Data Seizure, PDA, PocketPC, Windows Mobile

ABSTRACT

A PocketPC equipped with phone capabilities could be seen as an advanced smartphone, providing more computational power and available resources. Even though several technologies have emerged for PDAs and Smartphones forensic acquisition and analysis, only few technologies and products are capable of performing forensic acquisition on PocketPC platform; moreover they rely on proprietary protocols, proprietary cable-jack and proprietary operating systems. This paper presents the Mobile Internal Acquisition Tool for PocketPC devices. The approach we propose in this paper focuses on acquiring data from a mobile device's internal storage memory, copying data to an external removable memory (like SD, mini SD, etc.). Such task is performed without the need of connecting the device to PC. Thanks to this, forensic operators could avoid to travel with luggage plenty of one-on-one tools for every single mobile device. Finally, we will show some experimental results, comparing this methodology with standard products on real world devices.

INTRODUCTION

The mobile phone can be considered the ultimate disruptive technology: in fact, like telephony, radio, television, and the Internet, mobile phones are dramatically changing nearly every aspect of daily life, both inside businesses and in the daily lives of individuals, providing more applications and collecting more private data. The enriched capabilities of new smartphones (118 million in 2007, Canalys), such as multi-connectivity (HDSPA, Bluetooth, IR, WLAN) and multimedia recording, make the content of the mobile device memory very interesting from a criminal investigation perspective. In particular, the growing prominence of forensic sciences, in the investigation chain, led to a broad use of forensic tools to acquire mobile phone memory content, to witness the evidence of a crime. However, as rule of thumb, the crime-scene usually offers many different mobile phone/smartphone models, causing the forensic operators to be overwhelmed by using the one-on-one connectors for every single mobile device. In fact, current acquiring tools, adopted by forensic operators, extract the internal memory remotely (typically via USB), with proprietary mobile phone connectors: a forensic tool running on a laptop is connected with the target device and, using the OS services, it extracts the data like SMS, MMS, TODO list, pictures, ring tones etc. This approach has the advantage

- to minimize the interaction with the device and
- to automate the procedure of the interpretation of the seized data.

However, the main disadvantage is the partial access of the file system, which relies on the communication protocol. As discussed above, since many protocols are proprietary, we can not see how many effects the data exchanged have on the memory status. For this reason, we have developed a tool to acquire mobile phone memory by a SD/MMC (Secure Digital / MultiMediaCard) memory inserted in the available mobile phone SD/MMC slot, called MIAT (Mobile Internal Acquisition Tool) for Symbian (Me et al., 2008). In the fourth quarter of 2007, Canalys estimated that Symbian had a 65% share of worldwide converged device shipments, ahead of Microsoft on 12% and RIM on 11% (Canalys, 2007). In this paper we present the MIAT for the PocketPC platform running Windows Mobile (MIAT-WM5), which cannot be considered, roughly, as a porting because of the differences between Symbian and Windows Mobile operating systems. For this reason, the MIAT-WM5 takes advantage of Windows Operating system characteristics (e.g. filesystem, memory management etc) to maximize the outcome of the acquisition phase for Windows devices.

STATE OF ART

The term *PocketPC* refers to a Microsoft specification that sets various hardware and software requirements for a handheld-sized computer (*PDA*, Personal Digital Assistant) that runs the Windows Mobile operating system (Wikipedia, Pocket PC definition). As reported in Table 1, many tools perform forensic operations on a PDA. However, as Ayers *et al.* assert in (Ayers et al., 2007, 2004, 2005), the only NIST certified tool on a PocketPC is Paraben's *PDA Seizure*. Such tool performs data seizure of internal memory in a remote way. Actually, the

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

Table 1: PDA forensic tools

	Palm OS	PocketPC	Linux PDA
pdd	Acquisition	NA	NA
Pilot-Link	Acquisition	NA	NA
PDA Seizure	Acquisition, Examination, Reporting	Acquisition, Examination, Reporting	NA
EnCase	Acquisition, Examination, Reporting	NA	Examination, Reporting
POSE	Examination, Reporting	NA	NA
dd	NA	NA	Acquisition

forensic tool is connected to a device by cradle or USB cable and, through the Microsoft's ActiveSync protocol, it extracts data such as user's files, call logs, SMS, MMS, TODO list, etc. This approach has the advantage to minimize the interaction towards the device and to automate the process of seized data interpretation. The main disadvantage relies on the protocol closeness: we are not able to measure any memory alteration caused by data exchange. Moreover, to perform the acquisition process, PDA seizure degrades the evidence putting a dll file in the device's file system.

POCKETPC INTERNAL MEMORY AND STORAGE ARCHITECTURE

In Windows Mobile 2003 PocketPC and earlier, device's memory was split in two sections: a ROM section, containing all operating system core files, and a RAM section aimed in keeping the user storage (Storage Memory) and the memory space for running applications and their data (Program Memory). The user can choose the amount of memory to be reserved to Storage Memory and then to the Program Memory. The RAM chip was built on a volatile memory scheme, so a backup battery was required to keep the RAM circuitry powered up, even if the device was just suspended. In case battery power supply went down, all user's data were lost. Such scenario forced user to recharge battery within a time limit of 72 hours (as mandatory by Microsoft to devices manufacturers).

Since Windows Mobile 5, memory architecture was redesigned to implement a non-volatile user storage.

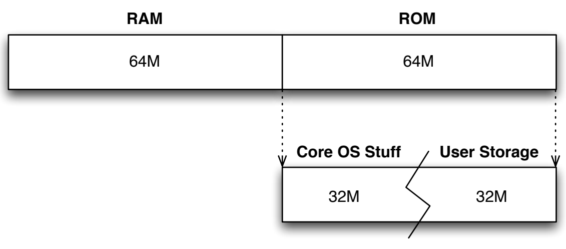

RAM
ROM
64M
64M
Core OS Stuff
User Storage
32M
32M
Memory sizes reported could change among different PPC models.

Figure 1: Windows Moble 5.0 memory architecture.

Currently, the memory is split in two section (see Figure 1): the RAM is aimed to hold running processes data, whereas the ROM keeps core OS code and libraries (called modules), the registry, databases and user's files. Such memory, also called Persistent Storage and contained within a flash memory chip, can be built using many different technologies (Santarini, 2005):

- **XIP model**, based on NOR memory and volatile memory, this technology enables device to store modules and executables in XIP (execute-in-place) format and allows the operating system to run applications directly from ROM, avoiding to copy them first in the RAM section. NOR memory has poor write performance.

- **Shadow model**, which boots the system from NOR and uses a NAND for the storage. This model is power-expensive, because the volatile memory requires to be constantly powered on.

- **NAND store and download model**, which reduces costs replacing NOR with OTP (one-time programmable) memory model.

- **Hybrid store and download model**, which mixes SRAM and NAND, covering them with a NOR-like access interface (to support XIP model).

Windows Mobile 5 and above place the great part of the applications and system data in the Persistent Storage. Core OS files, user's files, databases and registry are seen by applications and users in the same file system tree, which is hold and controlled by the FileSys.exe process. Such process is also responsible for handling the Object Store, which maps objects like databases, registry and user's files in a contiguous heap space. The Object Store's role is to manage the stack and the heap memory, to compress and to expand files, to integrate ROM-based applications and RAM-based data. For a comprehensive explanation about how Windows Mobile uses the Object Store and manages linear flash memory, see (Microsoft, Linear Flash Memory Devices on Microsoft Windows CE.) and (Microsoft, The Windows CE 5.0 Object Store.).

The strategy for storing data is based on a transactional model, which ensures that store is never corrupted after a power down while data is being written. Finally, the Storage Manager manages storage devices and their file systems, offering a high-level layer over storage drivers, partition drivers, file system drivers and file system filters.

OUR METHODOLOGY

The approach we propose in this paper focuses on acquiring data from a mobile device's internal storage memory, copying data to an external removable memory (like SD, mini SD, etc.). Such task is performed without the need

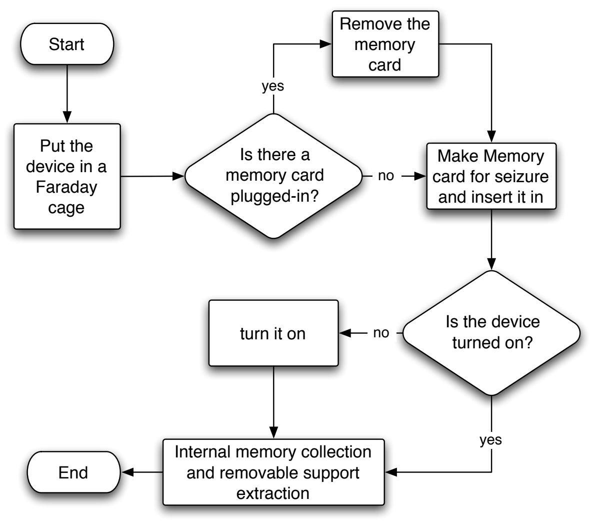

Start
Put the device in a Faraday cage
Is there a memory card plugged-in?
yes
Remove the memory card
no
Make Memory card for seizure and insert it in
Is the device turned on?
no
turn it on
yes
Internal memory collection and removable support extraction
End

Figure 2: Data collection workflow

of connecting the device to PC. Thanks to this, forensic operators could avoid to travel with luggage plenty of one-on-one tools for every single mobile device.

The complete data seizure process is shown in Figure 2. In order to acquire the memory content of a GSM, Bluetooth or Wi-Fi enabled mobile device, it is mandatory to shield the device with a Faraday cage (Leyland, 1992). Indeed, new incoming calls, SMS, e-mails, Bluetooth activity, connection status changes or GSM cell switch, could trigger events which may modify some file system's objects. Unlike old Symbian smartphones, where we were forced to remove battery supply to remove the memory card, in a standard PocketPC it is possible to plug-in a memory card (typically an SD) while the device is powered-on (*hotplug*). This is a great chance for collecting data which, otherwise, could be altered if the device was turned off before the seizure process. Therefore, we have to check first if a memory card is already plugged, and replace it with a memory card containing MIAT-WM5. Moreover, if the device is turned off, now it can be started. MIAT-WM5 can be set as autorunnable, to avoid to start applications like fexplore.exe to navigate through the filesystem and to launch the seizure application; indeed it is important to run as few applications as we can, to avoid locking problems and changes in the file system triggered by other processes. Anyway, MIAT-WM5 kills all non-necessary processes running on the system in order to avoid lock problems. MIAT-WM5 performs a hashing of each file before and after the copy, to ensure acquired image integrity. The report containing file hashes is saved in a log file. Data stored in the original memory card can be acquired using a MMC or SD reader (USB or integrated) and a byte stream imaging tool (like DD): binary data are read from source, then stored as an image file, representing all the single bytes, including file system's metadata. After that, it is possible to analyse the file allocation table to recover deleted data. When internal memory seizure has been done, SIM card could be removed and analysed with specific tools (Oxygen Software, Forensic; Casadei et al., 2005).

IMPLEMENTATION DETAILS

We have chosen to develop the application using a native C++ approach, fulfilling the requirement of having a tool to be launched from an external memory card, without the need of a pre-installed runtime environment (like java virtual machine), neither the need to install the tool on the device. The application runs in stand-alone mode, and it does not require any third party's dll. Since the tool uses the standard Windows Mobile APIs to access the file system (like Open, Read and Write, FileCopy), we can reasonably think that these APIs will not change in future versions of OS: then the forward compatibility can be assured. In Algorithm 1 is depicted the pseudo-code of the seizure process, that starts after the main application killed all the other non-vital running processes.

Algorithm 1 Seizure

```
Input: A path p.
Output: none.

for all objects obj (files and directories) in p do
  if obj is a directory then
    Create a directory named p in the SD Card
    Recursively call Seizure(p/obj)
  else if obj is a file then
    Compute MD5 hash of obj
    Copy obj in path p on the SD Card
    if obj has not been copied then
      Access to obj with CEDB APIs
      if obj could be accessed then
        recreate a similar database in path p on the SD Card
      end if
    end if
    Compute MD5 hash of the copied obj on the SD Card
  end if
end for
```

Such algorithm performs two main tasks:

- the *copy* task, which copies all internal memory's files of the mobile device on the memory card;
- the *hash* task, which ensures the integrity of the copied files and allows to discover which files have been modified during the seizure process.

The *Seizure* algorithm works using APIs like CopyFile Open Close, and it copies recursively every internal file system entry on the memory card. This task preserves the directory structure, copying files according to their original position. The hash task computes the MD5 hash of each file found in the device internal memory. Hashes are written in a log file saved in a separate directory. The

Table 3: MIAT-WM5 and PDA seizure comparison, and their hashes consistency

File	Paraben	MIAT
/Documents And Settings/default.vol	–	⋆
/Documents And Settings/system.hv	–	–
/Documents And Settings/default/user.hv	–	–
/Windows/*.dll	–	–
/mxip_notify.vol	√	⋆
/cemail.vol	√	⋆
/mxip_system.vol	√	√
/mxip_lang.vol	√	√
/pim.vol	⋆	√

– file not copied
⋆ file copied but its hash does not match
√ file copied and hash matches

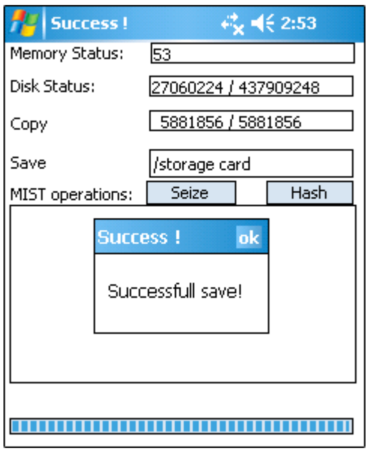
hash task can be launched as a separate function, and it surfs the whole filesystem to compute hash of every files.

The *Seizure* algorithm invokes the hash function before and after the copy of every single file, allowing to understand if changes happen during the copy from the internal filesystem to the Storage Card.

As reported in Section "POCKETPC INTERNAL MEMORY AND STORAGE ARCHITECTURE", Windows Mobile places OS's stuff in a lot of file-like objects in the same file system seen by the user (under /Windows directory). Most of these files are inaccessible by the standard file system APIs because they are objects that are in XIP format: most of the headers are removed and the addresses are fixed up so that the programs are able to run with no need to be loaded into RAM first. The binary has been stripped down and customized for that particular device (Yost, 2007). Such files are also flagged with file attributes like FILE_ATTRIBUTE_INROM and FILE_ATTRIBUTE_ROMMODULE. Our application skips these files: there is no reason to look for a method to access such files because they are firmware's modules and they could be replaced with new ones only by an advanced user (using the *ROM flashing* technique - e.g. if she is willing to upgrade her firmware with a new version of the operating system or she want to modify things like bootsplash). Moreover, there is another set of files that cannot be accessed by standard APIs: these files are database objects locked by operating system processes which cannot be killed. We reach to access their data using CEDB APIs and we are able to recreate such files in the external memory card. In Table 2 it is shown where most relevant data about user and system are stored in the file system.

EXPERIMENTAL RESULTS

The working approach of MIAT-WM5 and Paraben is quite different. MIAT-WM5 scans the filesystem of the PocketPC saving data in the external memory card, as Paraben seems to get data from ROM memory at a lower level than MIAT-WM5. There are some differences between Paraben and MIAT-WM5, first of all Paraben needs a computer to seize data from a mobile device, as it is a Windows application. As described above, MIAT-WM5 does not need any other device to run: the operator needs only an external memory card pluggable in the device. Looking at the results of some tests done on a physical HTC device and on a emulated one (on a Windows XP computer), Paraben and MIAT-WM5 seems to extract the same files. Some differences rely on hashes of some files: Paraben seems to modify some files, such as pim.vol, and it preserves others, but, as its source code is not available, we can not explain what happens. MIAT-WM5 modifies some files, because it access their data through database APIs and writes a new file containing same data, with a resulting hash different from the original (see Table 3). As described before, OS's core files are impossible to be extracted both for MIAT-WM5 and Paraben. Another relevant consideration to be done is that Paraben reach to recover something from deleted files (just erased before the seizure), but in all the experiments these files looked like zero-padded. That suggests that Windows Mobile replaces erased block sectors very quickly, probably because the memory size is too short to preserve them for the entire seizure time. In the testing phase, we used a PC AMD Athlon64 X2 Dual 1GB Ram and a QTEK9000 PDA (HTC Universal), equipped with a Kingston SD 2GB. On such hardware, seizure times between the two solutions are quite the same: Paraben's time depends only on how many files resides on the internal storage; MIAT-WM5's time relies both on files amount and on external memory access time.


Success !
Memory Status: 53
Disk Status: 27060224 / 437909248
Copy 5881856 / 5881856
Save /storage card
MIST operations:

Successfull save!

Figure 3: MIAT-WM5 screenshot after run a data seizure

Thanks to its hashing characteristics, MIAT-WM5 was also useful to enumerate all file system's changes when device sustains events like simple reset, on-line/off-line mode change or SIM card removal. Such results are shown in Table 4. Some files' hashes are impossible to be computed ("-" symbol) because those files are locked and the system does not allow one to perform read operations on such files.

Table 2: Windows Mobile 5.0 relevant files

Filename	Location	Description
System.hv	/Documents And Settings/system.hv	System registry hive.
User.hv	/Documents And Settings/default/user.hv	User registry hive for default user.
Default.vol	/Documents And Settings/default.vol	Object store replacement volume for persistent CEDB databases. This file contains MSN contacts
Mxip_system.vol, Mxip_lang.vol, Mxip_notify.vol, Mxip_initdb.vol	/	Metabase volumes, including language-specific data and storage for notifications.
Cemail.vol	/	Default SMS and e-mail storage.
Pim.vol	/	Personal Information Manager (PIM) data, such as address book, schedules, SIM entries, call logs.

Table 4: File system changes before and after a event and files lock status

File		Before	After
Device reboot			
/Windows/VSDApp.bin	⊙	□	□
/History.txt	⊙	□	□
/mxip_system.vol	-	□	□
/mxip_lang.vol	-	■	□
From on-line to off-line mode			
/Windows/Profiles/guest/Temporary Internet Files/ Content.IE5	-	□	■
/Windows/Profiles/guest/Cookies/index.dat	-	□	■
/Windows/Profiles/guest/History/History.IE5/ index.dat	-	□	■
SIM card removal			
/mxip_system.vol	-	■	□
/mxip_lang.vol	-	■	□
/History.txt	⊙	□	□
/Windows/VSDSIMInfor.bin	⊗	□	⊗
/Windows/VSDApp.bin	⊙	□	□
/Windows/Profiles/guest/Cookies/index.dat	-	■	□
/Windows/Profiles/guest/History/History.IE5/ index.dat	-	■	□
/Windows/Profiles/guest/Temporary Internet Files/Content.IE5	-	■	□
/Windows/Start Menu/Programmi/ SIM i.TIM.lnk	⊗	□	⊗

⊙ Modified in the transition
⊗ Disappear after the change
□ Unlocked
■ Locked
- Unavailable

CONCLUSIONS

In this paper we proposed an alternative methodology to seize data from PocketPC devices, based on our belief that forensic model should be opened and verifiable. Therefore, MIAT-WM5 could be proposed as an open-source software, in order to give it transparency and verifiability.

Currently we are working on combining files copy with a full internal memory dump. With this approach, we will ensure to extract a complete and consistent snapshot of the system. This approach could involve a device-dependent code because each manufacturer uses a different memory technology (as discussed above in Section "POCKETPC INTERNAL MEMORY AND STORAGE ARCHITECTURE"), therefore it implements its own low level storage SDK (needed to find the storage addresses range into the ROM).

Moreover, we are improving the application design to support Windows Mobile 6.0 as well.

ACKNOWLEDGEMENTS

We wish to express our thanks to Giuseppe F. Italiano, who reviewed the draft of this document.

REFERENCES

Me, G., Rossi, M. (2008). *Internal forensic acquisition for mobile equipments.* 4th Int'l Workshop on Security in Systems and Networks (SSN2008), Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), 2008, IEEE Computer Society Press, TO APPEAR.

Canalys.com (2007). *Smart mobile device shipments hit 118 million in 2007, up 53% on 2006.* http://www.canalys.com/pr/2008/r2008021.htm.

Wikipedia. *Pocket PC definition* http://en.wikipedia.org/wiki/Pocket_PC

Ayers, R, Jansen, W, Moenner, L, Delaitre, A. (2007). *Cell Phone Forensic Tools: An Overview and Analysis Update.* NISTIR 7387, March 2007

Ayers, R, Jansen, W. (2004). *PDA Forensic Tools: An Overview and Analysis.* NISTIR 7100, August 2004

Ayers, R, Jansen, W. (2005). *An overview and Analysis of PDA Forensics Tool.* The International Journal of Digital Evidence, 2005

Paraben's Forensics Software *Paraben Device Seizure.* www.paraben.com

Santarini, M. (2005) *NAND versus NOR.* EDN, October 13, 2005

Microsoft. *File System Boot Process.* MSDN, msdn2.microsoft.com/en-us/library/aa912276.aspx

Microsoft. *Linear Flash Memory Devices on Microsoft Windows CE.* Microsoft TechNet, www.microsoft.com/technet/archive/wce/plan/flashce.mspx

Microsoft. *The Windows CE 5.0 Object Store* MSDN, msdn2.microsoft.com/en-us/library/ms885891.aspx

Leyland, W. J. (1992). *Lightweight portable EMI shielding container.* United States Secretaty, The Navy US. OF. (US) 5136119 http://www.freepatentsonline.com/5136119.html

Oxygen Software. www.opm-2.com/forensic/

Casadei, F, Savoldi, A, Gubian, P. (2005). *SIMbrush: an Open Source Tool for GSM and UMTS Forensic Analysis.* In proceedings of Systematic Approaches to Digital Forensic Engineering (IEEE SADFE 2005), pgg. 105-119, Taipei, TW, 7-9 November 2005.

Yost, S. (2007). *Why can't I copy programs out of Windows?.* http://blogs.msdn.com/windowsmobile/archive/2007/12/29/why-can-t-i-copy-programs-out-of-windows.aspx

AUTHOR BIOGRAPHIES

FABIO DELLUTRI is a Ph. D. student in computer science engineering at the Computer Science Engineering Department, Università degli Studi di Roma "Tor Vegata". His interests include Web applications and OS security, mobile digital forensics and experimental algorythms. His email is `dellutri@disp.uniroma2.it`.

VITTORIO OTTAVIANI is a Ph. D. student in computer science engineering at the Computer Science Engineering Department, Università degli Studi di Roma "Tor Vergata". His interests include web applications mobile and OS security, mobile digital forensics and GIS applications. His email is `ottaviani@disp.uniroma2.it`.

GIANLUIGI ME, Ph. D., is an adjunct professor of Computer Security at the Università degli Studi di Roma "Tor Vergata". He holds a strong experience in managing training for law enforcement high tech crime units and serves as a member of the advisory board of the International Journal of Electronic Security and Digital Forensics. His research interests include mobile computing applications, digital forensics, electronic/mobile payments, and game theory. He is a professional member of the IEEE. His email is `me@disp.uniroma2.it`.

A proposal for securing a large-scale high-interaction honeypot

J. Briffaut, J.-F. Lalande, C. Toinard
Laboratoire d'Informatique Fondamentale d'Orléans
Université d'Orléans, rue Léonard de Vinci
45067 Orléans, France
{jeremy.briffaut,jean-francois.lalande,christian.toinard}@ensi-bourges.fr

KEYWORDS

High-Interaction Honeypot, Attack Monitoring, IDS

ABSTRACT

This paper presents the design of a secured high-interaction honeypot. The challenge is to have a honeypot that welcomes attackers, allows userland malicious activities but prevents from system corruption. The honeypot must be scalable to authorize a large amount of malicious activities and to analyze those activities efficiently. The hardening of the honeypot is proposed for two kinds of host. The first class prevents system corruption and has never to be reinstalled. The second class assumes system corruptions but easy reinstallation is available. A first cluster enables to deploy a wide range of honeypots and security sensors. A second cluster provides an efficient auditing facility. The solution is totally based on open source software and has been validated during one year. A statistical analysis shows the efficiency of the different sensors. Origin and destination of attacks are given. Moreover, the complementarities of the sensors are discussed. Ongoing works focus on recognition of complex malicious activities using a correlation grid.

INTRODUCTION

This paper proposes an architecture for securing high-interaction honeypots. The main objective is to welcome attackers and to provide different operating systems. The difficulty of high-interaction honeypot is that hackers can gain a complete control of the system. So, the risk is high that attackers use the honeypot to carry out severe attacks. Deploying high-interaction honeypot is a challenging research activity and few works address this problem with real operating systems and services. Classical approaches use low interaction honeypot but they limit the malicious activities. That is why high interaction is required. Currently, they need frequent reinstallations and advanced monitoring of the activities. The main objective is to prevent high-interaction honeypot from frequent reinstallation. The second objective is to efficiently monitor the malicious activities and to maintain a cluster of operating systems connected to public Internet addresses.

Usually low-interaction honeypots do not authorize the attacker to gain a login shell on the real system. In low-interaction honeypot all the services are emulated and even the login shell is emulated. The main drawbacks of these low-interaction honeypots are: the attacker can discover that he is connected to a fake system; the services are partially emulated; the vulnerabilities of these services are missing; host based attacks are impossible to capture.

A high interaction honeypot is required to capture host based attacks and to offer the vulnerabilities of the target system. As each operating system, distribution, service, software can contain different vulnerabilities according to their version, the deployment of a large heterogeneous cluster of honeypots is required to increase the number of possible attacks.

As attackers have a direct access to a real system and can exploit the vulnerabilities of the target system, they can gain administrator privileges and compromise easily the system. The main problems related to high interaction honeypots are: 1) An attacker can exploit a vulnerability in order to obtain administration privileges and stop the monitoring mechanisms; 2) The operating system can be stopped or broken; 3) The attacker can use the honeypot to attack other hosts on the Internet; 4) A large number of operating systems have to be deployed and monitored to offer a large amount of vulnerabilities; 5) The malicious activities generate a large amount of traces and analysis become difficult; 6) The volume of data, that has to be stored, is even larger when auditing system calls and when using a great number of complementary sensors.

The easiest solution to prevent the preceding problems is to reinstall frequently the operating system. First, it is not feasible for a large scale honeypot. Second, attacks are lost and monitoring is corrupted. Finally, the decision of reinstalling the system requires external analysis and is complex. Moreover, an attacker can discover that the operating system has been cleaned between two connections and possibly can understand that he is connected to a honeypot.

In this paper, a clustered honeypot is proposed that offers vulnerable systems accessible by public Internet addresses. Our high interaction honeypot provides two types of hosts, 1) secure hosts that let the attackers use the target system but avoid the compromising of the sys-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

tem and 2) classical systems with strong monitoring and reinstallation facilities. A second cluster allows collecting and analyzing the activities. Monitoring is available at various host levels (i.e. from system call level to shell level) but also at various network levels. The cluster is safe since it minimizes the possibility of using the honeypot to attack outside hosts.

STATE OF THE ART

Two types of honeypots are distinguished in the literature. The low level honeypots emulate a limited part of the services, mainly the network protocols. It allows to compute statistical results and to model the attack activities (Kaaniche et al., 2006). On the other hands, the high-interaction honeypots deploy real operating systems that allow capturing more complete information. Some high-interaction honeypots use VMware to emulate the host (Alata et al., 2006) but hiding the hypervisor is very hard to achieve: attackers may give up their attacks because the operating system will probably be reinstalled. For the papers that focus on the low interaction honeypots, or on the exploit of services (Diebold et al., 2005), the problem is simpler than ours since there is less security aspects addressed.

In different white papers about honeypots (Nguyen et al., 2005; Baumann and Plattner, 2002), the different generations of honeypots and their architectures are described. Our proposed honeypot is part of the second generation of honeypots described by (Nguyen et al., 2005). Several toolkits are presented as the well known Honeyd (Provos, 2004) which is used by Leurre.com, a distributed honeypot. These white papers cover the network configuration, host sensors and the modus operandi for collecting data at the date of year 2003.

The problem of configuring the honeypots is difficult because one have to find a trade-off between a real production host and an adaptive honeypot that let the attackers enter the system. If the honeypots are too easy to enter, the attacker could guess that the host is not a production host. On the other way, no results can be obtained if the honeypot is hidden or too hard to attack. A large variety of papers try to propose new kinds of honeypots that solve parts of this problem.

Paper (Hecker et al., 2006) presents how to configure honeypots dynamically based on network scans. It improves the initial work of Hieb and Graham to adapt the honeypot behavior (the opened ports) to the detected attacks. In (Kuwatly et al., 2004) the authors tries to build a honeypot that can be plugged into a local network and that find automatically unused IP. These works allow a system administrator to deploy a honeypot according to his network configuration and to let the honeypot evolves and adapts the services with the requests of attackers. These goals are crucial when deploying some honeypots on a large scale network with thousands of hosts but the choice of the operating system and the security of this operating system remains an opened question that we want to address with this paper.

Some papers address the techniques to detect that a host is a honeypot (Holz and Raynal, 2005; Innes and Valli, 2005). These techniques are more or less related to kernel analysis that allows detecting a User-mode Linux or a VMware host. This is precisely these papers that show that deploying real operating systems with real services is better for capturing attackers, but is possibly more dangerous for other hosts on the network.

In (Anagnostakis et al., 2005), the authors propose an advanced hybrid type of honeypots: shadow honeypots. They analyze the network traffic in order to redirect suspicious packets to a shadow honeypot. This shadow honeypot is a replication of the protected software that can be used to analyze the received attacks. These types of solution are complex to deploy and our purpose is to collect more information by letting intruders attack directly the host.

HIGH INTERACTION HONEYPOTS

The figure 1 describes the global architecture of the clustered high interaction honeypots. The architecture is separated in four parts: the Internet access management, the clustered honeypot, the cluster for auditing and storage, and finally the correlation grid. This section details each part of the proposed architecture and the objectives, the installed software and their configuration. Intentionally, focus is done on technical aspects to make possible reproducing this clustered architecture.

Network management

The honeypot hosts are directly connected on the Internet. Each honeypot has a public IP. All the connections from Internet to the honeypots are forwarded through a Honeywall host. The goals of the Honeywall are the following: 1) It limits the bandwidth usage from the inside to the outside to 10MB/hour; 2) It limits the number of TCP connection from the inside to the outside to 100 connections per hour; 3) It limits the bandwidth usage from the outside to the inside to 100MB/hour.

The Honeywall has two network interfaces in bridge mode (no IP, just forwarding packets) and this way cannot be seen by intruders. Limitation of bandwidth and connections is achieved by iptables rules in order to limit the impact of these possible attacks: 1) Denial of Service attacks (DOS) from the inside to the outside; 2) Port scans from the inside to the outside; 3) Denial of Service attacks (DOS) from the outside to the inside.

There is no firewall blocking some ports or services on the Honeywall host. Thus, the intruders see the honeypots as frontal hosts on the Internet. Those honeypots can be used to attack from the inside other hosts on Internet but with very limited resources.

In order to analyze the network traffic outgoing from the Honeywall, a hub replicates the packets to a network analyzer host. This host contains a network Intrusion Detection System (Snort IDS) that analyzes the traffic and

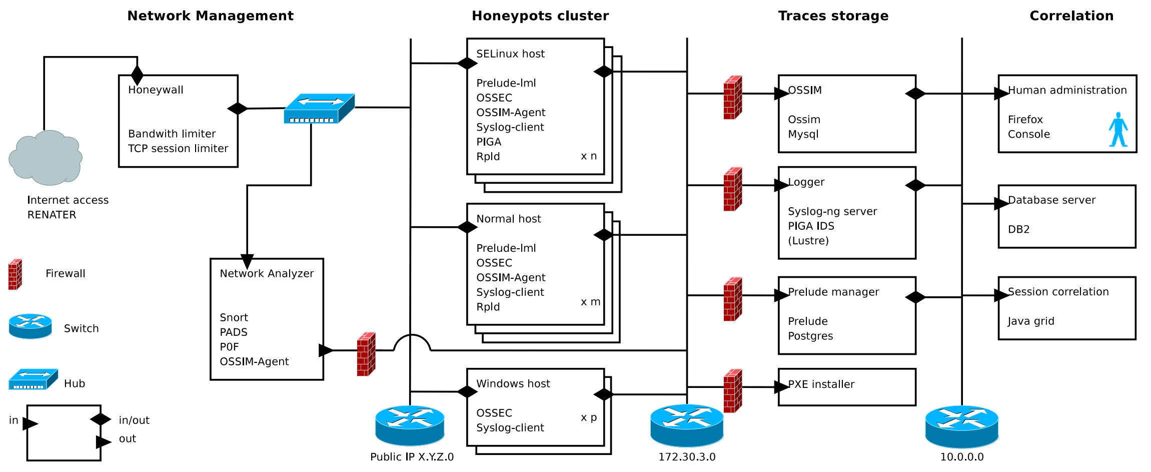


Figure 1: Clustered high interaction honeypots architecture

possibly send alerts to the local OSSIM agent. It keeps also a TCP dump (PCAP file) of all the incoming and outgoing network traffic. Traces and OSSIM alerts are sent to the second cluster for auditing and storage. Two other network IDS are installed in order to detect the operating system type of the attacker (P0F) and the services (PADS). The analysis is passive and thus invisible from the intruder side. The generated alerts are sent to the OSSIM agent.

The network analysis and the Honeywall are separated because an attacker can possibly exploit a vulnerability against one of the network IDS. This way, the network analyzer can be compromised or disabled but cannot be used as starting point to attack outside hosts because of the limitation guaranteed by the Honeywall.

In the network management part of our architecture, control of possible malicious network traffic is supported and monitoring is available, using network IDS, that provides alerts and event data for the correlation phase.

Clustered honeypots

The second part of our architecture is the honeypot hosts. Three kinds of honeypots are represented in figure 1: 1) Mandatory Access Control enforced the security of the operating systems (MAC) (Bell and La Padula, 1973); 2) “Classical” hosts without MAC but Discretionary Access Control systems (DAC) (Harrison et al., 1976); 3) Microsoft Windows operating system.

The MAC and DAC systems are deployed on clustered hosts with five GNU/Linux distributions: debian, gentoo, fedora and ubuntu. The Microsoft Windows systems are: NT 4.0, 2000, 2003, XP. Each host has two network interfaces, one with a public IP address and one with a local address 172.30.3.x. This way, all the honeypot hosts are connected to a local network that enables an intruder, when connected on one of these hosts, to attack other honeypots. No firewall is installed on the 14 honeypots.

Each GNU/Linux host has an OPENSSH modified service that facilitates the open of sessions for the intruders: when an attacker attempts to log on one of the honeypots with an ssh brute force scan, our openssh service creates randomly accounts with a probability of 1% using the attempted login/password. It gives the intruder a real account on the system with a home directory. The created account is persistent and authorizes the intruder to come later with this login/password and to continue his attack.

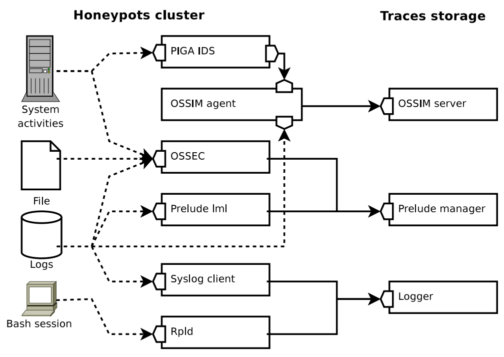


Figure 2: IDS and tools monitoring host activities

In order to capture the activities of the intruders, several host IDS and tools are deployed on the GNU/Linux and Windows systems. These IDS monitor four types of information sources: the system activities (system calls, processes), the integrity of the file systems, the logs of kernel and daemons, and the bash sessions. All these information are complementary and enable correlation between network and host data. On each host, the installed IDS and tools are the followings: 1) Prelude-lml is a system log analyzer that reports system activities (connections, daemon logs like apache, ...). 2) OSSEC checks the integrity of the system files, rootkit installation and modification of the registry and logs. 3) OSSIM agent collects the alarms generated by PIGA IDS and analyzes system logs. 4) PIGA IDS is a policy based IDS that detect violation of security properties (Blanc et al., 2006). 5) Syslog client forwards all kernel and system logs to the Logger host. 6) Rpld captures the shell activities performed by the attackers and allows to replay them.

On figure 2 we sum up the data used by each IDS (dotted lines). All the collected alarms, logs, sessions are sent to the next part of our architecture, the traces storage clusters.

Traces storage

The hosts of the third part of the proposed architecture are used to store and analyze the alarms and logs captured on honeypots. The hosts are connected on the 172.30.3.0 network and protected by firewalls allowing only incoming traffic. The opened ports are those used by the OSSIM server, Prelude server, and syslog server (logger).

Three analysis frameworks are used in order to collect all events and generate reporting alerts readable by humans: 1) OSSIM: it provides a framework to manage security information. It generates reports, aggregates alerts, send incident tickets, . . . It stores the collected data into a mysql server (possibly a clustered mysql). 2) Prelude: it aggregates the collected information and enables to visualize them on a website. All events are stored using the IDMEF standard. A postegreSQL server is used (possible a clustered postegreSQL). 3) Syslog (logger): it stores all the syslog traces of the honeypots on a distributed file system (LUSTRE).

All network and system events/alarms are stored and can be visualized by an administrator. As the three frameworks do not use the same standards, we needs this three servers for the correlation phase. Another goal of these servers is to prevent an attacker to delete the log/-traces/activities generated on honeypots as the reported events are transmitted in real time (it needs the time for the daemon to send the information to the server).

Correlation

This last part of our architecture is used to visualize the alarms and to test correlation algorithms between all kinds of IDS alarms. Indeed, the main goal is to characterize attacks using network IDS alarms, host IDS alarms, system events logged in traces. These algorithms send request to the database of OSSIM, Prelude and syslog using the private network 10.0.0.0. As the format of the events is not standard, we merge them in a neutral format into the database server used by our correlation algorithms. These algorithms, not described in this paper, are written in java and have high CPU and memory consumption. These algorithms cannot be done in real time and use a java grid to distribute the computation.

SECURITY OF HONEYPOTS

The main drawback of our architecture is to provide real systems with vulnerabilities that possibly allow an attacker to get administrator privileges. In this section we details MAC and DAC hosts.

MAC hosts

With a MAC host, a policy guarantees that the root (staff role) user does not have the privileges of the super administrator (admin role). If an attacker exploits a vulnerability and becomes root, the policy limits his privileges on the system as a normal user. The only way of becoming super administrator is to exploit a kernel vulnerability or to attack the MAC mechanism (SELinux). In this case, the system is compromised and have to be reinstalled using the PXE server. Nevertheless, during one year of deployment, we never observed such an attack.

The main advantage of these hosts is that they are time persistent. An attacker can come back later to finish an attack whereas a reinstallation will suggest that the host is a honeypot. Moreover attackers can explore the different home directories of other attackers and possibly reuse/delete/download the uploaded scripts. All these shell activities are logged by the Rpld server.

DAC hosts and modifications

This classical system can easily be compromised by an attacker. When an user gets administrator privileges, the host have to be reinstalled with the PXE server. The drawback of these honeypots is that they need more administrative tasks. Alerts have to be daily monitored and a decision has to be taken to know when the system is “too” compromised. We implemented a cron job that computes the differences that appear during the time. When a honeypot file differs from the corresponding file stored on the PXE, an alert is sent by OSSEC integrity analyzer and the difference is stored on the PXE SERVER for further manual analysis.

Auto-installation

The PXE server contains boottp and a tftp server in order to deploy fresh images of our distributions. It contains Linux images, Debian, Gentoo, Fedora and Ubuntu with and without SELinux MAC mechanism and Microsoft Windows systems, NT 4.0, 2000, 2003, XP. This server is used to reinstall a compromised honeypot using the PXE protocol on the 172.30.3.0 network.

STATISTICAL RESULTS

The presented statistics aggregate activities from February 27th 2007 to February 21th 2008 for 2 MAC honeypots (Debian, Gentoo) and 2 DAC honeypots (Debian, Gentoo) and 1 windows (NT 2000).

Per host results

Figure 3 represents the distribution of alerts per host. There are two reasons that explains the low number of alerts on the DAC host. First, we give public IP addresses in September 2007: attacks on these hosts are achieved on the private network before this date and Snort did not report anything during the six first months. Second, PIGA IDS cannot be deployed without a MAC mechanism, reducing the number of alerts. Note also that the logger

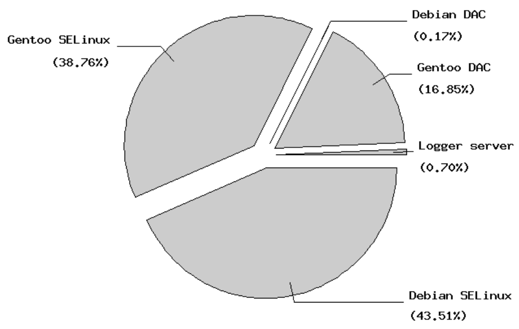


Figure 3: Alarms per host

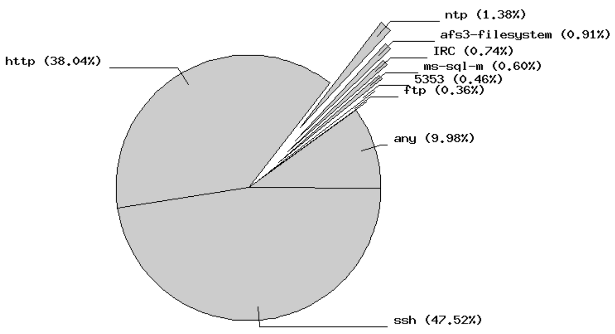


Figure 5: Alarms per port

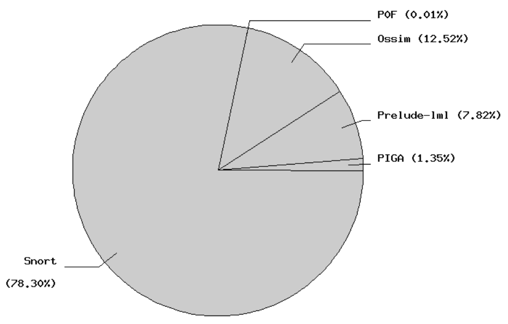


Figure 4: Alarms per sensor

Sensor	Description	Ocurences
Prelude-lml	SSHd: Root login refused	498,468
Snort	Destination udp port not reachable	452,011
Prelude-lml	SSHd: Bad password	49,329
OSSIM	SSHd: Possible brute force tentative	43,989
Prelude-lml	SSHd: Invalid user	43,311
PIGA	Integrity: system file modification	41,063
Prelude-lml	FTP bad login	21,366
Snort	Potential outbound SSH scan	19,983
PIGA	Confidentiality: information flow	16,191
	. . .	
Snort	Alarm for signature k	< 1,000
Snort	. . .	< 1,000
Snort	Alarm for signature k+1	< 1,000
	. . .	

Table 1: Main types of alarms

host has also been attacked even if this host has no public IP address and is protected by a firewall.

There is no real difference between a Debian or Gentoo distribution because attackers do not target one distribution. It can be explained because a large part of the reported alarms are networks alarms that does not target a specific host.

Statistics for IDS tools

In the database server 8,206,382 events and 302,543 alarms are stored i.e. 950 events per hour and 35 alarms per hour. An event is an information (as a user connection) and an alarm is a malicious event possibly a part of an attack (as a malformed packet). It is a lot of alarms to monitor, but the major part of these alarms is generated by Snort and is mainly false positives.

Figure 4 shows the distribution of the network and host IDS alarms. Snort reports a large amount of false positives. Indeed, it uses a signature base and detects attacks in packets using pattern matching: it cannot know if the attack succeeded or may succeed. This is the main drawback of network IDS. False positives are eliminated using PIGA IDS that detects only 45,590 opened sessions by scan robots and only 2,219 sessions performing activities on the corresponding host.

Table 1 reports the main types of alarms and the sensors that detect it. The most frequent alarm deals with the ssh daemon and is reported by Prelude-lml: it is the first step to enter our honeypots. Intruders use ssh scan tools and try to brute force passwords that generates a lot of alarms. Note that ftp accounts are also targeted by these scan attacks. For outgoing traffic, Snort reports 19,983 ssh scans because the intruders use gained accounts to attack outside hosts.

PIGA IDS detected the modifications of the configuration files that a normal user should not access. It detected also information flows mainly from those configuration files to the user space. Moreover, information provided by specific files like /etc/shadow or /etc/apache/httpd.conf have been stolen by the attackers.

The total amount of Snort alarms is 78%. It includes 452,011 UDP port scans plus a lot of alarms with lower rate associated to different IP addresses.

The figure 5 is consistent with table 1: the ssh port dominates the number of alarms. The http port is mainly attacked from the outside in order to install fishing websites. 10% of attacks are malicious ICMP packets. Classical ports are used in order to exploit classical windows vulnerabilities (afs3 filesystem, ms-sql-m): these attacks are worms trying to propagate themselves. The IRC port is used by IRC bots installed on the honeypots in order to be connected on outside IRC channels. The bots can be controlled by attacker's orders on the channel and could be used to launch DOS attacks.

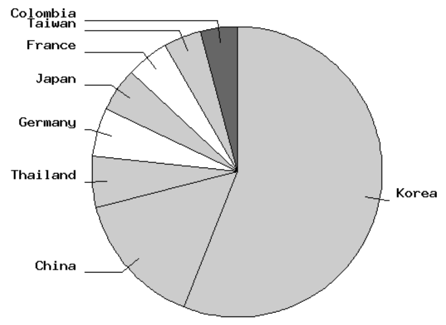


Figure 6: Alarms per country - incoming

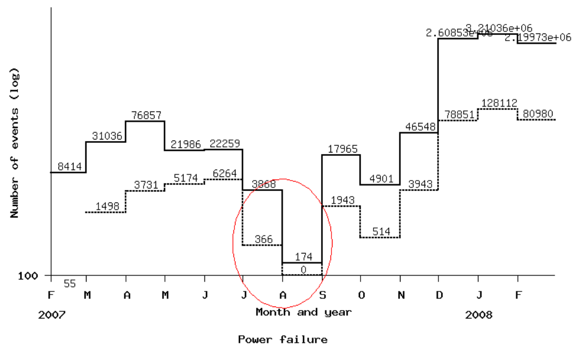


Figure 8: Number of events during a year (logarithmic)

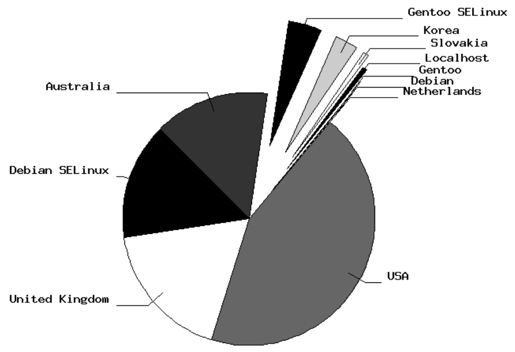


Figure 7: Alarms per country - outgoing

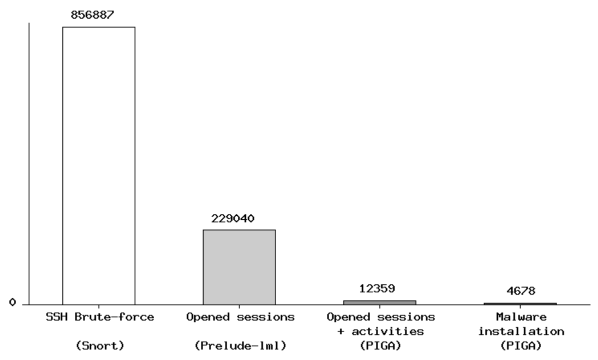


Figure 9: Number of malicious activities seen by sensors

Results per country

Figures 6 and 7 give the ratio for the total of reported alarms per country. Incoming and outgoing alarms are separated: the first ones are alarms generated when the intruders penetrate our honeypots. The second ones are alarms generated when the intruders use our honeypots as a base to launch attacks against outside or local hosts.

The world region that launches most of the attacks is Asia (grey). The other attackers are mainly from Europe (white). Note that no attack is incoming from the United States whereas the target of outgoing attacks on figure 7 is more than 40% against the USA.

On figure 7 we added the attacks against our local hosts. The ratio between attacks against external host or local host is biased because of the Honeywall that limits outgoing bandwidth and connections. The most attacked local host is the Debian SELinux host mainly because it has a public IP address (and not the Debian host): it suggest that attackers does not try to discover local hosts that have not an Internet address; they prefer to attack a second host that have an Internet address which is the Gentoo SELinux host.

Time results

Figure 8 shows the evolution of the number of events (solid lines) and alarms (dotted lines) during one year of experiments. We can observe a hole of events during summer because of a large power supply failure of 3 weeks. In December 2007 we added Prelude-lml to our architecture and 4 new hosts which increased the number of events and alarms. If we omit the failure problem and the arrival of Prelude-lml the number of alarms is constant with time which confirms that attackers are using random IP addresses on Internet.

We also computed the number of alarms per hour and we observed that the variations of events/alarms are not related to specific hours. It varies randomly from 230,000 to 750,000 events and from 9,000 to 20,000 alarms when considering a specific hour. As intruders use scripts in order to attack the honeypots, we cannot observe specific activities on some precise hours.

Discussion

Figure 9 shows the number of malicious activities seen by each sensor. Snort detects the number of brute force attacks (more than 850,000) but cannot decide if sessions are opened. Prelude-lml reports that 230,000 sessions have been opened. In theory the system offers 1 session over 100 brute force attacks. In practice, 229,000 sessions have been detected instead of the 85,000 theoretical sessions because the attackers use similar dictionaries and the tried logins have probably already been created. Moreover, the attackers can come back later with the created login/passwords which increase the number of sessions seen by Prelude.

The third sensor, PIGA IDS, detects opened session with real activities such as shell commands for exploring the host, download of software, execution of binaries or scripts. The number of activities (12,000) shows that 95% of sessions are never exploited by the intruder. Indeed, a bruteforce of ssh creates several accounts (because of the 1% creation policy). A successful intruder will only use one of the created accounts. PIGA reports the number of malware installation. For this purpose, PIGA factorizes a download, installation, execution of a software as a malware alarm. We conclude that 37% of the sessions deals with malwares such as IRC bots or ssh scanners.

These results show the complementarity of those sensors. Snort enables to know the different intrusion attempts but cannot decide if the attempts is successful. Snort cannot analyze further the attempt and complementary tools are needed. Moreover, for ssh connections, a network IDS is useless to analyze the use of the connection. Though, other host IDS are required to monitor the next steps of the attack. In order to do it efficiently, system calls must be monitored. In this direction PIGA IDS proved his efficiency to detect complex scenarios of attacks.

CONCLUSION AND PERSPECTIVES

During one year of experiments, MAC honeypots never needed reinstallation despite of the detection of 229,000 and 12,359 malicious activities. This result shows the robustness of the proposed architecture. The DAC hosts have been reinstalled only three times in three months. A good monitoring was proposed to both MAC and DAC system. Automation of reinstallation have been completed. Our study shows the need of complementary network and host IDS. Some tools generate a large number of false positive whereas other tools are more precise in the analysis of intrusions.

The decision of the reinstallation of DAC hosts was taken manually. In order to make DAC hosts as much persistent as possible, correlation between the different collected data is required to take the reinstallation decision. Ongoing works will compute a risk level using correlation results. Moreover, the data of a compromised system have to be manually analyzed using the modification files such as described in section . Future works will study how to associate those modifications to a known malware. This way unknown malware could be highlighted.

The next step of this work is to correlate events and alarms between all sensors in order to model complete session of attacks. This work has been done for the system events of a MAC host. The major difficulties are related to the correlation with network events and with distributed attacks.

REFERENCES

Alata, E., Nicomette, V., Kaâniche, M., Dacier, M., and Herrb, M. (2006). Lessons learned from the deployment of a high-interaction honeypot. In *6th European Dependable Computing Conference*, pages 39–44, France. IEEE Computer Society.

Anagnostakis, K. G., Sidiroglou, S., Akritidis, P., andE. Markatos, K. X., and Keromytis, A. D. (2005). Detecting targeted attacks using shadow honeypots. In *14th USENIX Security Symposium*, pages 129–144, Baltimore, MD.

Baumann, R. and Plattner, C. (2002). White paper: Honeypots.

Bell, D. E. and La Padula, L. J. (1973). Secure computer systems: Mathematical foundations and model. Technical Report M74-244, The MITRE Corporation, Bedford, MA.

Blanc, M., Briffaut, J., Lalande, J.-F., and Toinard, C. (2006). Distributed control enabling consistent mac policies and ids based on a meta-policy approach. In *Workshop on Policies for Distributed Systems and Networks*, Canada London. IEEE Computer Society.

Diebold, P., Hess, A., and Schäfer, G. (2005). A honeypot architecture for detecting and analyzing unknown network attacks. In *14th Kommunikation in Verteilten Systemen 2005*, Kaiserslautern, Germany.

Harrison, M. A., Ruzzo, W. L., and Ullman, J. D. (1976). Protection in operating systems. *Communications of the ACM*, 19(8):461–471.

Hecker, C., Nance, K. L., and Hay, B. (2006). Dynamic honeypot construction. In *10th Colloquium for Information Systems Security Education*, University of Maryland, USA.

Holz, T. and Raynal, F. (2005). Detecting honeypots and other suspicious environments. In *Information Assurance Workshop*, pages 29–36, University of Maryland, USA.

Innes, S. and Valli, C. (2005). Honeypots: How do you know when you are inside one? In Valli, C. and Woodward, A., editors, *4th Australian Digital Forensics Conference*, Perth, Western Australia. School of Computer and Information Science, Edith Cowan University.

Kaaniche, M., Deswarte, Y., Alata, E., Dacier, M., and Nicomette, V. (2006). Empirical analysis and statistical modeling of attack processes based on honeypots. In *Workshop on Empirical Evaluation of Dependability and Security (WEEDS)*, pages 119–124, Philadelphia, USA.

Kuwatly, I., Sraj, M., Masri, Z. A., and Artail, H. (2004). A dynamic honeypot design for intrusion detection. In *ICPS '04: Proceedings of the The IEEE/ACS International Conference on Pervasive Services (ICPS'04)*, pages 95–104, Washington, DC, USA. IEEE Computer Society.

Nguyen, H. Q., Pouget, F., and Dacier, M. (2005). White paper: Integration of honeypot data into an alert correlation engine. Technical report, Institut Eurecom, France.

Provos, N. (2004). A virtual honeypot framework. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, Berkeley, CA, USA. USENIX Association.

IDENTITY BASED DRM SYSTEM WITH TOTAL ANONYMITY AND DEVICE FLEXIBILITY USING IBES

Sharath Palavalli, U S Srinivas and Alwyn R Pais
Department of Computer Engineering
National Institute of Technology Karnataka, Surathkal, Srinivasnagar (PO), India-575025
Email: palavalli.sharath@gmail.com

KEYWORDS

Digital Rights Management System, Identity Based Encryption System, Smart card based DRM System, Total anonymity

ABSTRACT

Most of the Digital Rights Management (DRM) systems fail to cover all requirements like user anonymity, user fairness, security and others. Device based DRM systems, adopted by most providers, lack user fairness and mostly follow proprietary formats. On the contrary, Smart Card DRM systems satisfy user anonymity and fairness, but have certain vulnerabilities, as identified in this paper. We propose a DRM system using Identity Based Encryption System (IBES) that overcomes the deficiencies and vulnerabilities of the existing DRM systems. The proposed DRM system is an approach towards an open framework, wherein, the content processing application can be independent of the DRM provider, and the security is controlled with the help of the smart cards.

INTRODUCTION

Digital Rights Management (DRM) Systems are used to protect and manage digital content. DRM is basically an aggregation of Security Technologies to protect the interest of content owners so that they may maintain persistent ownership of their content (William and Chi-Hung, 2004). Content owners are concerned regarding the content security of individual items, e.g., pictures, videos, music, e-books, programs and games. The rampant piracy that was witnessed in the heyday of the Internet, with sharing of copyrighted videos and music, has driven the industry toward the digital rights management of content. This implies that the rights of viewing (or listening, reading, or forwarding) of each item can be controlled by the license holders, using, for example, a server that administers the rights. The rights can be administered in a number of ways, including (Amitabh, 2007):

- who can view the content,
- how many times the content can be viewed,
- at what time the rights to view content expire,
- in which geographical area the content can be viewed,
- on what devices the content can be viewed,
- restrictions on forwarding of content, and
- renewal of rights through payment mechanisms.

This mechanism of management of rights for content has given birth to the technology of digital rights management or DRM. DRM creates an essential foundation of trust, between authors and consumers, that is a prerequisite for robust market development. The desired requirements of a DRM system can be summarized as follows:

- Non-Restrictiveness
  - Device mobility
  - Communication networks Interoperability
  - Off-line usage (As applicable)
- Content Security
- Key/License management
- Consumer privacy/Anonymity
- Rights persistence
- Versatility/Content Format independent
- Monitoring/Traceability and Accountability
- Choice of multiple security levels

DRM Systems can be categorized into Device-based and Identity-based systems. Basically, the security of a Device based DRM comes from enforcing usage of compliant players and unique global device identifiers. A consumer purchasing some digital content from a server will be given a license. The license will explicitly specify the device on which the content can be used. These systems restrict user flexibility in moving content across different devices and also do not provide user/device anonymity.

Identity based DRM systems differ from device based DRM systems in the license description. Here the license would contain a user identity, which the user would prove to the application by means of a password, fingerprint scan, smart card, etc. Although in these systems the user can move content to any device, user anonymity is lost.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

To overcome this deficiency of identity based DRM systems, Hung-Min Sun et al proposed an IP based DRM system (Hung-Min et al., 2006), wherein the IP (Internet Protocol address) of the user device would be used as the identity. Although this does provide considerable user anonymity, it still can be traced back, and the approach for user fairness is accomplished in a very complex and tedious manner. Also, there is an enormous dependence on the content processing application for license validation, which decreases the system security.

A Smart card-based DRM system provides the users with more flexible usage of contents and fulfills the consumers' expectations according with "fair use" (Erickson, 2003) in the real world. Privacy has an equally important consideration. Anyone, including the server, should not know the relationship between the users and the contents. More precisely, the identity of a user for the purchased content should be anonymous, even to the servers. Besides, eavesdropping on the user-device communication channel, to the extent possible must be prevented, none the less, should not reveal any relation between the user and the content when intercepted. Conrado et al propose a Smart card based DRM system (Conrado et al., 2003) wherein the smart card acts as the user authentication device. Using the smart card provides a certain level of user anonymity and user fairness in terms of device mobility. Hung-Min Sun et al (Hung-Min et al., 2007) enhance this system to provide stronger user anonymity and also add content security to secure the content during transmission.

However, the system proposed in (Hung-Min et al., 2007) does not consider various desirable license parameters like validity, etc and more importantly, is vulnerable to a Smart Card Imposter Attack as discussed later. In our proposed approach, we employ Identity Based Encryption System (IBES) to provide anonymity and also simplify the license purchase and key retrieval processes. We modify the Identity based Encryption algorithm to give out obfuscated identities, as discussed further.

The rest of this paper is organized as follows. In Section "PREVIOUS WORK", we review the smart card-based DRM system proposed by Hung et al (Hung-Min et al., 2007) and discuss about Identity based encryption systems. In Section "PROBLEMS IDENTIFIED", we point out the various drawbacks and problems identified. In Section "THE PROPOSED APPRAOCH", we propose a new approach which improves on the weaknesses and vulnerabilities of (Hung-Min et al., 2007). In Section "SECURITY", we focus on the security issues of our approach. Finally, we conclude this paper "CONCLUSION".

PREVIOUS WORK

In this section, we review the smart card based DRM system proposed by Hung-Min Sun et al (Hung-Min et al., 2007).

Smart Card Based DRM System

The system proposed has three main entities, viz, the Content Server, the License Server and the User (Client). The Content server encrypts the content in a key K_c and stores it for download/streaming. The License server issues licenses to users for viewing the content hosted on the Content server. The License server would also take care of the payment options. The license would consist of the content encryption key, K_c, used for encrypting the content the user purchases. The User, in this case the smart card, talks to the License server through the client application, for the specific licenses, and also decrypts them for the client application. The complete process would consist of four phases. They are, the Register Phase, the Purchase Phase, the Play Phase, and the Download Phase.

1. *Register phase:* The content provider uploads the unencrypted digital content, to the content server using SSL or other secure mechanisms. The content server will package the content to an encrypted format using a content key K_c, which is randomly generated by the content server. The encrypted content is stored in the content server and hosted for download. After packaging, the content key K_c will be transmitted to the license server as $E_{PKls}(c\_id||K_c)$ and the associated signature, where E_{PK1s} is an asymmetric encryption, $PKls$ is the license server's public key, c_id is the content identity, and || represents the concatenation of strings. The license server will store c_id and K_c in its database and use them for creating licenses.

2. *Purchase phase:*

   (a) The user browses the shopping website and initiates a purchase.

   (b) The device generates a random number RAN and requests the smart card to calculate $H(RK||RAN)$ and $E_{PKls}(SSI||SK)$. Here H is a hash function, SSI is a Secret Security Identity in an anonymous payment system like 'eCash', and $SK = H(RK||c\_id||UK)$ is a content and user unique key. RK and UK are the secret keys installed apriori in the smart card. Because this calculation is executed inside the smart card, the user cannot know the secrets. SK would be used to encrypt the content key (K_c) in the license. The device transmits $\{c\_id,\ E_{PKls}(SSI||SK),\ H(RK||RAN),\ E_{UK}(RAN)\}$ to the license server.

   (c) The license server verifies the validity of the SSI and checks for sufficient funds. On successful completion of both these verifications, the license server will create the corresponding license:

$$\begin{aligned} Hash &= H(RK||RAN) \\ Details &= \{c\_id, Hash, E_{UK}(RAN)\} \\ User\_Right &= \{Details\}_{signLS} \\ License &= \{E_{SK}(K_c)||User\_Right\} \end{aligned}$$

Here $\{.\}_{signLS}$ is the signature of the message signed by the license server. $E_{SK}(K_c)$ and $E_{UK}(RAN)$ are both symmetric encryptions with keys SK and UK respectively. An license index $Index = \{c\_id||H(SK)\}$ would be created for quick access and public hosting.

(d) The license server sends the license to the user.

3. *Play Phase:*

(a) On "Play" initiation, c_id is extracted from the header of the downloaded content package. Then, the device calculates the license index $\{c\_id||H(SK)\}$ and retrieves the license. If the license does not exist in the device, it tries to fetch it from the License server. Failing to find it on the License server would imply that the user has not purchased the license.

(b) The license retrieved is passed to the smart card for validation, rights verification and content key decryption.

(c) The smart card extracts the c_id, $H(RN||RAN)$, $E_{UK}(RAN)$ and User Rights from the license by using the license server's public key. In order to get the value RAN, $E_{UK}(RAN)$ is decrypted. The RAN and the RK stored in the user's smart card are used to calculate the new $H(RK||RAN)$. If the value is not the same with the value extracted from the license, it means the license was not meant for this smart card, hence rejected. After checking $H(RK||RAN)$, the smart card checks if c_id, the identity of the content being played, matches that in the license. If they are not the same, it rejects on terms of license mismatch. If the smart card has passed both verifications, the user will be allowed to access the digital content. Then, the smart card decrypts the content key K_c. The key K_c will be transmitted to a protected memory in the device so as to restrict the access to unauthorized applications.

(d) The encrypted content will be decrypted in the protected memory of the device. Only the authorized applications can access this decrypted content and render it. When the user stops playing the content, the content and K_c in the protected memory will be deleted regardless of the content type.

4. *Download phase:* The user downloads the encrypted content package incase the content does not already exist in his device. When the user inserts his smart card into another device which does not store the content, the download phase will be re-activated. Together with the encrypted content package, the associated license also, if any, would be downloaded if necessary. In case of streaming content, only the URL and the license would be downloaded and stored.

The usage of smart card ensures that the user can play the content on any device that complies with the application requirements. Moreover it ensures the content providers that their content always remains secure, except in one case as identified further.

Identity Based Encryption System

Identity based Encryption System (IBES) is a public key cryptosystem designed mainly to remove the redundant complexity involved in the Public key Infrastructure's certification and certificate verification process. In this system, a recipients well known unique identity ID, like email address, mobile phone number, IP address, URL, etc is used as the public key for the encryption. The system architecture ensures that only the owner of this particular unique Identity has the private key for this ID, and hence none others can decrypt it. This is ensured by a Private Key Generator (PKG), the trust component of the system. Figure 1 depicts the system architecture and the various steps for secure message transfer.

The concept of IBES was proposed by Shamir way back in 1984. But the first successful and computationally feasible system was published in 2001 by Dan Boneh and M Franklin (Boneh and Franklin, 2001). Their system makes use of a concept called Weil Pairings, a bilinear function. But the computationally feasible system for mobile phones was first demonstrated in 2006, by J. S. Hwu, R. J. Chen, and Y.B. Lin with their Fast computation method for Weil Pairings (Hwu et al, 2006). With this advancement, IBES services can now be ported onto handheld devices also, which are prime content usage devices today.

The ID-based scheme consists of four algorithms: Setup, Extraction, Encryption, and Decryption. Setup is run by the PKG to generate a master key and the system parameters. This is done on input of a security parameter k_{ID}, which specifies the bit length of the group order and is regarded as the key size of the ID-based scheme. The Extraction algorithm is carried out by the PKG to generate a private key corresponding to the identity of a user. As with regular public key cryptography, the Encryption algorithm takes a message and a public key as inputs to produce a cipher text. Similarly, the Decryption algorithm is executed by the owner of the corresponding private key to decrypt the cipher text. These four functions are described as follows.

1. *Setup:* With the parameter k_{ID}, the algorithm works as follows:
   (a) Generate a random k_{ID}-bit prime p, two groups (G_1; +); (G_2; *) of order p, and the Weil pairing $e : G_1 \times G_1 \rightarrow G_2$. Choose an arbitrary generator $P \in G_1$.
   (b) Pick a random number $s \in Z_p^*$ and set $P_{pub} = sP$.
   (c) Choose cryptographic hash functions $h_1 : 0, 1^* \rightarrow G_1{}^*$ and $h_2 : G_2 \in 0, 1^n$ for some n. The public system parameters are $p, G_1, G_2, e, n, P, P_{pub}, h_1, h_2$ and the master key s is kept in secret by the PKG.
2. *Extraction:* For a given string $ID \in 0, 1^*$ as the public key, the algorithm works as follows:
   (a) Compute $Q_{ID} = h_1(ID) \in G_1$.
   (b) Set the private key $K_R = sQ_{ID}$, where s is the master key held by PKG.
3. *Encryption:* To encrypt a message M under the public key ID, the algorithm works as follows:
   (a) Compute $Q_{ID} = h_1(ID) \in G_1$.
   (b) Choose a random $r \in Z_p^*$.
   (c) Set the cipher text to be
   $C = (U, V) = (rP, M \oplus h_2(e(Q_{ID}, sP)^r))$
4. *Decryption:* To decrypt a cipher $C = (U, V)$ encrypted using the public key ID, the algorithm uses the private key $K_R = sQ_{ID}$ to compute $M = V \oplus h2(e(sQ_{ID}, U))$. This decryption procedure yields the correct message due to the bilinearity of the Weil pairing, i.e.,

$$e(sQ_{ID}, U) = e(sQ_{ID}, rP) = e(Q_{ID}, sP)^r)$$

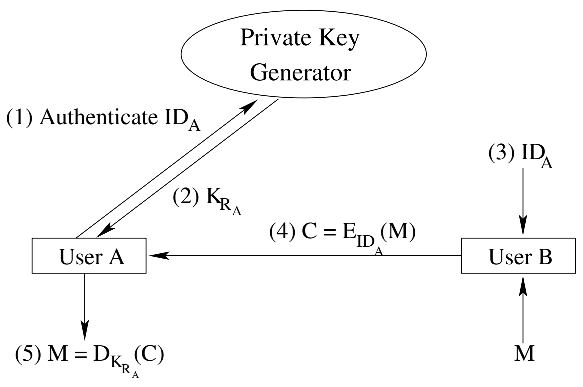


Figure 1: Identity Based Encryption System

In IBES, there is no public key exchange, nor certificate retrieval or verification, before a message transfer, as in other crypto systems. Hence a man-in-the-middle attack is not possible.

PROBLEMS IDENTIFIED

In this section we list the various drawbacks identified in various systems. Then we explain in detail the Smart Card Imposter Attack.

Drawbacks

Lists of problems identified in various types of DRM Systems:

- Device Based DRM systems
  - User fairness and flexibility
  - User Anonymity
  - Proprietary content processing applications
- IP Based DRM systems
  - User Anonymity
  - Complex process for user fairness
  - Dependence on content processing application for license validation
- Smart Card aided DRM systems
  - Vulnerable to Smart-Card Imposter Attack
  - License validity period not taken care of
  - Processing complexity in signature verification and data encryption when communicating with the license server
  - Complex algorithm with two private keys

Smart Card Imposter Attack

Hung-Min Sun et al (Hung-Min et al., 2007) assume all messages that the license server is receiving are from one of the system smart cards. But no where do they specify how to verify or authenticate if the messages are really coming from one of the system smart cards. The system security relies on the secrecy of RK and UK which are random secrets placed in each smart card.

A malicious or tampered program can impose itself as the smart card, choose two random secrets RK and UK and purchase license for a content c_id following the normal procedure, except that the program would perform the encryption operations instead of the smart card. On successful issuance of the license, which the license server would do without any suspicions, the program could decrypt the license and extract the content key K_c. Now the key can be used for decrypting content without any restrictions. Thus failing the purpose of the DRM as such.

The main intention of not verifying the identity of the smart card is to provide user anonymity, but it should not mean loss of authentication. There should be a mechanism where in the system can authenticate that the license purchase and decryption is done by a smart card only, but still not revealing which smart card. Our paper addresses this issue.

THE PROPOSED APPROACH

In this section, we propose a solution to solve the problems mentioned above. Moreover, we improve the efficiency for the low-computational-capacity smart cards. We propose to add an Identity based Encryption server to the architecture given in (Hung-Min et al., 2007), by the aid of which we would provide total user anonymity, as far as the DRM system is concerned, and also reduce the number of keys to be stored and manipulated by the smart card. Moreover it combats the Smart Card Imposter Attack mentioned above, which is a drawback of the system proposed in (Hung-Min et al., 2007).

Figure 2 depicts the architecture of the proposed system.

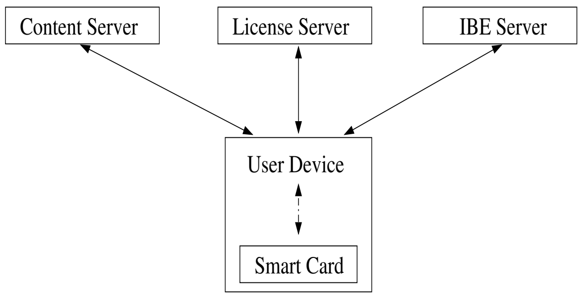


Figure 2: Architecture of Proposed System

This system would have four main roles, viz, the content server, the license server, the IBE server and the client (user). Each user would be issued a smart card which consists of a unique smart card ID and the IBE server generated private key for this ID, sQ_{ID}. It is assumed that the device used to play the content would have a card reader for the smart card. Hence the system would be able to work on any smart card based environment.

Protocol Overview

The protocol consists of four phases as in (Hung-Min et al., 2007). They are Content Registration phase, Purchase phase, Download phase and Play phase. The Content Registration phase occurs when the content provider uploads content to the content server. After the 'Content Registration Phase', the user can request purchase of the digital content on websites, and the Purchase Phase starts. Once the 'Purchase Phase' has finished, the digital content will be downloaded/streamed to the user's device in the 'Download Phase' and the user can play it in the 'Play Phase'Ḟinally, if the content which the user has bought disappears, or the user moves to a new device, the Download Phase can be re-done.

Content Registration Phase

In this phase, the content owner uploads the content to the content server, where in the content is encrypted with a randomly generated content key K_c, and a content Identity c_id is associated to it. The content server then communicates information about this content to the License server as $E_{PKls}(c\_id||K_c)$.

Purchase Phase

In the purchase phase, the user locates and shops for a content c_id. The media package would consist of c_id, the license, and the encrypted content or the streaming URL for the content. At this stage the smart card in the client device is supposed to generate an anonymous pseudo ID, SK, as per the system in (Hung-Min et al., 2007). Instead, here the smart card would reveal, to the license server, the point Q_{ID}, which is a point-mapping of the unique identity of the smart card. The smart card would send to the license server, through the user client, $E_{PKls}(SSI||Q_{ID}||c\_id)$ where SSI is the Secret Security Identifier of an anonymous payment system like 'eCash'. On receiving this message from the user, the License server would decrypt it, an extract the SSI, Q_{ID} and c_id of the content requested. It checks if the account presented has sufficient funds for purchase of content c_id and performs the financial transactions. On a successful transaction, it would create a license $L = E_{Q_{ID}}(c\_id||K_c||User\_Right\_XML)$. The license would include a small XML as to what kinds of rights are allowed and various parameters related to each right. Here the encryption followed would be that of Identity based encryption systems (Boneh and Franklin, 2001), with a slight modification that the input would be Q_{ID}, an elliptic curve point, rather than the receiver ID, and hence the initial point mapping algorithm would not be performed. We could follow an anonymous license delivery system as proposed in (Hung-Min et al., 2007), where in the license server would publish the license into a public directory with an index $I = H(Q_{ID}||c\_id)$ or could deliver the license to the user directly, as preferred by the user.

Download/Streaming Phase

This phase allows the user to download the encrypted content package when the content does not exist in his/her device. When the user inserts the smart card into another device which does not have the content, the download phase will be activated. There is no validation/restriction on who can download encrypted content from the content server, as only licensed users will be able to decrypt it correctly. Although this could lead to Denial of service attacks in case of on demand videos, this would not affect broadcast systems.

Play Phase

In this phase, when the user selects to open some content, the application would retrieve the license for the corresponding c_id and submit it to the smart card for decryption. Since the license was encrypted in Q_{ID}, only the smart card with the private key for this would be able to decrypt it, and none other. The smart card would try to decrypt the license and extract c_id. If the extracted c_id

and the content c_id whose license was submitted, are equal, then the license was generated for this smart card only, else invalid. Further, the smart card extracts various user right details like license validity, etc, checks for the expiry and if found valid, extracts the content key K_c and returns it to the application for use on a 'Protected Memory'. The application would use the content key to decrypt the content and process it as per the users request.

SECURITY

Secure against Smart Card imposter attack: Unlike as in (Hung-Min et al., 2007), here the smart card identity is a part of the license, which makes sure that the license server is talking to a smart card only and not an imposter application. Although an imposter application could submit a random Q_{ID}, or any specific Q_{ID} to the license server for procuring the license, the private key for this point would not be available, as the IBE system would insert private keys directly into the smart cards during manufacture, and also would update them, if necessary, in a secure manner. This way, there is no chance for any non-smart card program to get access to a private key in this IBE system.

Content Key Protection

Since the license was encrypted in Q_{ID}, provided by the smart card, only it would be able to decrypt it. Moreover a smart card is tamper resistant, which makes it secure against malicious or tampered applications. Now, irrespective of what application is being used to process the content, the system remains secure, as the smart card is the only one who can decrypt the license and also validate it.

User Privacy/Anonymity

One major benefit of this protocol would be total user anonymity as far as the DRM system is concerned. Since the smart card is giving out Q_{ID} instead of its ID, there is no way for anyone else to find out what the identity of the smart card is, as Q_{ID} is obtained from a mapping function that includes a one way hash and other steps. So although the license server is satisfied that the license can be decrypted only by a smart card, neither it nor any adversary, track as to who is purchasing and using what content, giving total user anonymity.

CONCLUSION

In this paper, we propose a smart card based DRM system that employs Identity Based Encryption System to combat various vulnerabilities and deficiencies identified in the previous systems. The system security relies totally on the security of the IBE server, License server and the smart cards. Assuming all these are secure enough, which is a fair enough assumption, the application that processes the content need not be a proprietary one and can follow an open architecture with different flavors. However, there is one threat, that the application gets hold of the content key, although in a protected memory area. To make the system more robust, a key evolving scheme could be employed where in the content key of the delivered content keeps changing periodically. The system provides total user anonymity and also the user fairness, as the smart card can be plugged into any compliant device to play the content. This is desirable by both the content consumers and the content providers, as now the content provider is sure that his content would stay secure knowing that only those many copies of the content are usable as many as licenses have been issued.

ACKNOWLEDGEMENT

We would like to express our gratitude towards the Information Security Education and Awareness (ISEA) Project, MCIT, DIT, Government of India, for its sponsorship, and the Department of Computer Enginering, NITK, Surathkal for providing us the necessary infrastructure and co-operation to complete this paper. We would like to thank “Prof. Asoke K Talukder”, Indian Institute of Information Technology-Bangalore (IIITB), for his motivation, guidelines and review comments.

REFERENCES

William Ku, and Chi-Hung Chi. (2004). Survey on the technological aspects of Digital Rights Management. *7th Information Security Conference*, LNCS 3225:391-403.

Amitabh Kumar. (2007). Mobile TV: DVB-H, DMB, 3G Systems and Rich Media Applications. *Focal Press publications*, ISBN 13: 978-0-240-80946-5.

Hung-Min Sun, King-Hang Wang, and Yih-Sien Kao. (2006). IP-Based DRM - A Fair and Privacy Preserving DRM Framework. *International Computer Symposium 2006*, Taipei, Taiwan.

Erickson, J. S. (2003). Fair use, DRM, and Trusted Computing. *Communications of the ACMV*, April, volume 46:34-39.

Conrado, C,; Kamperman, F,; Schrijen, C. J,; and Jonker, W. (2003). Privacy in an Identity-based DRM System. In *Proceedings of the 14th IEEE International Workshop on Database and Expert Systems Applications*, 389-395.

Hung-Min Sun, Chi-Fu Hung and Chien-Ming Chen. (2007). An Improved Digital Rights Management System based on Smart Cards. *Inaugural IEEE International Conference on Digital Ecosystems and Technologies*, IEEE DEST.

Boneh, D and Franklin, M. (2001). Identity-based Encryption from the Weil Pairing. *Advances in Cryptology*, CRYPTO'01:213-239.

Hwu,J.S,; Chen, R. J,; and Lin, Y.B. (2006). An Efficient Identity-based Cryptosystem for End-to-end Mobile Security. *IEEE Transactions on Wireless Communications*,September, Vol. 5, No. 9.

AUTHOR BIOGRAPHIES

Sharath Palavalli was born in Karnataka, India. He completed his Bachelor of Engineering in Information Science from JSS Academy of Technical Education, Visveshwaraiah Technological University, Bangalore. He is currently pursuing his Master of Technology Degree in Computer Science from NITK, Surathkal. His email is palavalli.sharath@gmail.com.

U S Srinivas was born in Andhra Pradesh, India. He completed his Bachelor of Engineering from Mother Teresa Institute of Science and Technology, Sathupally, affiliated to JNTU Hyderabad, Andhra Pradesh. He is currently pursuing his Master of Technology Degree in Computer Science from NITK, Surathkal. His email is chus84@gmail.com.

Alwyn Roshan Pais was born in Karnataka, India. He completed his Bachelor of Engineering from the Mangalore University, Karnataka and Master of Technology from IIT Bombay. He is currently pursuing his PhD from NITK, Surathkal. He is currently serving as a Senior Scale Lecturer in the Department of Computer Engineering, NITK, Surathkal. He is also the coordinator of the ISEA project at NITK. His areas of interest are Algorithms, Cryptography and Computer Vision. His email is alwyn.pais@gmail.com and has personal webpage at http://compengg.nitk.ac.in/alwyn.htm.

Requirements and Initial Design of a Grid Pseudonymity System

Joni Hahkala, Henri Mikkonen, Mika Silander, and John White

***Abstract*—Traditionally, grid users have been identifiable and traceable beyond reasonable doubt by their digital certificates. However, Grids are used in an ever-increasing variety of contexts and thus, the number of usage scenarios has augmented accordingly. In bio-medicine and other health-related fields a need for anonymous access to grid resources has been identified. Anonymous access to resources prevents the resource owners and other external parties from tracing the users and their actions. Such anonymity of resource usage in Grids is needed above all in commercial contexts, e.g. protecting the development process of a new medicine by anonymizing the accesses to medical research data bases. In this paper we identify the requirements and give an initial design for pseudonymity system addressing these needs.**

***Index Terms*—Authentication, Authorization, Grid Security, Pseudonymity.**

I. Introduction

The Grid computing model envisages a heterogeneous fabric of computing resources that is provided to users in a transparent way. In this model, Grid users may run processes on computing resources and store and access data on storage resources that may not be owned by them or even their parent organization. The use of resources on any Grid infrastructure entails a balance between the owner's need to oversee and account for the resource usage and the user's privacy requirements.

From the Grid users' point of view, complete anonymity is desirable for maximum protection. This requirement can come from researchers in a field of competitive, commercial or basic, research [1], [2]. These researchers may wish to work in secrecy and prevent their competitors from following their actions on a Grid. This would include being able to anonymize the credentials used for job submissions and the reading and writing of data. In general, this is not possible due to requirements that a Grid user should be traceable for accounting purposes and in the case of usage policy violation.

Hence, the anonymity problem is to find a compromise between the requirements of the Grid resource owner and users. The proposed solution to this problem is the concept of a lesser degree of anonymity, *pseudonymity*. A pseudonymous identity, or *pseudonym* for short, is a unique anonymous identity given by a trusted third-party (service) to a Grid user. Only this trusted third party is able to re-establish this identity association later if necessary. In situations where resource owners detect misuse of their resources, the trusted third-party can act as an middle man to solve grievances or in serious cases can be requested to disclose the true identity of the suspected abuser, subject to the policies of the particular Grid infrastructure or the law.

The system presented in this paper is designed to hide the identity of the user invoking the operations on the Grid. If used properly and provided there is a sufficiently broad mix of operations and end users, the system will also prevent the correlation of operations and thus ensure the resource owners cannot identify users by workflow tracking.

The rest of the paper is organised as follows: Chapter II looks into work generally related to pseudonymity. Chapter III offers a detailed study of the requirements. Chapter IV analyses the problem in light of the requirements and constraints. Chapter V discusses some architectural and functional issues as Chapter VI describes various solutions to the problem and specifies the one chosen. We summarize our findings in Chapter VII and propose future work in Chapter VIII.

II. Related work

Pseudonymity and pseudonym identifiers have already been covered by several specifications and software. Their definitions and interpretations are discussed in this section.

A. Shibboleth and SAML

Shibboleth[1] is Internet2's project to provide Single Sign-On (SSO) on the Web. The current version (1.3) bases on Security Assertion Markup Language (SAML) 1.1 [3] specifications, but the upcoming Shibboleth 2.0 will support major portions of SAML 2.0 [4] Both the current version of Shibboleth and the SAML 2.0 standard support short-lived opaque name identifiers. A typical use case starts at a Service Provider (SP) which wants to know some attributes of the user. Instead of authenticating the user directly, the SP redirects the user to the Identity Provider (IDP) for authentication. Once authenticated and authorized, the IDP generates an opaque name identifier for the user and communicates it to the requesting SP. The name identifier is then utilized by the SP for obtaining user attributes from the IDP.

Manuscript received February 20th, 2008. This work was supported in part by the European Union grant EGEE-II INFSO-RI 031688 and by the Joint Committee of the Nordic Research Councils for Natural Sciences (NOS-N) through its Nordunet-3 programme project "Innovative services and tools for Nordugrid".

Joni Hahkala, Henri Mikkonen and John White are with Helsinki Institute of Physics, CERN/PH, CH-1211 Geneva 23, Switzerland (phone: +41-22-76-76179; fax: +41-22-76-73600; e-mail: firstname.surname@cern.ch).

Mika Silander is with Helsinki Institute of Physics, PL 9250, 02015 TKK, Finland (phone: +358-9-8562-0909; fax: +358-9-451-5194; e-mail: mika.silander@hip.fi).

[1]Shibboleth site - http://shibboleth.internet2.edu

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

As the name identifier is opaque and short-lived, the SP cannot determine any additional user information apart from the attributes that are provided by the IDP. The attributes may include only the Virtual Organization (VO) membership information that is enough for the SP to authorize but not to individualize the user.

The integration of Shibboleth's SAML attribute framework and Grid security has been studied by the GridShib project [5]. The goals of the project include e.g. utilization of the Shibboleth attributes in the user authorization process, but also pseudonymous access for the Grid users [6]: Shibboleth's opaque name identifiers are used in the subject fields of the X.509 certificates which are issued to the users online after Shibboleth authentication. The users' Shibboleth attributes can be utilized with these pseudonym certificates too.

B. idemix

The approach described in the previous section is very IDP-centric as the IDP keeps track of its users' accesses to the SPs. Anonymous credential systems allow the user's transactions to be carried out in a way that they cannot be linked to the same user [7]. A software called idemix (identity mix) [8] is one example implementation of such a system. The users establish pseudonyms with SPs that are used for creating credentials containing a set of attributes. Afterwards, the users present the credentials with desired sets of attributes to the same or other SPs by using zero-knowledge proofs. These proofs ensure the legitimate possession of the credentials but reveal no information about the true identity of the user employing them. The credentials can be used for obtaining new credentials, but only one master secret is related to all of them. Mechanisms for retrieving the identity of a user locally (one SP) or globally (all the SPs) exist, but they require the user's cooperation.

C. WS-* specifications

As an alternative to SAML, WS-* (Web Services) specifications also provide a model for federation between IDPs and SPs. From the set of specifications, WS-Federation defines a Pseudonym Service which maintains alternate identity information for its users [9]. The pseudonym identifiers are part of the security tokens that are used by resources' Security Token Services (STS) for authentication and authorization purposes. In addition to the Shibboleth-style short-lived (or one-time) opaque pseudonym identifiers, anything between them and constant clear-text identifiers are supported. As the communication with the service itself can occur via IDPs, the resources' STSes, or directly with the resource or the requestor, numerous use cases are supported. A Pseudonym Service and the claims-based authorization model can be used to describe the set of attributes required to access a resource and the IDP can assert that a particular Grid user possesses those attributes, without divulging their actual identity.

III. Requirements

In order for a pseudonymity system to function with current Grid middleware it should fulfill the following requirements.

Requirement 1. ***Confidentiality***
The pseudonymous identity must hide the true identity of the user.

The true identity must remain unknown to the service provider sites, their administrators and other legitimate Grid users as well as external parties. This implies encrypted communications is needed between pseudonym attestee and attester.

Requirement 2. ***Non-repudiation/Retrievability***
The true identity of a Grid user must be, a posteriori, unambiguously traceable via the pseudonymity system.

This functionality is mandatory in cases of misuse and may be imposed by regulatory or law enforcement issues. To this end, the pseudonymity attester must authenticate pseudonym requestors and maintain a record of the pseudonyms issued.

Requirement 3. ***Uniqueness and Short Life-time***
A pseudonym must be a unique, short-lived one-time identity in the Grids in which it is to be employed.

A pseudonym's Distinguished Name (DN) must not clash with the existing user DNs, nor with other pseudonyms as this would undermine the overall user authentication and violate the earlier requirement of retrievability. And ideally, only one Grid operation or set of operations should be performed under the protection of a pseudonym. For the next operations, a new pseudonym should be requested. This approach reduces the ability of outside observers to collect data for correlation attacks with the intent of discovering the true identity of the user. Pseudonymous credentials should be ephemeral to reduce the damage in cases of credential compromise. Due to ephemerity and large volume of issued credentials, the issuance itself should involve no manual intervention nor procedures.

Assuming there are several independent pseudonym attesters active in a Grid, each must be assigned an own unique name space. This name space prevents attesters from accidentally issuing pseudonyms with identical DNs.

Requirement 4. ***Identity Protection***
The pseudonymity attester must be the only party able to obtain the true identities of users.

The pseudonymity attester must adequately protect the records of issued credentials and the systems into which they are stored. Only authorized people are entitled to uncover the true user identities.

Requirement 5. ***Credential Source Compatibility***
The pseudonymity system should interoperate with different sources of Grid user credentials.

Even though the user authentication is based on credentials, they may not necessarily come directly from the user's client software. The user's short- or long-term credentials can be stored in online credential repositories or be delegated to other Grid services acting on a user's behalf. For example, some portal usage scenarios involve the delegation of the user's proxy certificate [10] directly to the portal with no user intervention. Hence, the pseudonym system must support a broader set of use cases, not only those implied by direct user access.

Requirement 6. ***Information leakage prevention***
The pseudonymity system must actively counteract the leakage of information that allows the unique identification of a pseudonym user.

The operations and actions a pseudonym user performs and the set of additional personal attributes the user may have requested for inclusion into the pseudonym credentials, may provide enough information to uniquely identify the user. The pseudonymity system should therefore attempt to actively reduce and hide sources of such information. In the case of personal attributes from auxiliary authorization systems, the pseudonymity system should either prevent uniquely identifying pseudonyms to be issued, or, warn the user about the high probability of disclosure prior to using such a credential. Other covert sources of information, e.g. IP numbers of job submission hosts, metadata in submitted files and job description language attributes are harder to deal with and their removal or anonymization is ultimately up to the users themselves.

Requirement 7. ***Maintaining security***
The pseudonymity system must not provide ways to circumvent existing security.

The pseudonymity must not erode the security of the systems. There might be cases where enforcing a detailed policy would need the user's identity to be revealed and in these cases the policies can't be enforced at the time. Later, the authorized persons can do the enforcement of these policies and the corresponding actions if needed.

IV. Problem Analysis

Many of today's Grid middleware systems authenticate users with PKI certificates. Thus, in addition to the requirements presented in the previous section, we impose on ourselves an implementation constraint, that of compatibility: *The pseudonymity certification must be compatible with the certificate-based authentication and interoperate seamlessly with existing Grid middleware*. This also means the pseudonymity certification must work in concert with other commonly used auxiliary authorization systems such as Virtual Organization Membership Service (VOMS [11]) and Community Authorization Service (CAS [12]) without any significant changes to them.

Existing Grid user certificates and software can be employed with little effort to ensure that pseudonym requesters are unambiguously authenticated with cryptographically strong mechanisms. For the same reason, SSL/TLS channels can easily be set up to guarantee the confidentiality of communications. These fulfill the requirements 1 and 2 and comply with the above implementation constraint.

The pseudonym credential itself can be modeled as a standard X.509 [13] user certificate but having an anonymizing DN. The set of resources available to a Grid user acting under a pseudonym will be more limited than if the user had employed their ordinary user certificate. This is due to the fact that authentication and authorization decisions must be based solely on auxiliary attributes provided by auxiliary authorization systems, the exact user identity being unavailable. Thus, users should be able to request some of their real identities' attributes to be included in the pseudonyms and this implies the pseudonymity system needs to interact directly with auxiliary authorization services. The VOMS auxiliary authorization service models the user attributes as Attribute Certificates [14] (AC). It is commonplace to include these into the extensions of X.509 certificates and this is a further motivation to use X.509 certificates as the format for pseudonym credentials.

Certificate Authorities (CAs) may freely issue certificates unconstrained by any name space. In Grids however, the uniqueness of certificates is guaranteed by reserving specific name spaces for each CA. Only those certificates issued in conformance with the name space restriction are accepted as valid credentials in a Grid. The uniqueness of pseudonym credentials implied by requirement 3, can be ensured similarly by assigning unique name spaces to the pseudonym attesters.

Requirement 3 also states the pseudonym credentials need to be short-lived which implies a high volume of credentials to be issued. Hence they should be generated programmatically. The pseudonym system must therefore incorporate functionality similar to online certificate authority (online CA) services, e.g. EJBCA [15].

Requirement 4 has two implications: firstly, the pseudonym credential must not contain any information as to the identity of its requester, secondly, the internal security procedures and measures of the pseudonym attester must ensure the access to the records is strictly limited to authorized personnel.

Requirement 5 describes the different types of sources from which pseudonym credential requests may originate. Pseudonym certificates requested by the user, either with the help of their long-term user certificates or a proxy certificate generated from the former, will leverage the existing X.509 authentication as is. In the course of using Grid resources, the user may delegate their rights to further components acting on their behalf such as the credential repository service, MyProxy [16]. These, in turn, may delegate the credentials further to Grid portals and hence, pseudonymity requests from portals need to be handled. In order to increase entropy and thus hinder statistical correlation attacks (req. 6), a portal should request new unique pseudonym credentials for each job launched. Portals will delegate the pseudonym user's rights further to the point where the job reaches a Computing Element(CE) [17]. The CE is responsible for collecting the resources defined by the job description and selects a computing node for the execution of the job. The collection is done with the permissions of a limited proxy. The CE may also decide to request new pseudonym credentials for each resource access needed for the staging of the job, thus, the pseudonymity system must accept requests from CEs as well.

Requirement 6 is the most difficult to address since there are many sources that indirectly provide more information concerning the pseudonym user's identity. The pseudonym attester may however, guarantee that the additional personal attributes the user wishes to include in the pseudonym credential, e.g. role, group membership information, capabilities etc, will not uniquely identify the user. This requires modification of auxiliary authorization services since these must provide the pseudonymity system information about whether the requested user attribute combination is uniquely identifying or not.

According to the requirement 7, adding a pseudonymity system into the overall security infrastructure must not weaken security nor introduce new security holes. The ability of the pseudonymity system to circumvent purely identity based limitations like blacklists is at first sight one such hole. However, an abuser of pseudonyms will be detected equally and can be deprived the usage of pseudonyms. Other limitations on the user's credentials like a proxy certificate limitation, must prevail.

The Functional Requirements (FR#) and software components that are minimally needed to implement a working pseudonymity system are summarized below:

FR1 The pseudonymity system should authenticate all requests relying on existing Grid security mechanisms, i.e. SSL/TLS communication and X.509 certificates.

FR2 The communications in all interactions should be protected with authenticated and encrypted SSL/TLS channels.

FR3 Pseudonym credentials should be modeled as X.509 certificates.

FR4 Additional individual attributes (role, group etc) should be modeled as Attribute Certificates.

FR5 Pseudonym credential requests should be honoured to entities authenticating with user long-term X.509 certificates and proxy X.509 certificates.

FR6 The credential issuance of the pseudonymity system must not include manual operations, in other words, it should operate in the same manner as an online CA.

FR7 Auxiliary authorization services must offer functionality that allows the pseudonymity system to judge whether additional user attributes to be included in the pseudonym credential identify the user uniquely.

FR8 A pseudonym credential requested with a credential having rights limitations, must, if granted, return a pseudonym credential with identical limitations. A limited proxy is an example of such a credential.

FR9 The pseudonymity system must not create ways to circumvent the security of the system.

V. Discussion

In this section we discuss some architectural and functional issues of the pseudonymity system before describing the alternative solutions in the next section.

A. *On the architecture of the pseudonymity system components*

We outlay our solution alternatives using three independent components: the pseudonymity service, the online CA service and an Attribute Authority. Having this separation allows us to benefit from existing and well-tested attribute authority and online CA software. Also, we avoid reimplementing their functionality within the pseudonymity service. In this setting, the pseudonymity service acts as a Registration Authority authenticating the users and validating their requests before forwarding the requests to the online CA. The pseudonymity service fulfills the traceability requirement 2 by maintaining records of the pseudonyms issued to the authenticated users. Having an automated online CA ensures the timely delivery of pseudonym credentials in accordance with FR6.

Due to the short-life time of pseudonymous credentials, we anticipate that certificate revocation functionality is not vital. However, such functionality can be added later non-intrusively if deemed necessary. This is similar to the fact that Certificate Revocation Lists (CRLs) are not used against individual proxy credentials, but against the underlying long-lived ones.

It is likely that a virtual organization offering Attribute Authority services will also provide a pseudonymity service for its user community. Therefore, even though this division into separate components apparently violates requirement 4, we consider this an insignificant relaxation of constraints.

B. *On generating the pseudonymous identities*

In principle, the pseudonymous identifiers could be requested by the user, generated by the pseudonymity system, or, possibly even the online CA in some cases, depending on the implementation. This process could also include some modifications to the request for supporting different protocols and message formats in the communication with the online CA by one single message schema between the client and the pseudonymity system. In this paper we limit ourselves into stating that all of the above options can be made sufficiently secure for the purposes of pseudonymity and postpone the final choice.

After this first stage the Grid user has a pseudonymous credential with a random DN. This credential does not necessarily have the user's attributes attached as these are granted by their Attribute Authorities. Therefore, in order for the users to gain access to additional resources enabled by their attributes, the attributes have to be retrieved on a second stage from a trusted Attribute Authority that has the knowledge of who possesses the pseudonymous identity. Variations to this two-step creation of a pseudonymity credential are described in further detail in the next section.

C. *On required modifications to Attribute Authority components*

In all the scenarios described in the next section, the internal data models of Attribute Authorities need to be extended to associate pseudonyms as aliases of real users. Upon reception of an attribute request related to a pseudonym, the Attribute Authorities should return information on the degree of uniqueness of the user attributes as stated in FR7. This is not normal feature of Attribute Authorities and thus implies slight changes to the ones supported. The threshold degree and what is done when this limit is reached need be configurable on a pseudonym service basis. Ultimately it is however the task of the VOs to attempt to ensure the user groups remain sufficiently large to prevent this from occurring. Also the Attribute Authority must have a way of cleaning up the old expired pseudonyms so that the pseudonymity list doesn't become unmaintainably large over time.

VI. Solutions

We discuss the pros, cons and differences of three alternative architectures. The last one which we propose for implementation, is in our understanding the most advantageous architecture.

A. First scenario: All-in-one pseudonymity service

In the first alternative fulfilling the requirements of chapters III and IV, the Grid user communicates only with the pseudonymity service to acquire a complete pseudonym credential including the desired set of auxiliary user attributes. First, as shown in Fig. 1, the Grid user requests a pseudonymous credential from the pseudonymity service. Next, the pseudonymity service contacts the user's Attribute Authority and the user attributes are added to the pseudonym credential request. This request is then passed on to the online CA for signing. Once signed, the pseudonymity service returns the now valid credential back to the user.

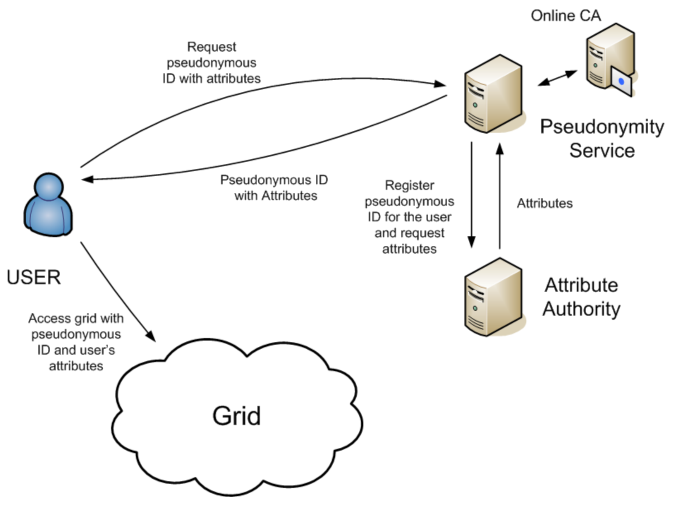

Online CA
Request pseudonymous ID with attributes
Pseudonymity Service
Pseudonymous ID with Attributes
Register pseudonymous ID for the user and request attributes
Attributes
USER
Attribute Authority
Access grid with pseudonymous ID and user's attributes
Grid

Fig. 1. All-in-one pseudonymity service. The pseudonymity service contacts both the Attribute Authority and the online CA.

This scheme has some disadvantages in that the pseudonymity service will have to handle attribute requests from the Grid user. If the attribute requests are not handled by the pseudonymity service, then the Attribute Authority will have to return all attributes with the pseudonymous credential for that particular user each time. Another complication is that the pseudonymity service needs to implement all the APIs, communication and error handling of the Attribute Authorities that need be supported. If any of these change, the pseudonymity service has to be changed accordingly.

An advantage of this scheme is that the pseudonymity service has the possibility to decline to issue a pseudonymous identity altogether if the requested user attributes are uniquely identifying. The pseudonymity service would basically replace the Attribute Authority so when the user wants a short lived proxy, he vould use the pseudonymity service instead of Attribute Authority resulting in pseudo proxy instead of normal proxy.

B. Second scenario: Pseudonymous identities with user-driven identity registration

By making the users themselves register their pseudonymous identities to the Attribute Authorities as shown in Fig. 2, we eliminate the need to support this registration in the pseudonymity service. In addition, the users will request their auxiliary attributes directly from the Attribute Authorities exactly as in current Grid middleware. This latter point removes the burden of supporting the different APIs of Attribute Authorities in the pseudonymity service. Both features simplify the API and the internal architecture of the pseudonymity service.

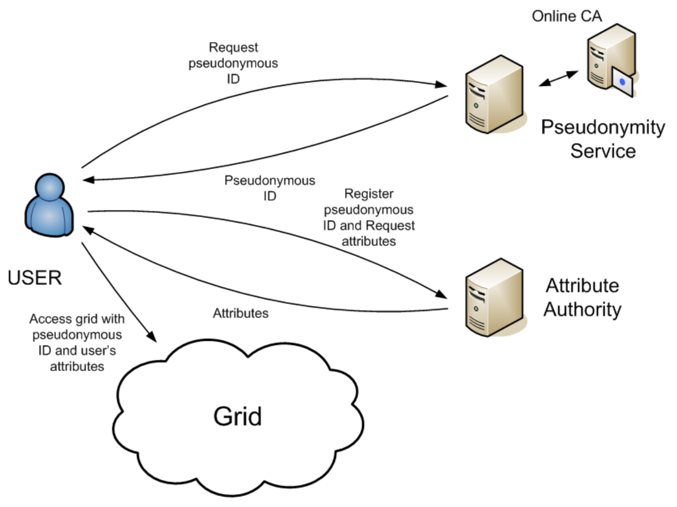


Fig. 2. Pseudonymous identities with user-driven identity registration.

This two-phase approach differs technically from the first scenario in that the attributes will have to be included in a proxy certificate derived from the pseudonymous certificate. From the end user perspective this is irrelevant since it is functionally equivalent to ordinary proxy certificates used for single sign-on.

The pseudonymous credential is associated to the real user in the Attribute Authority by having the user pass this mapping directly, as shown in Fig. 2. This method poses a possibility for misuse as the user can pass somebody else's pseudonym thus causing mismatch between the mapping in the online CA and that in the Attribute Authority. Complex and rigorous controls would have to be implemented for this method.

In contrast to the first scenario, the pseudonymity service has no possibility to discern whether the requirement of non-uniquely identifying user attribute sets (FR7) is met. This responsibility is pushed entirely to the Attribute Authorities.

C. Third scenario: Pseudonymous identities with automatic identity registration

A second method of mapping the pseudonym to a real user within the Attribute Authority is for the pseudonym service to contact the Attribute Authority and register the pseudonymous credential on behalf of the user. This prevents the Grid user from tampering with the pseudonymous to real user credential mapping in the Attribute Authority. Once the Attribute Authority has this mapping, the Grid users can contact the service and request their user attributes to be added to the pseudonymous credential just like they do with their real credentials.

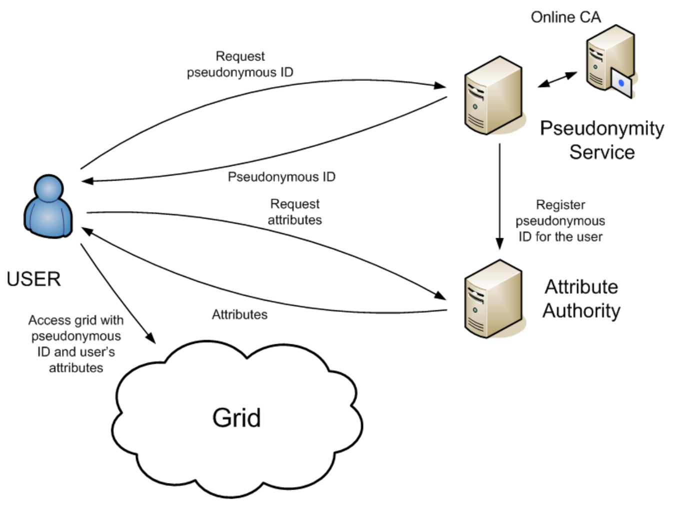


Fig. 3. Pseudonymous identities with automatic identity registration.

In addition to the simplification of the pseudonymity service described in the second alternative, this final solution has the benefit of removing the possibility of tampering with the pseudonymous identity. Another benefit is that the existing client software implementations can be used for requesting the attributes from the Attribute Authorities by pointing them to use pseudonym credentials instead of the real user credentials.

This solution is the one deemed most promising as it takes the best advantage of the existing systems incurring the least changes to them. Also from the user point of view the system has a distinct pseudonymity step and after that the Grid systems operate the normal way.

VII. Conclusions

This paper describes the requirements and initial design of a general pseudonymity service that provides pseudo anonymous access to the Grid. The system allows the users to employ their attributes to access the Grid while hiding their true identity.

The repercussions of hiding the user's identity are hard to determine without getting real world experience with a prototype. A prototype will also shed light on the magnitude of the information leakage problem (req. 6). Currently, to reduce the risk of leaks both the community of users employing pseudonyms and the mix of actions and operations they perform in a Grid need to be large. Ideally, every action on the Grid would use a different pseudonym and use it only once. This makes it difficult to correlate different actions of any single user. On the other hand, even different one-time pseudonym identifiers may be correlated if they are used from the same IP address and this address is not used by any other pseudonym user. The pseudonymity system may thus only partially counteract the leakage problem. Ultimately, the users themselves are required to actively reduce such risks.

The large groups needed for preventing the correlation of actions to a single user pose also a problem. For example using pseudonyms for file access means that the access has to be based solely on the groups and attributes of the user. If the groups are large, it means that there are many people that have access to the files, thus there is less privacy of the files. Also there is less compartmentalization of users and thus in case of compromise of a user has bigger potential for damage. In the end the group size is a balancing act between the VO needs of identity hiding and resource security.

Also, some legislative concerns must be addressed. For example, in some countries the law expects site administrators to know the real identities of the users. The pseudonymity system does maintain the link, that is obtainable, between the real user identity and the pseudonymous version but this may not be enough for some regulations. Unless some governmental identity escrow is available, this effectively bans the usage of pseudonyms within these jurisdictions.

It is foreseeable that for large-scale deployments the certification policy and the pseudonym user's authentication sequence should be approved by a Grid Policy Management Authority (GridPMA). Another issue in probable conflict with most site policies is the generation of long-term pseudonym credentials for anonymizing long running jobs. Also, a long-term credential implies a higher security risk than an ephemeral one.

VIII. Future Work

The near term work is to implement a prototype of Fig. 3. It will allow us to gain practical experience of the system and the identified problem areas, especially the information leakage problem.

Another important goal to pursue is to ensure the mix of operations and pseudonymous users is sufficiently broad to prevent correlation attacks. Also, the connection source tracking needs to be investigated. To this end, grid portals are ideal: using pseudonyms through a portal effectively prevents IP addresses from being tracked assuming job results are also retrieved through the portal or stored into the grid storage using a pseudonym. We will explore the benefits and drawbacks of including portals into the overall architecture.

ACKNOWLEDGMENT

The authors wish to hereby thank Olle Mulmo, Ákos Frohner and David Groep for fruitful discussions, ideas and feedback contributing to the preparation of this paper.

REFERENCES

[1] EGEE Design Team, "EGEE middleware architecture and planning (release 1)," Tech. Rep., Aug. 2004.

[2] O. Mulmo, "Global security architecture for web and legacy applications," Enabling Grids for E-science in Europe, Tech. Rep., Sep 2005.

[3] J. Hughes and E. Maler, "Security Assertion Markup Language (SAML) 1.1 Technical Overview, Committee Draft," May 2004, http://www.oasis-open.org/committees/security/.

[4] ——, "Security Assertion Markup Language (SAML) 2.0 Technical Overview, Working Draft 04," Apr. 2005, http://www.oasis-open.org/committees/security/.

[5] The Globus Alliance, "GridShib Project Web Site," http://gridshib.globus.org.

[6] V. Welch, T. Barton, K. Keahey, and F. Siebenlist, "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration," in *4th Annual PKI R&D Workshop: "Multiple Paths to Trust"*, NIST Gaithersburg MD, USA, Apr. 2005.

[7] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation," in *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*. London, UK: Springer-Verlag, 2001, pp. 93–118.

[8] J. Camenisch and E. V. Herreweghen, "Design and implementation of the idemix anonymous credential system," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2002, pp. 21–30.

[9] BEA Systems, BMC Software, CA, IBM Corporation, Layer 7 Technologies, Microsoft Corporation, Novell, and VeriSign, "Web Services Federation Language (WS-Federation), Version 1.1," Dec. 2006, http://www.ibm.com/developerworks/library/specification/ws-fed/.

[10] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile," RFC 3820, IETF, June 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3820.txt

[11] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lőrentey, and F. Pataro, "VOMS, an Authorization System for Virtual Organizations," in *1st European Across Grids Conference*, Santiago de Compostela, Spain, Feb. 2003.

[12] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, "A Community Authorization Service for Group Collaboration," in *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 50.

[13] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280, IETF, Apr. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3280.txt

[14] S. Farrell and R. Housley, "An Internet Attribute Certificate Profile for Authorization," RFC 3281, IETF, Apr. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3281.txt

[15] EJBCA, "The J2EE Certificate Authority Web Page," http://ejbca.sourceforge.net/.

[16] J. Basney, M. Humphrey, and V. Welch, "The MyProxy online credential repository," *Software: Practice and Experience*, vol. 35, no. 9, pp. 801–816, July 2005.

[17] P. A. et al., "CREAM: A Simple, Grid-accessible, Job Management System for Local Computational Resources," in *Proceedings CHEP06, Mumbai, India*, 2006.

SOPAS: A LOW-COST AND SECURE SOLUTION FOR E-COMMERCE

Marc PASQUET[1], Delphine VACQUEZ[2], Christophe ROSENBERGER[1]
[1] Laboratoire GREYC: ENSICAEN – Université de CAEN – CNRS
[2] DRI (Department of Industrial Relationships): ENSICAEN
6 boulevard Maréchal Juin, F-14020 CAEN (France).
marc.pasquet@ensicaen.fr
delphine.vacquez@ensicaen.fr
christophe.rosenberger@ensicaen.fr

KEYWORDS
Smartcards, Authentication, Security for E-Business, Commercial and Industry Security.

ABSTRACT

We present in this paper a new architecture for remote banking and e-commerce applications. The proposed solution is designed to be low cost and provides some good guarantees of security for a client and his bank issuer. Indeed, the main problem for an issuer is to identify and authenticate one client (a cardholder) using his personal computer through the web when this client wants to access to remote banking services or when he wants to pay on a e-commerce site equipped with 3D-secure payment solution. The proposed solution described in this paper is MasterCard Chip Authentication Program compliant and was experimented in the project called SOPAS. The main contribution of this system consists in the use of a smartcard with a I^2C bus that pilots a terminal only equipped with a screen and a keyboard. During the use of services, the user types his PIN code on the keyboard and all the security part of the transaction is performed by the chip of the smartcard. None information of security stays on the personal computer and a dynamic token created by the card is sent to the bank and verified by the front end. We present first the defined methodology and we analyze the main security aspects of the proposed solution.

INTRODUCTION

E-commerce is one of the most challenging issue in computer science nowadays. Many e-payment architectures have been proposed in the last decade (Kleist, 2004; Konar, & Mazumdar, 2006; Ekelhart et al., 2007). Nethertheless, very few have been used in real conditions for e-commerce. One major reason is that the defined solution must be supported by major card schemes such as Mastercard or/and Visa. In the following, we present two solutions that were defined within this context.

To limit the risk that the customer can repudiate his payment transaction, a set of companies (Visa, MasterCard, GTE, IBM, Microsoft, Netscape, SAIC, Terisa system, Verisign) have developed, in the eighties one solution call SET (Secure Electronic Transaction). The customer's bank sends him one certificate issued from one CA (Certification authority) of a PKI (Public Key Infrastructure) which is stored on his computer. When he wants to realize a payment on the Web, the customer must sign with the PKI keys (Rennhard et al., 2004).

Another solution for electronic payments is 3D secure (3D–Secure Functional Specification, 2001) developed by VISA and used by MASTERCARD. The commercial trademarks are « Secure Code » for MasterCard and « Verified by Visa » for Visa. The term 3D is the contraction of "Three Domains":

- Acquiring domain (acquiring bank and merchant) ;
- Issuer domain including the customer authentication;
- Interbank field which makes it possible the two other fields to communicate on Internet.

The client realizes his purchase on a merchant's Website that is 3D-secure compliant and click on the payment icon ("MasterCard SecureCode" or "Verified by VISA"). He is invited to enter his card scheme, card number and expiration date. The MPI (Merchant Plug-In) installed in the merchant's website, contacts the Visa or MCI directory to obtain the Internet address of the issuer. Then, using the client's personal computer, the MPI contacts the issuer with a formal PAReq (Payer Authentication Request) message. The client's authentication is under the bank responsibility. When that last task is realized, the bank issuer answers to the MPI of the merchant's website with a formal PARes (Payer Authentication Response) message. The MPI sends an authorization request to the acquiring bank which transmits it to the issuer which will answer with an authorization number. This last dialog is realized to be completely EMV compliant (Europay MasterCard and Visa). The internationally agreed standards for chip payment cards. EMV standards are maintained by EMVCo (EMVCO, 2000). In fact, with 3D-secure, the authentication problem from the customer / merchant domain is replaced by the customer / issuing bank domain (see Figure 1).

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

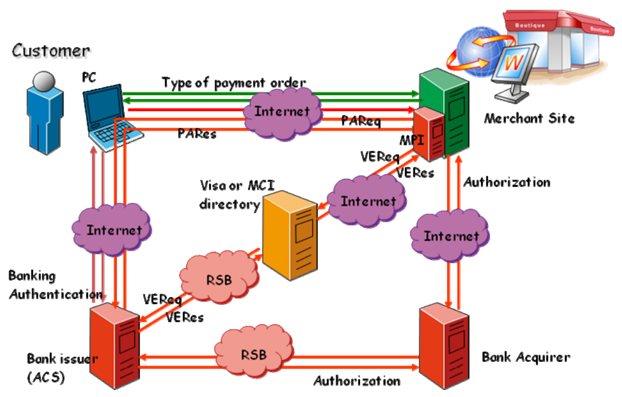


Figure 1: The different communications in 3D-secure payment

The most important challenge today in the 3D-secure architecture for a bank issuer, is to authenticate one client with as many guarantees as possible with the lowest cost. The goal of the SOPAS project in which we are involved in, is to propose new e-transaction architecture. The objective is then to develop a secure and a low-cost solution that can be attractive for banks considering security and commercial issues. We show in this paper some elements on the security of the proposed architecture and the reasons why we think this solution can be supported by major card schemes.

We describe in the next section, the proposed architecture defined within the SOPAS project. In the third section, we focus on the security issues of the proposed solution. Conclusions and perspectives of this work are given in the lat section.

SOPAS PROJECT

The idea of the SOPAS project is to fulfill two services for one client. The first one is the payment on an e-commerce site equipped with a 3D-secure payment solution. The second service deals with remote banking and concerns the use of a personal computer by a client through the web to access to his bank account and to realize different operations (consultation or bank transfers for example). We think that the proposed solution must allow this last service for economical reasons. A bank could be ready to adopt the solution even it will cost some more money if it can offer an additional service for a client. Remote banking is generally a service that is rarely free for a client. A more secure remote banking could be more expensive for a client but will provide also some more secure e-commerce possibilities. We present in the next subsection some more details on targeted services by the SOPAS project.

Objectives

First, we have in particular to fulfill the client needs to use Internet to carry out its remote banking operations. Today, implemented solutions have the main drawback to be based on a password authentication that is not really secure (Pfitzmann et al., 1997). Thus, the SOPAS project has two major objectives:

- to gain the user's confidence;
- to provide a secure solution whose cost of deployment is as cheap as possible.

The client must be able to realize different operations such as those detailed in Table 1:

Table 1: Remote banking operations

Operations	Examples
Standard operations	Consultation, transfer, direct debit …
Credits	Consummation credit, real estate credit…
Assurance	Assurances subscriptions for automobiles, home…
Saving & Shares	Opening of a saving account, stock buying & selling

These transactions are very sensitive if we consider the financial impacts of an uncontrolled use. So, before any access to a banking site, a preliminary authentication is required. When the client is authenticated, the remote banking site proposes all the possible operations.

For certain operations realized by the client, it could be necessary:

- To protect against all alterations, the transaction exchanges between the client and the bank;
- To guaranty the good achievement of the transaction to the client;
- To have the client's agreement proof.

All that objectives (authentication, integrity, good achievement and client's agreement proof) can be realized by question/answer mechanisms:

- The bank generates a question and the client uses a personal device to generate an answer to the bank;
- The bank verifies this answer for the authentication process or to validate the transaction.

Second, one client must be able to make a payment on e-commerce websites in an easy and a secure way. We assume here that the merchant is 3D-secure compliant. This is not a strong hypothesis as it is supported by MASTERCARD and VISA.

General principles

We propose to use three elements in order to guarantee the client's authentication for remote banking and for 3D-secure compliant e-commerce:

1. smartcard: a client is also a cardholder. This smartcard is considered by the banks as very secured and have been personalized by the issuer bank with cryptographic keys to achieve many secure operations. The belonging of this smartcard and the knowledge of the PIN code by the cardholder gives some good guarantees for the bank issuer for its authentication.
2. Personal device: the personal computer is not a secure environment for a strong authentication of the cardholder. We propose to use a separate device as an interface between the smartcard and the personal computer. This personal device must be very secure and low cost. The solution is here to use a box just equipped with a 2x12 figure screen, a 4x4 keyboard, a card reader and no chip. It is the smart card itself which pilots directly the personal device by its I^2C bus and communicates with the personal computer by its USB bus. This solution is very different to the solutions which use a device which is able to compute. Here the “intelligence” and the security of this personal device is completely delegate to the smart card. When the smart card is not connected to the personal device, this one has no secret at all and can be produced everywhere in the word for a very low cost.
3. CAP (Chip Authentication Program) (MasterCard, 2004): CAP provides online chip-based cardholder authentication within the SecureCode™ program. It encompasses the chip application, the terminal, and the issuer server used in the authentication process, and the interfaces between these components. When the smartcard is slip in the personal device, the cardholder is invited to tape is PIN code on the keyboard. The PIN code goes directly to the smartcard and this one computes a token sent to the bank issuer via the personal device, the computer and the network without any modification.

Such a solution makes it possible to guarantee a complete security of the access to remote bank applications via Internet, ready to develop the confidence of the users.

The Figure 2 shows the SOPAS scheme for a remote banking application. The user has a SOPAS smartcard and a SOPAS personal device giving him access to the service. The user proves with the card that he is the legitimate cardholder by entering his PIN code. The card generates a token call “CAP token” which is used as authentication proof by the user to his bank. The token thus generated is transferred from the card to the user’s personal computer via the personal device, then, to the front end of the bank via the Internet network. This device is currently not used to require the user’s assent at the time of a significant banking operation (as in case of purchase stock for example). Indeed, the device would make it possible to seal a transaction; this seal is for the bank a proof of the user’s assent.

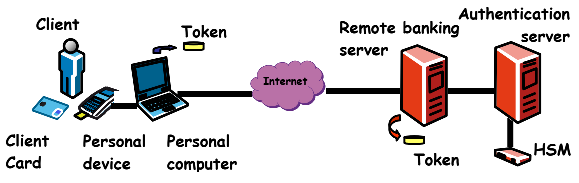


Figure 2: SOPAS solution for remote banking

The SOPAS solution is used mainly to authenticate the user to his bank. This solution, based primarily on the concepts of CAP authentication (MasterCard), should moreover be easily transposable everywhere in the world (see Figure 3).

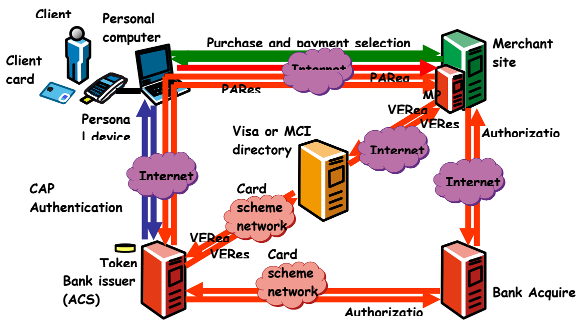


Figure 3: SOPAS solution for e-commerce

Interface protocols

The protocol used for the authentication is of challenge/answer type. The bank sends a random number to the card which turns over a token function of the received random number. This mechanism avoids the attack by replay, contrary to the systems of authentication having a static signature. Figure 4 illustrates the communication protocols used with the interfaces of the various entities intervening in the proposed solution.

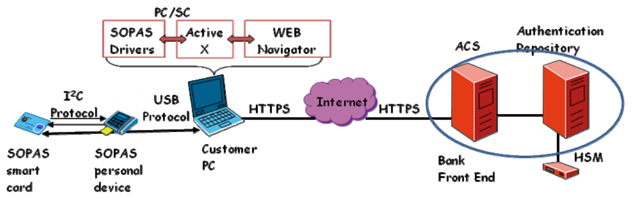


Figure 4: Interface Protocols

We can highlight the different parts of the figure 4:

- The SOPAS Card communicates directly with the personal device (equipped with a keyboard and a LCD screen) by a different interface than which is used to communicate with the personal computer. The protocol used is then I²C (ISO, 1995). This is particularly important from the security point of view of the solution. This bus makes it possible the card to interact directly with its cardholder by presenting him some information via the LCD screen and while requiring some information (like his PIN code) via the keyboard of the personal device. These two operations thus do not require the intervention of the computer which is considered as a non secure element.
- The SOPAS card communicates directly with the user's personal computer with USB protocol via the personal device.
- The user's personal computer is exchanging information with the front end of the issuer bank using HTTPS protocol because the network is Internet.

Architecture

The following diagram (see figure 5) details the architecture and the relationships between the card and the personal device. We can observe that the USB and I²C bus allows the card, either to communicate with the customer's personal computer via the USB interface, or to communicate directly with the personal device in order to reach its keyboard and its screen.

The second circuit (I²C bus) strongly takes part in the security solution. The CAP token is calculated by the card, after the PIN code verification, then sends via the different devices without any modification and control to the HSM (Hardware Security Module) connected to the Bank Front End. So, only the two secure devices (Card and HSM) are able to calculate or verify the Token.

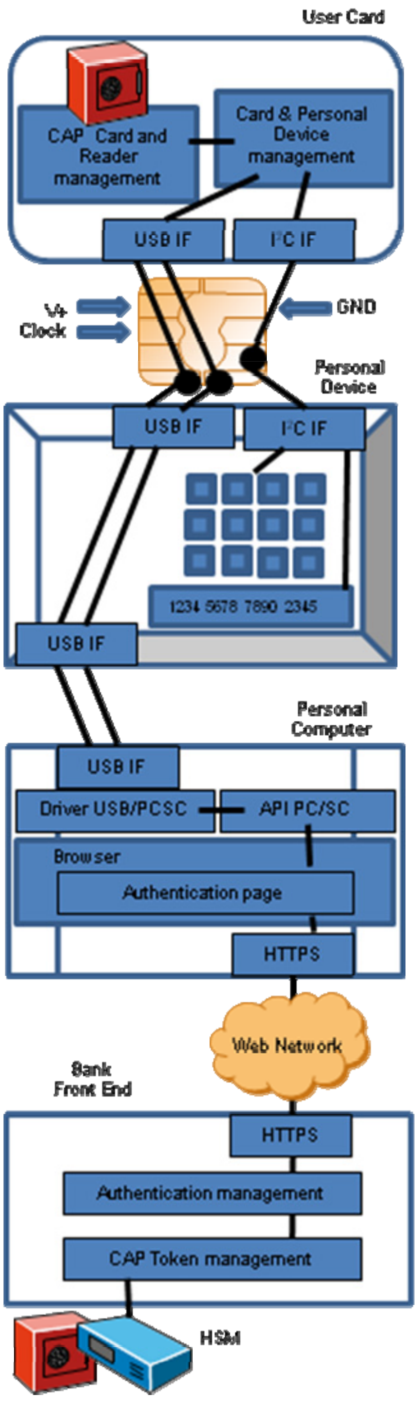


Figure 5: SOPAS architecture

SECURITY ANALYSIS

The objective of the section is to provide an analysis of the SOPAS solution as regarding the security aspects. We study the whole chain in order to determine the potential risks, then to provide some associated countermeasures. This analysis can lead us to possibly modifications of the specifications of the final solution. This is particularly justified by different attacks (phishing and pharming) against remote banking services and the different well known attacks in e-commerce and e-payment. We will endeavor to show that these attacks are completely identified within the framework of this analysis. It will appear that the SOPAS solution can then, in addition to being a solution of customer's authentication by his bank, can be a good solution for the bank authentication by the customer, making thus inoperative the previous attacks.

Methodology

To realize that study, we have used the EBIOS method (DCSSI, 2004). The card operating system answers the safety requirements evaluated according to common criteria (ISO, 2006). During the personalization of the card, the later remote applet loading is blocked. The card and the personal device are delivered by the bank, and the card delivery follows the standard bank card protocol (security requirement) and is delivered in a face to face situation by the bank. The delivering of the PIN is sent to the cardholder by the standard PIN mailer procedure.

Due to its cost, the personal device is an object which cannot be repaired and which is the subject to a standard exchange in the case of problems (in that eventuality the material is destroyed). The cardholder uses the SOPAS architecture in a personal environment and known conditions as standard use (for example without a company network environment…). The personal computer operating system is an area of risk whose protection is out of the study perimeter. The remote banking server (software and hardware) follows completely the security bank requirements. The bank is supposed to have correctly dimensioned and protected its architecture against mass attacks. The contract aspect between the cardholder and the bank must be reviewed by the bank lawyer and are not covered by this study. The SOPAS Smart card is not only a debit or credit card but includes also a CAP capability.

Results

The perimeter includes the following security domains:

- The user,
- The SOPAS smart card,
- The personal device (with its screen and keyboard),
- The link between the personal device and the client personal computer,
- the client personal computer,
- The bank server,
- The link between the bank server and the client personal computer.

The components, directly concerned by the SOPAS solution, appear in the top left hand in Figure 6. The total perimeter of the study is represented by an ellipse in Figure 4. The red entities inside the perimeter are those whose risks are excluded by the assumptions or whose countermeasures do not concern directly the SOPAS solution. For example, the SOPAS solution cannot ensure that the client personal computer is free from any virus software. In the same way, SOPAS cannot ensure that remote banking server is suitably configured, dimensioned... Nevertheless, for the red elements belonging to the perimeter, the analysis will be able, if necessary, to propose a countermeasure.

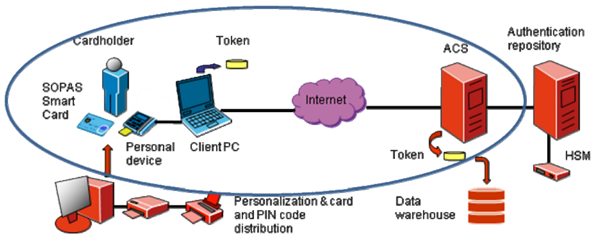


Figure 6: Study perimeter

The perimeter of this study integrates the data processing sequence of the authentication, from the card to the interface of the banking server. Before using the SOPAS smart card, procedures of personalization and distribution are necessary. Although, these last do not belong to the perimeter of the SOPAS solution.

The study of the vulnerabilities realized enables us to formulate a list of risks incurred by the essential elements. The transformation of these risks in scenario makes it possible to better apprehend them and judge their gravity. In this study, we formulate 19 risks. The majority of them concerns the banking data of the user or the technical information allowing the authentication of the customer by his bank.

The incurred risks are:

- The lost of availability ;
- The usurpation of identity ;
- The break of the RSA Keys of the SOPAS smart card (Anderson, 1994) ;
- The deterioration of banking data ;
- The disavowal of action ;
- The right abuse ;
- The divulgation ;
- The illicit processing of data.

During this study, a certain number of threats were identified. The threats which were retained are those which have a direct impact on the authentication mechanism. Additional threats, mainly on the remote banking server (except authentication function) were sometimes retained because it will have been judged that the SOPAS smartcard and the SOPAS personal device could thwart these last. They are mainly the threats and risks induced by the use of a personal computer to which remote banking services cannot grant its confidence. Indeed, it is not rare that the computer has been infected by a Trojan horse and became victim of the technique known as of the pharming.

It was shown during the study that the SOPAS solution makes it possible to cover the risks thus identified by associating to him a functionality of checking to a banking server certificate. That prohibits a fraudulent site to be recognized as being the bank. The user's personal computer not being confident, it is of primary importance so, on one hand, the checking of the server certificate must be embedded in the smartcard and, and on the other hand, the result of this checking must be shown on the personal device screen.

Finally, the risk of disavowal an action was retained because the authentication of a user does not have any value of assent on an action realized between the beginning and the end of connection. This implies the need for the user to sign each remote banking operations (of a sufficient amount). The signature functionality is in fact already present in the SOPAS smartcard but is just used for the user authentication by the bank.

This analysis also showed that, so far as we suppose that the user personal computer is safe (what is not the case but that nevertheless is posed like assumption), the encryption of the communications between the SOPAS smartcard and the user personal computer is not necessary. Indeed, the messages forwarding between these two devices are challenge/answer type, and are secured by that way. Coding from beginning to end would be a solution to mitigate the vulnerability of the personal computer which, by the presence of the malevolent programs, could deteriorate the banking data. This solution is however not realistic since at one time or another, the banking data must be posted on the screen of the personal computer.

To conclude this part, the SOPAS smartcard decreases the risks induced by the potential vulnerabilities of the personal computer. Indeed, the secrecies of connection of the user cannot be recovered any more by a simple keylogger or other spyware and attacks it by replay is not more exploitable. The use of a certificate embedded in the card and the checking of the bank certificate by the SOPAS smart card could further decrease the risks induced by phishing and pharming techniques. Nevertheless, the use of a personal computer that is not controlled (by the bank) remains the Achilles' heel of this service. Recurring problems here are found: how to protect data in an hostile environment?

CONCLUSIONS AND PERSPECTIVES

The SOPAS solution is made up of a personal device (card reader, screen and keyboard pilot via the I^2C bus by the card) and a smart card (Multi applicative card with the embedded SOPAS solution and standard EMV), the cost of the card is a little bit more expensive than a standard EMV chip card (6 to 8 €) but the personal device is very cheap (10 to 20 €). This makes it possible for the bank to deliver cards and personal devices to their clients interested for secure remote banking services and e-commerce.

Thus, the equipped user is able to generate a "CAP token" that he transmits to the bank like an authentication value, when he wishes to reach his remote banking services or to pay on the Web. The bank is convinced to deal with the good person because the smartcard, before generating the token, requires from the customer to enter his PIN code (known only by the card and the card holder), thus resolving the problem of the CAP token generation.

The security analysis of that solution shows that if we consider the limits created by the use of a unsecure personal computer, the SOPAS approach is a very good and secure solution compared to is deployment price.

There are some perspectives of this work. Two main changes are possible in order to limit the possibility for the user to repudiate his action:

1. To oblige the user to sign each remote banking operations (of a sufficient amount).
2. To use CAP Token generation options. In the Cap protocol, it is optionally possible to include the transaction amount and currency in the CAP transaction. This option is indicated by a flag in the card application, bit 8 of the IAF (Internet Authentication Flags).
3.

ACKNOWLEDGMENTS

Authors would like to thank all SOPAS project members: Alliansys, Credit Mutuel, Cartes Bancaires, Fime, Gemalto, the Basse-Normandie Region, and the French Ministry of Industry (DGE), for their kind cooperation.

REFERENCE

Anderson, R. (1994) Why Cryptosystems Fail. Communications of the ACM. pp. 32-41 ftp://ftp.cl.cam.ac.uk/users/ rja14/wcf.ps.gz.

DCSSI (2004) EBIOSV2: expression of needs and identification of security objectives

Ekelhart, A., Fenz, S., Tjoa, A.M., & Weippl, E.R., (2007) Security Issues for the Use of Semantic Web in E-Commerce. BIS 2007, LNCS 4439, Springer-Verlag, pp. 1–13

EMVCO (2000) EMV 2000 specifications. http://www.emvco.com/specifications.cfm

ISO (2006) ISO/CEI 15408 Version 3.1 Common Criteria for Information Technology Security Evaluation.

Kleist, V.F., (2004) A Transaction Cost Model of Electronic Trust: Transactional Return, Incentives for Network Security and Optimal Risk in the Digital Economy. Electronic Commerce Research, vol. 4, pp. 41–57

Konar, D. & Mazumdar, C. (2006) An Improved E-Commerce Protocol for Fair Exchange. ICDCIT 2006, LNCS 4317, Springer-Verlag, pp. 305–313.

ISO 7816 (1995) Standardization of smartcards

MasterCard (2004) Chip Authentication Program – Functional Architecture

Pfitzmann, A. (1997) Trusting Mobile User Devices and Security Modules. Computer, pp. 61-68.

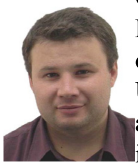
Rennhard, M., Rafaeli, S., Mathy, L., Plattner, B., & Hutchison, D. (2004) Towards Pseudonymous e-Commerce. Electronic Commerce Research, Springer, vol. 4, pp. 83-111.

Visa Corporation. (2001) 3D–Secure Functional Specification, Chip Card Specification v1.0.

AUTHOR BIOGRAPHIES

Marc PASQUET is an assistant professor at ENSICAEN, France. He obtained his Master degree from ENSAM (Ecole Nationale Supérieure des Arts et Métiers) in 1977. He worked for 13 years for different companies belonging to the signal transmission field and 15 years for the banking sector in the field of electronic payment. He joined ENSICAEN (National Engineer School of Caen in France) in 2006 where he is now leading researchs in the field of electronic payment.

Delphine Vacquez is a Research & development engineer at ENSICAEN, France. She obtained her Master degree in 2006 from the University of Caen (computer science department). She has been working in the field of secure and contacless e-payment.



Christophe Rosenberger is a Full Professor at ENSICAEN, France. He obtained his Ph.D. degree from the University of Rennes I in 1999. He works at the GREYC Laboratory. His research interests include evaluation of image processing and biometrics. He is involved recently on e-transactions applications.

Performance of Security Mechanisms in Wireless Ad Hoc Networks

Matthias Becker, Martin Drozda and Sven Schaust
FG Simulation und Modellierung, Institute of Systems Engineering, G. W. Leibniz University of Hannover
Welfengarten 1, 30167 Hannover, Germany.
Email: {xmb,drozda,svs}@sim.uni-hannover.de

Abstract— **Adding security mechanisms to computer and communications systems without degrading their performance is a difficult task. This holds especially for wireless ad hoc networks that, due to their physical and logical openness, are easier to attack than wired networks. Additionally, such networks are expected to have less resources in terms of computational capabilities and must also rely on batteries for power supply.**

We investigate the impact of two misbehavior detection mechanisms based on Neural Networks and Artificial Immune Systems. In the performance analysis we assume that wireless devices, in many scenarios, must work with extremely limited computational resources. An example of such a scenario are sensor networks. This implicates that the security system of choice has to be very efficient in order not to disturb the normal wireless device operation.

Index Terms—**Artificial Immune Systems, Neural Networks, Sensor Networks, Misbehavior Detection.**

I. Introduction and Motivation

Adding security mechanism to computer and communication systems results in their decreased performance. The decrease can even result in a complete system failure caused by the security mechanism, e.g. an email server might be forced to stop some services because of its spam filter exhausting the processor.

Since security mechanisms can have an impact on the quality of service of stationary systems with huge computational and power resources, the effect can be much more severe in computationally limited and energy constrained systems.

In this work we consider wireless sensor networks, which are a specific kind of ad hoc wireless networks. Ad hoc networks have no fixed infrastructure (as opposed to wireless clients that connect to a fixed base station). Typical devices employed in an ad hoc network are laptops, PDAs, mobile phones. A sensor network is a collection of small wireless devices, or sensors, that are able to monitor the deployment area. The sensors then transmit all measured information to a collection station (sink). Such measured information is, in general, forwarded by intermediate sensor that lie between the information acquisition sensor and the collection station. Sensor networks are expected to find applications in many engineering as well as military or rescue scenarios.

A typical security problem in sensor networks is to detect and isolate misbehaving nodes. The misbehavior might be a malfunction due to the damage of a sensor node, or a malicious attempt to disturb or misuse the functions of the sensor network. The nature of a misbehavior is rather unforeseeable, thus an intelligent approach is needed that would also respond to new kinds of attacks. This is in contrast to employing hard-wired attack signatures that do not offer the flexibility needed. Additionally, there is usually not enough memory on sensor nodes to include any attack signature database (e.g. virus signatures).

Approaches for misbehavior detection in wireless ad hoc networks based on Artificial Immune Systems (AIS) have been presented in [9], [14] and with focus on sensor networks in [22]. An approach based on Neural Networks (NN) can be found in [23].

Both approaches have been compared in [16]. NN have been successfully used in pattern matching and for detection in various fields, among them also detection of user anomaly [18], [19] and detection of network intrusion [17]. There exist also flavors of NN where their training continues during operation which should be valuable for highly dynamic systems such as ad hoc networks.

In [23], NN have been evaluated for misbehavior detection in sensor networks and a comparison to AIS with respect to the detection quality has been done.

In this work, the scenarios used in [23], [22] and [24] will be the base for this study of the impact of the security mechanisms (AIS and NN) on system performance. We will study the computational costs of the learning phase as well as the computational resources needed in the detection phase, and how both security mechanisms affect the performance of a sensor node.

This document is structured as follows. First, we will shortly review the mechanism of AIS and NN. Second, an outline of the scenario will be given and third, the detection performance of AIS and NN as well as the computational resources needed for the detection will be discussed.

II. Artificial Immune Systems

The Human Immune System is able to distinguish between what is usually is part of the human body (self antigen), and everything else (non-self antigen), which is usually some kind of foreign agent, be it something mechanical (splinter etc.) or a disease (virus, bacteria).

Inspired by the Human Immune System, Artificial Immune Systems have been developed and have already been considered to be a viable means for misbehavior detection in com-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

munication networks, and especially in flexible environments such as ad hoc and sensor networks.

An Artificial Immune System basically can be divided into two parts, namely a learning phase and a testing phase. In the learning phase the system uses a so called self-set, which represents the expected behavior, to create a set of anomaly detectors. During the testing (or detection) phase, this detector set will be used to detect anomalies. It is necessary that any detector is unable to match a normal behavior. Otherwise *false positives* may occur. A false positive is a detected anomaly which is caused by an incomplete self-set during the learning phase. We will discuss the process of learning in more details in the following subsection. The testing phase is the phase in which the detection actually happens. For an observed time window the system measures different network and traffic features for the neighborhood of a single node and computes an antigen. Each computed antigen is then matched against the complete detector set of the particular single node.

A misbehavior detection system for ad-hoc wireless networks based on AIS has been introduced in [14], [15]. The authors also suggest to use a co-stimulation for misbehavior classification in the form of a danger signal. The function of the danger signal is to inform nodes on a forwarding path about deterioration in terms of a quality of service (QoS) measure. The signal is sent from the connection source to the connection sink, thus propagating QoS information along the connection route. Danger signals were introduced in [3].

One of the earlier proposals of AIS for misbehavior detection was given in [9]. The authors describe an AIS able to detect anomalies at the transport layer of the OSI protocol stack, considering only TCP connections in a stationary wired network. They define the normal network behavior (self) as normal pairwise TCP connections. Each detector is represented as a 49-bit string. The pattern matching is based on r-contiguous bits with a fixed $r = 12$.

In [12] the authors discuss a network intrusion system that aims at detecting misbehavior by capturing TCP packet headers. They report that their AIS is unsuitable for detecting anomalies in communication networks. This result is questioned in [4] where it is stated that this problem occurs because of the specific choice of problem representation and also due to the choice of matching threshold r for r-contiguous bit matching.

In [22] an AIS approach, measuring network features at the link and network layer of the OSI stack, has been studied for sensor networks. The authors use a fixed ad hoc sensor network scenario with approx. 1700 nodes, of which 236 are misbehaving, and 10 concurrent connections with a constant bit rate traffic. The implemented misbehavior in their experiments is probabilistic packet dropping. They conclude that their AIS is capable of detecting such a misbehavior in the described environment with about 80% accuracy.

A. Learning Mechanism of Artificial Immune Systems

The process of T-cells maturation and selection in the thymus is often used as an inspiration for learning in AIS. In AIS, the characteristics of self and non-self can be represented as bit-strings. The role of detectors is to detect non-self antigen. A popular algorithm for matching detectors and non-self antigen, if represented as bit-strings, is the r-contiguous bits matching rule. Two bit-strings of equal length match under the r-contiguous matching rule if there exists a substring of length r at position p in each of them and these substrings are identical.

In order to generate a set of working detectors, pseudo random detectors are created and tested with self. Only if a detector does not match any self-antigen, the detector is added to the detector set. This generate-and-test approach (*negative selection*) described above is analyzed in [6]. They assume that both self and non-self sets, as well as detectors can be modeled as bit-strings of length l. Let the size of the self set be N_S, the probability that a randomly chosen detector and a string from the self set match be P_m and the probability that a string from the non-self set is not matched by any detector be P_f. Then the time and space complexity of this algorithm for a fixed matching probability P_m is $O(\frac{-ln(P_f)}{P_m(1-P_m)^{N_S}}N_S)$ and $O(lN_S)$, respectively. This algorithm requires that the number of required candidate detectors is exponential to N_S. The advantage of this algorithm is its simplicity and good experimental results in cases when the number of detectors to be produced is fixed and small [14]. A review of other approaches to detector computation can be found in [2].

III. Neural Networks

Since Neural Networks and the learning via back-propagation algorithm are well known, we only shortly review the way Neural Networks work here. Details can be found in standard textbooks such as [20].

A. Sketch of NN Algorithm

NN learn a function from an input vector to an output vector. In the so called learning phase input vectors are presented to the NN and an output vector is calculated. Depending on whether the output is correct or not, the weights in the NN are corrected using the back-propagation algorithm in order to achieve the right output. These procedure is repeated until for all input vectors, the right output is calculated (within some epsilon error interval).

The data is divided in a training set and a test set. In the learning phase, only the training set is used. The test set is used in order to evaluate the ability of the NN for *prediction* of unknown input.

Without these two sets there is the danger of creating a NN that works perfect on the set used for training (100 percent correct classification of input vectors), but fails when unknown input is presented. A similar behavior is often caused by so called over-fitting, see e.g. [21] for a mathematical analysis of the problem.

IV. Sensor Networks

The sensor network simulation scenario used here is described in detail in [22], so we only give a sketch of the scenario here.

The sensor network consists of 1718 nodes with a radio radius of 100m. The distribution of the nodes over a square area of 2,900m×2,950m is a snapshot of the nodes moving by the random waypoint mobility model. The motivation in using this movement model and then creating a snapshot are based on results on structural robustness of sensor networks [5]. The traffic in the network is generated by 10 CBR (Constant Bit Rate) connections. The connections were chosen so that their length is $\sim$7 hops and so that these connections share some common intermediate nodes. For each packet received or sent by a node the following information has been captured: IP header type (UDP, 802.11 or DSR [10] in this case), MAC frame type (RTS, CTS, DATA, ACK in the case of 802.11), current simulation clock, node address, next hop destination address, data packet source and destination address and packet size. We refer the reader to [11] for more information on sensor networks. We chose source and destination pairs for each connection so that several alternative independent routes exist; the idea was to benefit from route repair and route acquisition mechanisms of the DSR routing protocol, so that the added value of AIS based misbehavior detection is obvious. Glomosim [7] was used as simulator, with 4-hours of simulated traffic. We choose to collect traffic information within 28 non-overlapping 500-second windows in our simulation. In each 500-second window self and non-self antigens were computed for each node. The experiment was repeated 20 times with independent Glomosim runs using different random seeds.

A. *Encoding of genes*

From the captured simulation data the following measures have been calculated and then coded into genes:

MAC Layer:

#1 Ratio of complete MAC layer handshakes between nodes s_i and s_{i+1} and RTS packets sent by s_i to s_{i+1}. If there is no traffic between two nodes this ratio is set to ∞ (a large number). This ratio is averaged over a time period. A complete handshake is defined as a completed sequence of RTS, CTS, DATA, ACK packets between s_i and s_{i+1}.

#2 Ratio of data packets sent from s_i to s_{i+1} and then subsequently forwarded to s_{i+2}. If there is no traffic between two nodes this ratio is set to ∞ (a large number). This ratio is computed by s_i in promiscuous mode. This ratio is also averaged over a time period. This gene was adapted from the watchdog idea in [13].

#3 Time delay that a data packet spends at s_{i+1} before being forwarded to s_{i+2}. The time delay is observed by s_i in promiscuous mode. If there is no traffic between two nodes the time delay is set to zero. This measure is averaged over a time period. This gene is a quantitative extension of the previous gene.

Routing Layer:

#4 The same ratio as in #2 but computed separately for RERR routing packets.

#5 The same delay as in #3 but computed separately for RERR routing packets.

Encoding of self and non-self antigens was done as follows. Each gene value was transformed in a 10-bit signature where each bit defines an interval[1] of a gene specific value range. We created self and non-self antigen strings by concatenation of the defined genes. Each self and non-self antigen has therefore a size of 50 bits. The interval representation was chosen in order to avoid carry-bits that make the binary representation less compact.

Simulation runs were done for one of $\{10, 30, 50\%\}$ misbehavior levels (packet dropping) and "normal" traffic with no misbehavior, so that 'self' could be learned from the normal behavior and then the quality of the misbehavior detection could be evaluated either with NN or AIS, using the genes extracted for the misbehaving nodes for the simulation runs with misbehavior.

V. Training and Detection Quality of AIS

As described in section II-A, for each node in the network an AIS has been set up with detectors obtained via the negative selection algorithm using the genes that encode the traffic characteristics met around this node during normal network operation. Only data from nodes with enough traffic has been used. The experiments in [22] showed that the detection rate is rather independent of the packet thresholds. Packet threshold of e.g. 500 means that a node had at least 500 packets to forward in both the learning and misbehavior (test) phases; this number is measured over the whole 4-hour simulation period. Except for some extremely low threshold values the detection rate stays constant.

Then the genes from the sensor network that includes misbehaving nodes have been presented to the AIS of the single nodes for misbehavior detection. A node is defined to be detected as misbehaving, if it gets flagged in at least 14 out of 28 possible windows. This definition is equivalent (under reasonable assumptions) to saying that the time to detection is double the size of the window, i.e. 1000 seconds in this case. In [22] it is shown that if the misbehavior level is set very low, i.e. 10%, the AIS usually struggles to detect misbehaving nodes, because the traffic pattern of misbehavior is not distinct enough from the noise appearing in normal traffic, where also packets are dropped sometimes. At the 30 and 50% misbehaving levels the detection rate stays solid at about 70-85%.

Beyond the pure detection abilities, in this work we have a closer look at how parameters related to the computational complexity of the AIS algorithm influence the detection rate. Figure 1 shows the impact of r on detection rate. When $r = \{7, 10\}$ the AIS performs well, for $r > 10$ the detection rate decreases. This is caused by the inadequate numbers of detectors used; in general the number of detectors should be doubled when r is increased by one.

Besides r, the number of detectors is the crucial parameter regarding the detection rate and especially computational

[1]The interval encoding of genes is adapted from [14]. This way only one of the 10 bits is set to 1, i.e. there are only 10 possible value levels that it is possible to encode in this case.

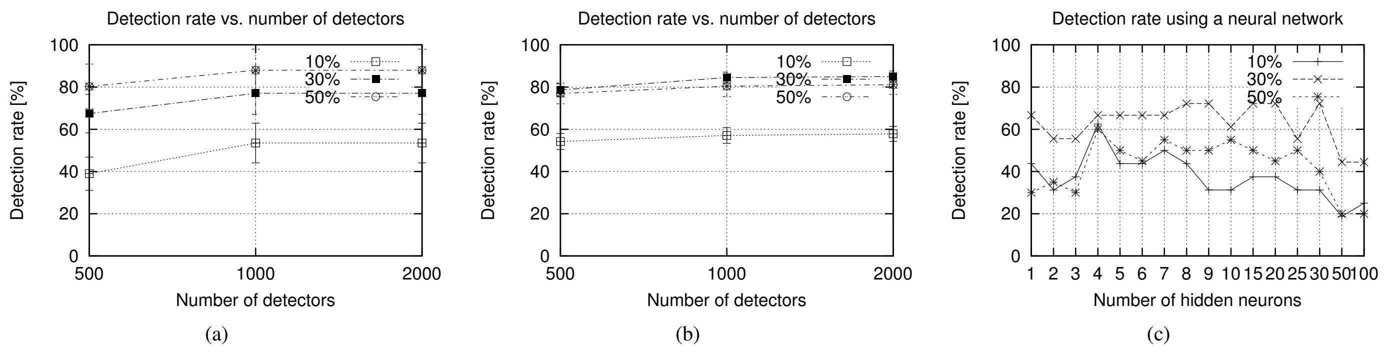


Fig. 2. Detection rate vs (a) number of detectors using 10 connections, (b) number of detectors using 50 connections, (c) the mean number of hidden neurons. All scenarios use a Poisson based traffic model. Packet threshold was 1000 packets.

complexity. Figures 2 (a) and (b) show that for the actual problem, less than 1000 detectors are not sufficient for a good detection rate, 2000 detectors show best success, while more detectors do not improve the detection rate anymore and only increase computational complexity and memory requirements. The figures are based on two scenarios using Poisson traffic [24], one having 10 connections the other 50 connections. We used the same packet threshold value of 1000 packets for both scenarios in order to verify whether a node qualifies for detection or not.

VI. Training and Detection Quality of NN

Back-propagation networks with three layers using the FANN library [8] have been deployed here for misbehavior detection. We used the same representation of good/bad behavior, i.e. a binary vector of length 50. Therefore the Neural Networks had 50 input neurons and only one output neuron (zero indicating good, one indicating bad behavior). The data from the simulation experiments has been divided into a training and test set (approx. $\frac{2}{3}$ training, $\frac{1}{3}$ test). Only data from experiments/nodes has been used, where enough data was present (a bad node cannot develop a bad node pattern, if he is outside the traffic), and where the data was unambiguous.

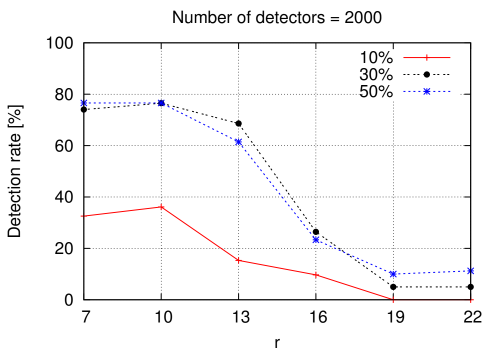


Fig. 1. Detection rate vs r-value of the matching algorithm. Packet threshold was 1000 packets.

The main parameters of a Neural Network is the number of hidden layers and the number of neurons in the hidden layers. For the problem complexity considered here, one hidden layer is appropriate. The question is, how many neurons the hidden layer should have, in order to be able to distinguish good and bad behavior, but also avoiding over-fitting (when a Neural Network learns the data too exactly, it is not able to succeed on new, unknown data any more, i.e. the ability to abstract/predict is lost). Moreover the number of neurons in the hidden layer is the only free parameter in the Neural Network that has a direct effect on the computational complexity, since the number of input and output neurons are predetermined by the input vectors (genes) and the number of different outcomes. Thus the detection rates have been evaluated for an increasing number of hidden neurons.

A. *Experiments*

Since each training of a Neural Net is an individual process (depending in the random initial weights and the presentation order of the vectors during the learning phase) and results in a different Neural Net finally, several experiments have to be made in order to get a valid result with mean outcome and mean deviation. Thus for each configuration (consisting of a certain number of hidden neurons and a data set) 20 Neural Nets have been trained and evaluated, figure 2 (c) thus showing mean values. The variance of the exact value of the output neuron was very low so that the classification resulting from the exact value of the output neuron (good/bad) showed no variance any more. Figure 2 (c) shows how many percent of the **test** vectors have been identified correctly as good or bad. This shows the prediction power of the Neural Net, that is the correct response to **new** attacks. The training set has been learned almost perfectly (Training stopped when over 95% of the training set had been learned correctly).

If, for comparability with the AIS approach, the training set would also be considered, then the detection rate would be around 90 percent. Note that the training data is recognized correctly with 100 percent.

Regarding the number of hidden neurons (representing the complexity of the Neural Net) we observe, that a medium number of hidden neurons (between four and nine) shows best overall success. This corresponds to the experience, that a certain number of hidden neurons is necessary for the correct analysis of the data, but on the other side, if there are too many hidden neurons, the net overfits the data. That means the training set is recognized correctly (100 percent), but the net does not have the ability to judge about unknown data, since it has just learned exactly the training set. As could be expected, a low level of misbehavior makes it harder recognize the difference between good and bad behavior.

VII. Complexity and Performance of NN and AIS Algorithm

Comparing the abilities of both AIS and NN in terms of detection rate for unknown misbehavior, which is between 40 and nearly 80% depending on the level of misbehavior and other parameters, it can be said that both algorithms perform well fulfilling this task. Both algorithms are suited for misbehavior detection in sensor networks.

For the design of a Neural Network, the number of hidden neurons is the key parameter. We showed that not many hidden neurons are required for a decent detection rate.

The performance concerning detection rate of NN and AIS is comparable. Nevertheless both methods are different in some key aspects when thinking of an application for a real sensor network. An AIS (with no danger signals utilized) can be only trained using the normal operation traffic. A Neural Network however needs also bad behavior to be represented in its training phase. Otherwise the Neural Network might learn, that it should always output 'self' constantly, disregarding the input. A solution to the described problem would be to generate non-self patterns, either by injecting errors in the sensor network or by generating artificial 'non-self' training vectors.

A. Computational Requirements of AIS and NN

Once the AIS and NN are trained, an AIS needs nearly six times more memory resources for its task than a NN (see [25], [26] for complexity issues in NN). In our setup, good detection performance was achieved with $d = 2,000$, where d is the desired number of detectors, and with $h \leq 9$, where h is the number of hidden neurons. Within this setup (bit-string length of 50 bits), the memory needed for AIS is approx. $50 \times d = 100$ kbits $= 12,500$ Bytes. A NN is represented by a weight matrix of size $50 \times h + h \times 1 = 459$ weight values, for $h = 9$. When the weights are coded as `float` data types of size 4 Bytes, the memory requirements is $1,800$ Bytes. We remind the reader, Crossbow Mica2 sensors [1] have 4kB RAM and 512kB EEPROM data memory available; these resources are however shared among all applications and the operating system.

Figure 3 (a) shows the memory requirements in Kilobytes for a given problem size. The problem size is normalized to the case discussed above, that is problem size of one means $h = 9$ number of hidden neurons in the Neural Network case, and $d = 2000$ detectors in the AIS case. It can be seen that for larger problem sizes, the memory requirements for AIS grow to a size which might not be suitable for small sensors anymore, while the memory requirements stay moderate even for a large problem size in the NN case.

The computational requirements for detection in AIS is in worst case $d \times r \times (50 - r)$ operations (bit comparisons) (r being the value from the r-contiguous bits matching). That is $800,000$ comparisons for $r = 10$. The reason is, when an antigen is constructed, there are possibly up to d comparisons with the detectors necessary. The detection in NN can be measured in multiplication and additions: $50 \times 10 + 10 \times 1 = 510$ multiplications plus approx. other 510 additions.

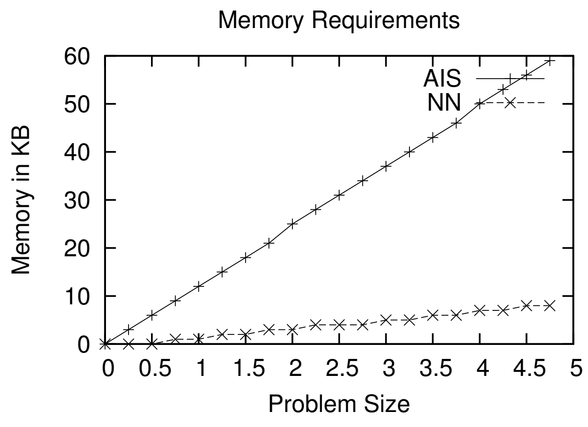
Figure 3 (b) shows the growing number of operations with higher problems size (normalized to the problem above, with constant $r = 10$ in the AIS case). The operations in the AIS are mainly bit comparisons, while in the NN the operations are multiplications and additions. Note that we chose a logarithmic scale on y-axis, since the operational requirements differ by about three orders of magnitude. Since the computational requirements are also proportional to the energy consumption this fact clearly indicates that NN have an advantage in energy constrained sensor networks.

VIII. Conclusions and Future Work

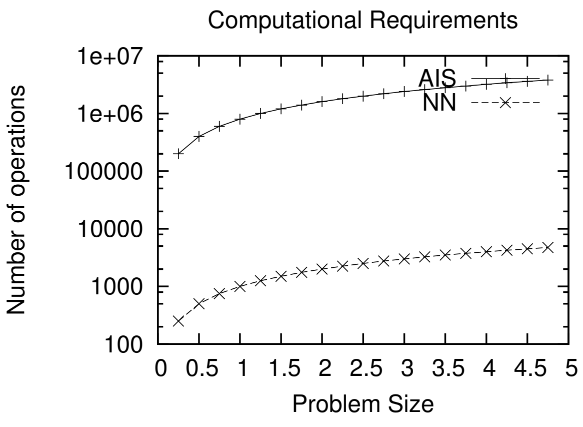
We compared performance and complexity of two approaches known from computational intelligence, when used as a means for misbehavior detection in wireless sensor networks.

We conclude that they meet the memory requirements of to-date sensor platforms which have a memory in the order of hundreds of kilobytes, thus a Neural Net as described above would need less than 1 percent of the memory. Also the computational effort of doing a few hundred operations (additions, multiplications or comparisons) is negligible on sensors with a processor-speed in the order of MHz.

The results in this work suggest that Neural Networks are better suited for sensor networks with restricted resources, because NN use less memory and need far less operations for the detection. While the need of memory of both NN and AIS is negligible compared to the memory of actual sensors, the issue of operations in the detection phase is more crucial. NN need approximately three orders of magnitude less operations for a classification. This means that each detection in an AIS needs much more time, slowing down the operation of the sensor network. While this slow down might be negligible in sensor networks whose operation is usually not time critical, the resulting bigger energy consumption of AIS is more important, and might tip the scale in favor of NN. However both approaches are different regarding the length of the preprocessing phase, memory requirements, speed of computation and the rate of false positives. Both approaches are suitable for misbehavior detection in sensor networks, the decision which approach to choose for a specific sensor network depends on the details of the scenario.



(a)



(b)

Fig. 3. (a) Memory requirements vs problem size (normalized, problem size $1 \sim h = 9$, $d = 2000$), (b) Number of operations vs problem size (normalized, problem size $1 \sim h = 9$, $d = 2000$)

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) under the grant no. SZ 51/24-2 (Survivable Ad Hoc Networks – SANE).

REFERENCES

[1] Crossbow Technology Inc. www.xbow.com

[2] U. Aickelin, J. Greensmith, J. Twycross. Immune System Approaches to Intrusion Detection - A Review. *Proc. the 3rd International Conference on Artificial Immune Systems (ICARIS 2004)*, 2004.

[3] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod. Danger theory: The link between ais and ids. *Proc. International Conference on Artificial Immune Systems (ICARIS'03)*, 2003.

[4] J. Balthrop, S. Forrest, M. Glickman. Revisiting lisys: Parameters and normal behavior. *Proc. Congress on Evolutionary Computing (CEC02)*, 2002.

[5] C. L. Barrett, M. Drozda, D. C. Engelhart, V. S. Anil Kumar, M. V. Marathe, M. M. Morin, S. S. Ravi, J. P. Smith. Understanding Protocol Performance and Robustness of Ad Hoc Networks Through Structural Analysis. *Proc. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2005)*, 2005.

[6] P. D'Haeseleer, S. Forrest, P. Helman. An immunological approach to change detection: Algorithms, analysis and implications. *Proc. IEEE Symposium on Research in Security and Privacy*, 1996.

[7] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A Scalable Network Simulation Environment. UCLA Computer Science Department Technical Report 990027, May 1999.

[8] FANN Library. http://leenissen.dk/fann/

[9] S. Hofmeyr, S. Forrest. Immunity by Design: An Artificial Immune System. *Proc. Genetic and Evolutionary Computation Conference (GECCO-1999)*, 1999.

[10] D. Johnson, D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, Tomasz Imielinski and Hank Korth, Eds. Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[11] H. Karl, A. Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, 2005.

[12] J. Kim, P.J. Bentley. Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection, *Proc. Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, 2001.

[13] S. Marti, T. J. Giuli, K. Lai, M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proc. the 6th annual international conference on Mobile Computing and Networking*, 2000.

[14] S. Sarafijanović, J.-Y. Le Boudec. An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. *Proc. the 3rd International Conference on Artificial Immune Systems (ICARIS 2004)*, 2004.

[15] J.-Y. Le Boudec, S. Sarafijanović. An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. *Proc. Bio-ADIT'04*, 2004.

[16] D. Dasgupta. Artificial Neural Networks and Artificial Immune Systems: Similarities and Differences. *Proc. IEEE Systems, Man, and Cybernetics Conference (SMC)*, pp. 873-878, 1997.

[17] M. Moradi, M. Zulkernine. A Neural Network Based System for Intrusion Detection and Classification of Attacks. *Proc. 2004 IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*, 2004.

[18] J. Ryan, M.J. Lin, R. Miikkulainen. Intrusion Detection with Neural Networks. *Advances in Neural Information Processing Systems*, vol.10, pp. 943–949, MIT Press, 1998.

[19] H. Debar, M. Becker, D. Siboni. A Neural network component for an intrusion detection system. *Proc. IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 240-250, 1992.

[20] S. Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall, 1999.

[21] A.B. Owen. Overfitting in Neural networks. In *Computing Science and Statistics. Proc. 26th Symposium on the Interface*, pp. 57–62, Interface Foundation of North America, 1994.

[22] M. Drozda, S. Schaust, H. Szczerbicka. Is AIS Based Misbehaviour Detection Suitable for Wireless Networks? *Proc. IEEE Wireless Communications and Networking Conference (WCNC '07)*, pp. 3130-3135, Hong Kong, 2007.

[23] M. Becker, M. Drozda, S. Jaschke, S. Schaust. Comparing Performance of Misbehavior Detection Based on Neural Networks and AIS. IEEE Systems, Man, and Cybernetics Conference (SMC'08), Singapore, submitted.

[24] S. Schaust, M. Drozda, H. Szczerbicka. Impact of Packet Injection Models on Misbehaviour Detection Performance in Wireless Sensor Networks. *Proc. 3rd IEEE International Workshop on Wireless and Sensor Networks Security (WSNS)*, 2007.

[25] A. Engel. Complexity of learning in artificial Neural networks. *Theor. Comput. Sci.*, vol. 265, no. 1, pp. 285-306, 2001.

[26] M. Roisenberg, J.M. Barreto, F.M. De Azevedo. Neural network complexity classification based on the problem. *Proc. IJCNN - IEEE International Joint Conference on Neural Networks*, vol.3, pp. 2413-2418, 1998.

Efficient Tiny Hardware Cipher under Verilog

Issam Damaj
Dept. of Electrical and Computer Engineering
Dhofar University
P.O. Box 2509, Salalah 211, Oman
i_damaj@du.edu.om

Samer Hamade, and Hassan Diab
Dept. of Electrical and Computer Engineering
American University of Beirut
P.O. Box 11-0236, Beirut, Lebanon
smh22@aub.edu.lb, diab@aub.edu.lb

KEYWORDS
Gate Arrays, Cryptography, Algorithms, Hardware Design

ABSTRACT

Embedded hardware security has been an increasingly important need for many modern general and specific purposes electronic systems. Minute security algorithms with their expected low-cost and high-speed corresponding hardware realizations are of particular interest to fields such as mobile telecommunications, handheld computing devices, etc. In this paper, we analyze and evaluate the development of a cheap and relatively fast hardware implementation of the extended tiny encryption algorithm (*XTEA*). The development will start by modeling the system using finite state machines (*FSMs*) and will use *Verilog* hardware description language to describe the design. Minimizing the chip area will be our primary target rather than the construction of a multi-way massively parallel implementation with its expected high-speed and large silicon area. Many hardware design tools are used to try reaching the best possible optimized syntheses. The targeted hardware systems are the reconfigurable *Altera's Stratix II* and *Xilinx Virtex II Pro* modern field programmable gate arrays (*FPGAs*).

INTRODUCTION

Security of information has become a main issue in the ever evolving world of small mobile devices such as personal digital assistants (*PDAs*) and cell phones. In such minute devices, the fight over high performance and low power consumption, besides security, are primary targets. However, a great deal of assistance in creating low-power and high-speed cores, comes from the simplicity of the selected algorithm for embedding as a hardware component.

Many encryption algorithms are now available in the market (Kelsey et al. 1996), and the selection of a specific one is dependent on the relatively tight constraints in small devices. The selected algorithm should be small, relatively secure, with a proven history of overcoming possible well known attacks on it. The Tiny Encryption Algorithm (*TEA*) (Wheeler and Needham 1994), and hence its successor the Extended-TEAs (*XTEAs*) (Needham and Wheeler 1997; Russell 2004; Kelsey et al. 1997; Moon et al. 2002) are among the best choices available for the above taut requirements and to be implemented in the research in hand.

Other requirements are still of no less important than the issues of performance and power consumption; these include the ease of modifiability, upgradeability and reuse of the designed security components. The type of hardware circuits to be used for implementing the developed cores, largely affects the above modifiability properties. Here we propose reconfigurable computers; more specifically field programmable gate arrays (*FPGAs*), as a possible solution with their famous property of programmability to satisfy the addressed need for modifiability.

FPGAs, nowadays are important components of reconfigurable systems; they have shown a dramatic increase in their density over the last few years. For example, companies like *Xilinx* and *Altera* have enabled the production of *FPGAs* with several millions of gates, such as in *Virtex-II Pro* and *Stratix-II FPGAs*. The versatility of *FPGAs*, opened up completely new avenues in high-performance computing. These programmable hardware circuits are aided with various co-design tools and flexible design methodologies to form a powerful paradigm for computing.

The traditional implementation of a function on an *FPGA* is done using logic synthesis based on *VHDL*, *Verilog* or a similar *HDL* (hardware description language). These discrete event simulation languages are rather different from languages, such as *C*, *C++* or *JAVA*. Many *FPGA* implementation tools are primarily *HDL*-based and not well integrated with high-level software tools. Furthermore, these *HDL*-based *IP* (intellectual property) cores are expensive and they have complex licensing schemes. In the presented designs, the hardware implementations are carried out under *Verilog*, employing different co-design tools. The targeted *FPGA* systems are *Altera Startix II* and *Xilinx Vertix II Pro*. The hardware design tools involved in this project are *Altera's Quartus*, *Xilinx ISE*, *Mentor Graphics HDL Designer*, *Leonardo Spectrum*, *Precision Synthesis* , and *ModelSim*.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

THE TINY ENCRYPTION ALGORITHM

In cryptography, the Tiny Encryption Algorithm (*TEA*) is a block cipher notable for its simplicity of description and implementation (typically a few lines of code). The cipher was initially presented by (Wheeler and Needham 1994). *TEA* operates on 64-bit blocks and uses a 128-bit key. It has a Feistel structure with a suggested 64 rounds, typically implemented in pairs termed cycles. It has an extremely simple key schedule, mixing all of the key material in exactly the same way for each cycle.

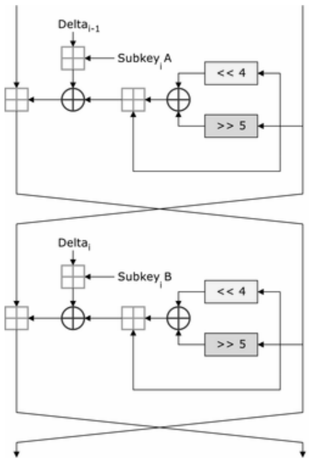


Figure 1. A single XTEA round with its internal computational constructs. The crossed square for the sum, crossed circle for an XOR, >> for a right shift, << for a left shift.

XTEA is a symmetric block cipher designed to correct weaknesses in *TEA*. Like *TEA*, *XTEA* is a 64-bit block Feistel network with a 128-bit key and a suggested 64 rounds. Several differences from *TEA* are apparent, including a somewhat more complex key-schedule and a rearrangement of the shifts, XORs and additions (Hong et al. 2003; Ko et al. 2004). Figure 1 show the block diagram of an XTEA single round.

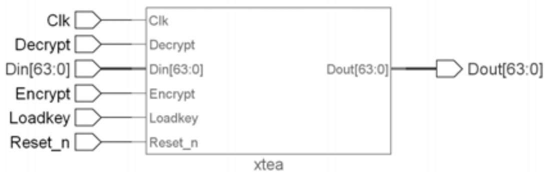


Figure 2. Block Diagram generated from the top model by Leonardo spectrum

High-speed hardware implementations of the *XTEA* under *VHDL* were suggested in (Ghazzawi et al. 2006). The power consumption of the *XTEA* were studied in (Kelsey et al. 1996) and compared with results for the *RC5* algorithm.

XTEA HARDWARE

In Figure 2, the block diagram of the created chip is shown. The 128-bit key is input through the *Din* pins entered 64 bits at a time. The same *Din* pins are used to enter the plaintext (or ciphertext in the case of decryption) 64-bits at a time, while *Dout* pins are used for outputting the ciphered (or plaintext in the case of decryption). The remaining pins are for clocking (*Clk*) and control signals, moreover, loading the key (*Loadkey*), enabling encryption or decryption (*Encrypt* or *Decrypt*), and resetting the system (*Reset_n*).

The development of the *XTEA* core is started by creating a finite state machine (*FSM*) with four possible states. The system will be initially in its *IDLE* state till the control signals are received. The transition that takes you from the *IDLE* state to *BUSY_KEY* state is controlled by the external event *Loadkey*. The state *BUSY_KEY* is responsible for inputting the key. After finishing the key inputting process the system returns automatically to its *IDLE* state. The system will undergo a transition to its *BUSY_ENC* (encryption state) or *BUSY_DEC* (decryption state) according to the transitions controlled by the events expected on the *Encrypt* and *Decrypt* pins. The system continuous operation is done by returning to the IDLE state on finishing the encryption or the decryption is shown in Figure 3.

A single *XTEA* round is to be repeated 32 times. A sequential version on the round level would mean the creation of circuit corresponding for a single round, then the output is fed-back to the circuit to become the input of the following round; this is to happen 32 times. A different degree of parallelism could be reached if the designer decides to unroll the loop to construct a fully-pipelined network of rounds. In this paper, we show a sequential version avoiding any resources replication and accordingly any additional expected increased area, cost, and power consumption. In Figure 4, a circuit block diagram was obtained using *HDL Designer* from *Mentor Graphics*. In Figure 4, the feedback wires are clearly shown going from the *eb1* block to the *eb2* main block. The *eb2* block contains all the inputs and outputs, and synchronized by a master clock. The second block *eb1* contains continuous assignment statements for the main functions (or core) of the encryption (decryption) algorithm. The output generated by *eb1* is fed-back to *eb2* synchronized by the master clock.

As shown in Figure 1, an *XTEA* round implementation requires the construction of the following computational elements:

- Addition and Subtraction modulo 32.
- Bitwise XOR
- Shift left and right operations

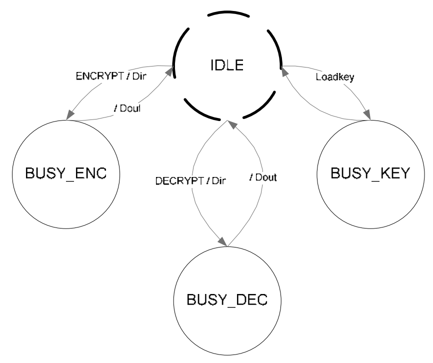


Figure 3. Finite State Machine

PERFORMANCE EVALUATION

For the purpose of analysis we present first the results of testing the design using *Altera's Quartus* tool, where we build the simulation cases graphically. From the performed simulation, the number of clock cycles needed to complete a single encryption or decryption process is 68 cycles (key loading 2 cycles, encryption/decryption 32 * 2 cycles, and wait states 2 cycles). A pipelined version will, with no doubt, enhance the performance by decreasing the process total time, but as a quid pro quo for silicon area.

In Figure 5, we show the waveforms of testing the encryption process. The *reset* signal was activated at first to insure that all the registers are cleared before starting any operation, note that the asynchronous reset is active low once. After that the *Loadkey* control was activated for two clock cycles to store the selected key that will be used in the next process.

Now, the module is ready to either encrypt or decrypt. One can distinguish between these two by the control signals provided as an input to the system. After engaging the *Encrypt* signal, the system will enter the *BUSY_ENC* state, and will finish the encryption after 64 clock cycles. You can notice that the output is available at that time and that the system is returned to the *IDLE* state waiting to the next control signal to operate.

In Figure 6, we depict the waveforms of decryption state transition testing. The encryption and the decryption processes are quite the same in architecture, but the decryption operates in a reverse manner on the data, and uses a subtractor rather than an adder. This test shows the transition of the state from the *IDLE* to the *BUSY_DEC*, the output will be available after 64 clock cycles.

In Figure 7, we show the simulation for testing the key loading process. In order to reduce the number of IOs used, the key was distributed over the *Din* input pins. Knowing the fact that the key length is 128 bits while the Din is only 64 bits, we need two clock cycles to load the key to its internal register inside the *FPGA*. This will cost using the *Loadkey* control driving the system to the *BUSY_KEY* state. The *BUSY_KEY* state will need two clock cycles to exit and return to the *IDLE* state again.

The number of IOs needed in this project is fixed and can easily be calculated from the module's event list, this number is equal to 133 IOs. The next step of assessment is to map the developed design onto different *FPGA* systems comparing the use of resources in each case. The synthesis tools used in this comparison are *Xilinx XST*, *Mentor Graphics Precision Synthesis RTL and physical 2004*, *Altera's Quartus*, and *Leonardo Spectrum 2004*. Both, *Precision Synthesis* and *Leonardo Spectrum* are vendor free third party tools developed to synthesize popular *FPGAs*. In Table 1, we show the different findings of compiling the developed *XTEA* design to *Altera's Stratix II FPGAs* with three different sizes. In Table 2, we show the findings after compiling the design to *Xilinx Virtix II Pro FPGA*. The chart in Figure 8 shows clearly the maximum speed of 134 Mbps achieved by mapping our sequential small-sized design onto the Virtix II Pro *FPGA*.

It is clear from the results shown in this section that the aim of obtaining a relatively small area has been achieved. The advantage of having a small design with a small occupied area had its impact on speed, where a maximum speed of 134 Mbps was achieved with the suggested sequential design. A similar design achieved only a speed of 16.8 Mbps in (Ghazzawi et al. 2006), but was enhanced by eliminating the large area occupied by the controller part and accordingly reducing the propagation delay and the number of clock cycles. The enhanced design in (Ghazzawi et al. 2006) reached a speed of 800 Mbps; more manual designs where offered expecting speeds above 1 Gbps.

The tiny *XTEA* for sure is not comparable to any of the powerful ciphers like the *AES* finalists, but it had the following summarized advantages:

- Small expected hardware silicon area.
- It is relatively secure enough with a number of rounds above 16.
- Fast enough to accompany other projects.
- Low power consumption.

Based on the comparison done between various synthesis tools, the following is concluded:

- For a small application like the development of the *XTEA*, no big improvement is gained by selecting one synthesis tool over the others.
- Choosing different FPGA device from the same family will not affect largely the amount of occupied area.

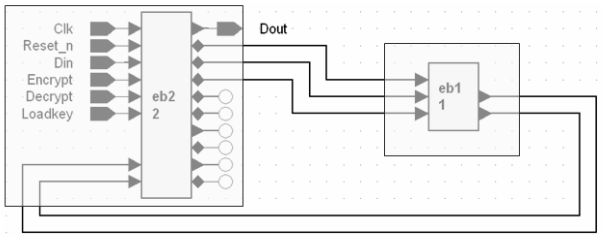


Figure 4. Block Diagram of the Implemented Sequential System

Increasing the depth of investigation concerning accelerating the *XTEA* would lead us, however, to parallel processing including pipelining. Thus, generic reasoning about the parallelization of the algorithm in hand is a possible extension for the proposed work, besides investigating the correctness of various parallel hardware implementations. Developing correct hardware leads to the adoption of a formalization framework. Through this formal mathematical framework, different parallel designs could be generated systematically, using provably correct rules of refinement. An example of such a framework is the *Bird-Meertens* Formalism (BMF) by which one generates data parallel programs from abstract specifications using the skeleton approach. The essence of this approach is to design a generic solution once, and to use instances of the design many times for various parallel architectures. Another frame work starts by formulating an algorithm by a generic formal functional specification step and generates parallel programs described in a concurrency framework - *CSP* (Communicating Sequential Processes). Through such developments, our implementations will benefit from the advances in the area of hardware/software co-design to generate efficient hardware *XTEA* circuits.

The generation of an efficient hardware solution for the *XTEA* would with no doubt satisfy the need for speed and efficiency. The parallelized designs could easily be mapped to various parallel architectures such as clusters, grids, *FPGAs*, complex programmable logic devices (*CPLDs*), dynamically reconfigurable systems (the *MorphoSys* (Bagherzadeh et al. 1999), etc. The availability of such systems with different sizes and speeds obliges us to study the parallelization of our algorithm not only in its sequential or pleasantly data-parallel version, but also with different degrees of parallelism. This will include reasoning about the use of pipelined blocks, partially sequential blocks, etc. Again, the parallelization is to be n a systematic parallelization framework which is done in a straightforward manner.

CONCLUSION

The research presented in this paper is motivated by the need for low-power, fast, tiny, and cheap hardware security cores. We have presented a sequential, small-sized, relatively fast implementation of the extended tiny encryption algorithm (XTEA). The best achieved synthesis was by mapping the design onto a Virtex Pro II FPGA with a tiny area and a speed of 134 Mbps. The mapped design employed 32 rounds, although 16 rounds are assumed to be secure enough. Many extensions of the work in hand are present. The extensions could include the formal development, for the sake of correctness, of the XTEA and its successors the XXTEA and block XTEA. Multi-way massively parallel implementations are expected to increase the throughput at the expense of silicon area.

REFERENCES

Bagherzadeh N. Kurdahi F. Singh H. Lu G. Lee M. and Filho E.1999. "MorphoSys: An integrated reconfigurable system for data-parallel computation-intensive applications." IEEE Transactions on Computers.

Ghazzawi W., R. Saraeb, and I. Damaj. 2006. "Hardware Development of the Extended Tiny Encryption Algorithm," in the ACS/IEEE International Conference on Computer Systems and Applications, Dubai/ Sharjah, United Arab Emirates (Mar).

Hong S., Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, and Sangjin Lee. 2003, "Differential cryptanalysis of TEA and XTEA." In *Proceedings of ICISC 2003*, 2003b.

Ko Y., Seokhie Hong, and Wonil Lee. 2004. "Related key differential attacks on 26 rounds of XTEA and full rounds of GOST."In *Proceedings of FSE '04*, Lecture Notes in Computer Science, Springer-Verlag,

Kelsey J., Bruce Schneier, and David Wagner.1997. "Related-key cryptanalysis of-WAY, Biham-DES, CAST, DES-X NewDES, RC2, and TEA." *Lecture Notes in Computer Science*, 1334: 233-246.

Kelsey J., Bruce Schneier, and David Wagner. 1996. "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES." Lecture Notes in Computer Science, 1109: 237-251.

Moon D., Kyungdeok Hwang, Wonil Lee, Sangjin Lee, and Jongin Lim. 2002. "Impossible differential cryptanalysis of reduced round XTEA and TEA."*Lecture Notes in Computer Science*, 2365: 49-60.

Needham R. M. and David J. Wheeler.1997. "Tea extensions." Technical report, Computer Laboratory, University of Cambridge(Oct).

Russell M. D. 2004. "Tinyness: An Overview of TEA and Related Ciphers." http://www.users.cs.york.ac.uk/~matthew/TEA/TEA.html

Wheeler D. J. and R. M. Needham.1994. "TEA, a tiny encryption algorithm." Fast Software Encryption, Leuven, LNCS 1008, , pp. 363-366.

Issam W. Damaj received his Bachelor of Engineering (B.Eng.) in Computer Engineering from Beirut Arab University in 1999 (with high distinction), and his Master of Engineering (M.Eng.) in Computer and Communications Engineering from the American University of Beirut in 2001. He was awarded his Ph.D. degree in Computer Science from London South Bank University, London, United Kingdom in 2004. Currently, he is an Assistant Professor of Electrical and Computer Engineering and the chairperson of the Department of Electrical and Computer Engineering, Dhofar University, Oman. His research interests include hardware/software co-design, embedded systems design, reconfigurable computing, parallel processing, and software engineering.

Samer Hamade is a graduate student at the American University of Beirut. He is a student in the Faculty of Engineering and Architecture, Department of Electrical and Computer Engineering. His major is Computer and Communications Engineering.

Hassan B. Diab received his B.Sc. (with Honors) in Communications Engineering from Leeds Metropolitan University, U.K. in 1981, his M.Sc. (with Distinction) in Systems Engineering from the University of Surrey, U.K. in 1982, and his Ph.D. in Computer Engineering from the University of Bath, U.K. in 1985. Dr. Diab is a Professor of Electrical and Computer Engineering at the Faculty of Engineering and Architecture, American University of Beirut (AUB), Lebanon and has over 22 years of experience. He has 116 publications in internationally refereed journals and conferences. His research interests include cryptography on high performance computer systems, modeling and simulation of parallel processing systems, embedded systems, reconfigurable computing, simulation of parallel applications, system simulation using fuzzy logic control, and the application of simulation for engineering education.

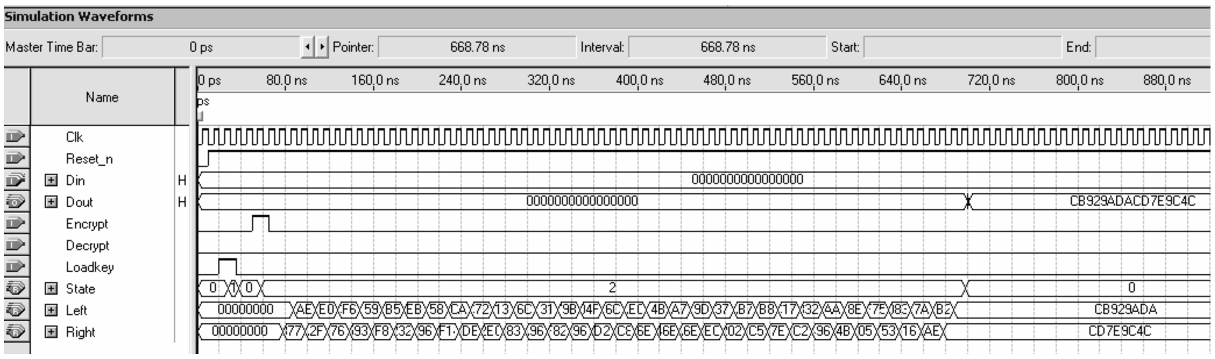


Figure 5. Test Case: Encryption Testing, Input Key: 0x0, Plaintext: 0x0, Ciphertext: 0XCB929ADACD7E9C4C

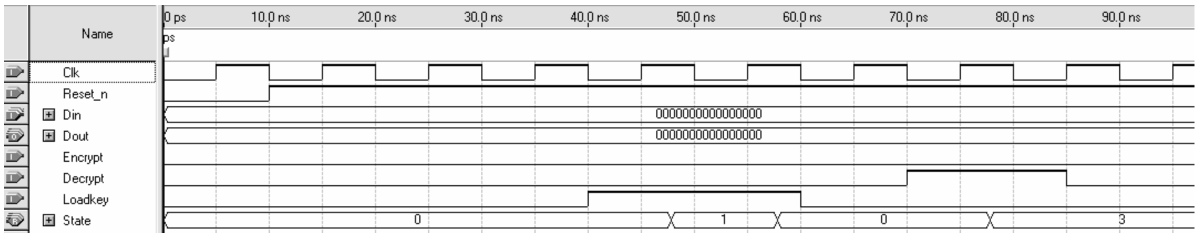


Figure 6. Test Case: Decryption State Transition

Table 2. Results of Mapping the Developed Design to *Xilinx Vertix II Pro FPGA*

Devices	**Leonardo Spectrum**	**Xilinx XST**	**Precision Synthesis**
Device Number:	**2VP2fg256**		
Number of IO:	133 out of 140 (utilization = 94.3%)		
CLBs Slices (408 available)	294	393	539
Maximum Frequency in MHz	91.2	120.3	142.4
Speed in Mbps	85.83	113.22	134
LUTs (2816 available)	588	634	624
Slice Flip Flops (2816 available)	298	307	305

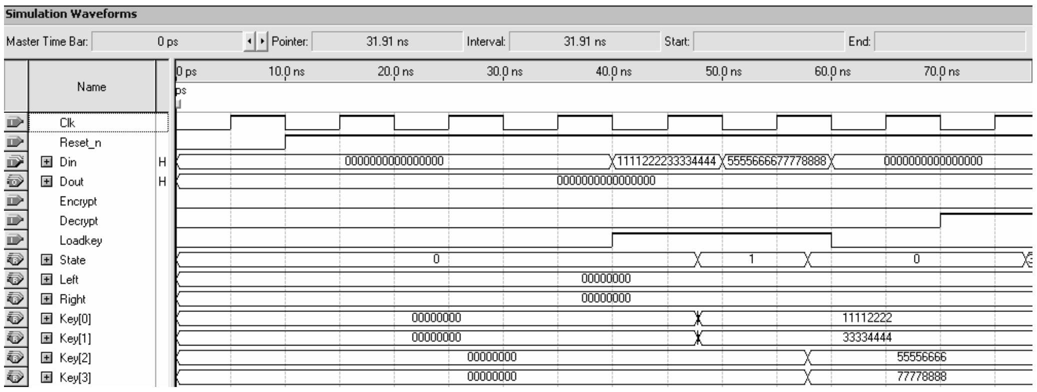


Figure 7. Test Case: Key Loading.

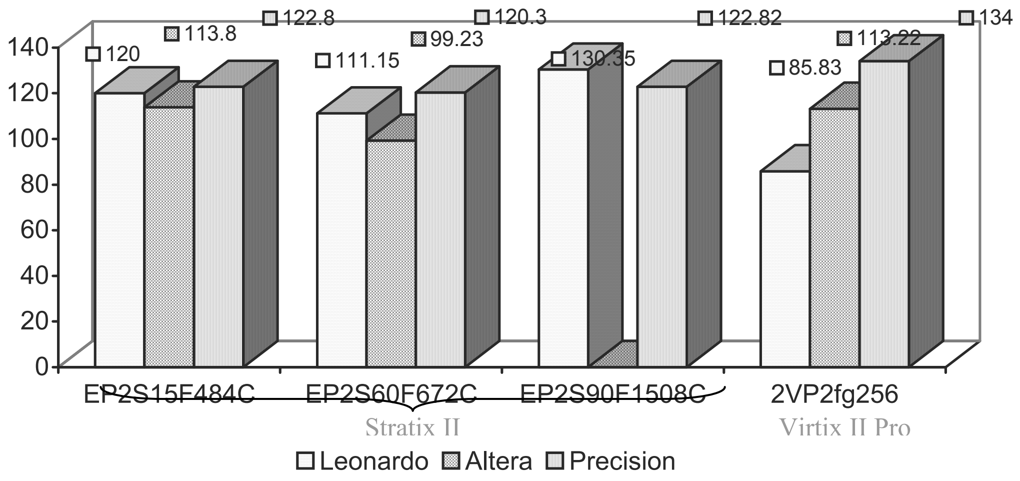


Figure 8. Speeds in Mbps of the Developed Implementations.

Table 1. Results of Mapping the Developed Design to *Altera's Stratix II FPGAs* with Three Different *FPGA* Sizes.

Devices	**Leonardo Spectrum**	**Altera Quartus**	**Precision Synthesis**
Device Number:	**EP2S15F484C**		
Number of IO:	133 out of 343 (utilization = 38.78%)		
LUTs used (12480 available)	526	573	539
Maximum Frequency in MHz	127.5	120.95	130.5
Speed in Mbps	120	113.83	122.82
Registers used (14410 available)	297	297	305
Device Number:	**EP2S60F672C**		
Number of IO:	133 out of 493 (utilization = 26.98%)		
ALUTs used (48352 available)	526	573	539
Maximum Frequency in MHz	118.1	99.23	120.3
Speed in Mbps	111.15	93.39	113.22
Registers used (51182 available)	297	297	305
Device Number:	**EP2S90F1508C**		
Number of IO:	133 out of 903 (utilization = 14.73%)		
ALUTs used (48352 available)	526	NA	539
Maximum Frequency in MHz	138.5 MHz	NA	130.5 MHz
Speed in Mbps	130.35	NA	122.82
Registers used (51182 available)	297	NA	305

Special Session on Web Services and the Semantic Grid (WSSG'08)

ACCESS CONTROL OF WEB SERVICES USING GENETIC ALGORITHMS

Nabila SEMMACHE and Sadika SELKA
Department of Computer Science
University of The Science And The Technology,
1505-El-Mnouar, Oran, ALGERIA
E-mail: Semmache_hanane@yahoo.fr

KEYWORDS
Web service, SOAP, UDDI, WSDL, WS-Security, Genetic Algorithm.

ABSTRACT

Although Web services have been simplified a lot in terms of development, the problem of the security of these services is crucial and remains still confusing and complex. This last one lies around three axes: the Identification and the authenticity of a user, Protection of the confidential data and the authorization of access to applications of the web services. In this article, among the three axes we are interested in the authorization of access of the users to the applications of the Web services. For that purpose, we propose an approach based on genetic algorithms, so that the tasks of the Web services are secured.

INTRODUCTION

Web service can be defined as a mechanism of communication between distant applications through the Internet network, independent of any programming language and any platform of execution. It is based on standard Web protocols such as XML (*eXtensible Markup Language*), for the coding of the parameters and the values of return, SOAP (*Simple Object Access Protocol*) (Gudgin et al. 2007), for the transport of messages, WSDL(*Web Description Language services*) (Chinnici et al. 2007) for the description and UDDI (*Universal Description, Discovery and Integration*) (Clément et al. 2005) for the publication.

Although these protocols allow today to build applications and to put them in production, numerous evolutions remain to be brought to offer the consideration of quality criteria of services such as protected from delivery, the transactions and the security.

These last years, the research in the field of Web services was very active. A big part of this one was dedicated to the security.

One of the solutions proposed to secure Web services was to assure the reliability of the connection between the customer and the server. With a transport secured as SSL(*Secure Sockets Layer*), (Freier and Karlton 1996) the services do not have to manage themselves the integrity and the confidentiality of every message; they need to relay on the mechanisms of the layer transport underlying. However, they turn out that a security at the level transport (Merrells 2004) is a limited solution to exchanges of messages from a point to the other one.

For this security at the level message (Merrells 2004) was proposed to maximize the reach of the Web services. The standards of security used at this level are: WS-Security (Atkinson et al. 2002), WS-Trust (Della-Libera et al. 2002), WS-Privacy (Nagaratnam et al. 2003), WS-Policy (Curbera et al. 2003), WS-Federation (Bajaj et al. 2003), WS-SecureConversation (Dixon et al. 2002) and WS-Authorization (Della-Libera et al. 2002).

In this article, we propose another security at the level of the tasks of the Web services by using the genetic algorithms, with the aim of controlling the access of the users.

ARCHITECTURE OF WEB SERVICES

The efforts of research and development about Web services led to a number specifications which define the architecture of the Web services (McCabe et al. 2004).

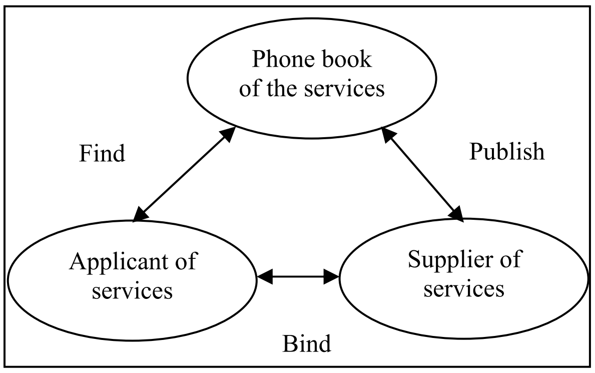


Figure 1 : Architecture of Web services

The architecture of the Web services articulates around the following three roles:

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

- Supplier of service: corresponds to the owner of the service. From a technical point of view, it is constituted by the platform of reception of the service;

- Applicant of service: corresponds to the applicant of service. From a technical point of view, it is constituted by the application which is going to look for and to call upon a service;

- The phone book of the services: corresponds to pad of descriptions of services offering opportunities of publication of services for the suppliers as well as opportunities of research for services for the customers.

ARCHITECTURE OF SECURITY OF WEB SERVICES

For to secure Web services, several mechanisms must be addresses so as to guarantee the security of the exchanges of information:

- The authenticity: allows to verify that the access to an application or to a distant resource is made only by entities having proved their identity;

- The integrity: allows to verify that the message received by the addressee was not modified during its transmission;

- The confidentiality: allows to verify that the transmitted messages cannot be read by entities others than the receiver and the transmitter of the message;

- The non-repudiation: guarantees to the addressee of a message that the transmitter of that this is well the one that it claims to be;

- Authorization of access: put at the disposal of mechanisms allowing to authorize or not the access to such or such resource.

The standards of security used at the level message and that allow to assure these various mechanisms, are illustrated in the figure 2;

WS-Secure Conversation	WS-Federation	WS-Authorization
WS-Policy	WS-Trust	WS-Privacy
WS-Security		
Fondation SOAP		

Figure 2 : Architecture of security of web services

- WS-security: Specification of secured messaging using SOAP messages. It processes the following three aspects:

1. Specification of mechanisms allowing to assure the integrity of messages SOAP. It deals more particularly as the joint use of the specification XML signature (Bartel et al. 2002) and as the tokens of security.

2. Specification of mechanisms allowing to assure the confidentiality of messages SOAP. It deals more particularly as the joint use of the specification XML encryption (Takeshi et al. 2002) and as the tokens of security.

3. Definition of a mechanism to associate tokens of security with the headings of messages SOAP. Without recommending of specific format, it specifies a method to create new formats of tokens of security as well as mechanism of encoding of binary tokens.

- WS-Policy: allows describing in terms of security the requirements of the addressee as well as the capacities of security of the transmitter of a message.

- WS-Trust: describes a model allowing establishing reliable relations which can base itself on a system of tokens of security.

- WS-Privacy: describes the policies of discretion and confidentiality.

- WS-Secure Conversation: describes how two entities can communicate by authenticating mutually and by forming a context of security.

- WS-Federation: allows building global reliable spaces so allowing making authenticity between entities using different methods of authenticity.

- ➢ WS-Authorization: describes how to manage and to create rules of authorization, to certify authorizations via tokens, to exchange these tokens and to interpret these so as to control correctly the accesses to the services.

SECURITY OF WEB SERVICES USING GENETIC ALGORITHMS

In this article, we are interested in the security of the Web services at the task level to control the access of the users by using genetic algorithms (Rennard 2002). The approach which we propose consists of two phases: phase of discovery and phase of optimization which is really the genetic process.

Phase Of Discovery

A supplier implements Web service, he defines his description in the form of a file WSDL and publishes it by saving it in a phone book UDDI. The example which follows will be studied throughout the article.

Let us consider the case of a travel agency which offers Web service of grouped booking combining a plane ticket, a booking of hotel room and a rent of car among others. For that purpose, the agency calls Web services of an airline company, a hotel chain and a renter out of automobiles.

Here are the tasks executed by every Web service:

- The task executed by Web service of an airline company is: the booking of ticket;

- The task executed by Web service of a chain of hotel is: the booking of room;

- The task executed by Web service of a renter out of automobiles is: the rent of car.

The Parameters of every task, described in WSDL are:

- The parameters of the task booking of ticket: family name, first name, date of departure, date of exit, Country of destination;

- The parameters of the task booking of room: family name, first name, dates of the beginning of stay, Dates of the end of stay, type of room to be reserved;

- The parameters of the task rent of car: family name, first name, dates of the beginning of rent, Dates of the end of rent, Type of car to be rented.

All these parameters will be added to the other parameters which are not described in the file WSDL. For example for the spot of booking of tickets the parameters which will be posted (shown) in WSDL are: family name, first name, and date of departure, date of exit and country of destination. Those which are non-visible are: the number of passport and number of the ID card. After the discovery of the Web services, the phase of optimization follows.

Phase Of Optimization

This phase is the release of the genetic process by the supplier during the invocation of Web service by the customer for the access control of this last one. This genetic process passes by:

- A coding of individuals

- The generation of initial population

All the specific parameters in every task of Web service, supplied by the supplier, are set of individuals which represent the potential solutions to control the access of the users.

Coding Of Individuals

The coding which we propose for our approach is illustrated in table below follows;

Table 1: Table Recapitulating the Correspondences between Main Terminologies of the Genetic Algorithms and Web Services

Natural notion	Definition	Coding of Individuals
Chromosome	One or several chromosomes form together the global genetic plan for the construction and the functioning of a body	The tasks of the Web services
Genes	We say that chromosomes are constituted by genes	The parameters of the tasks of Web services
Alleles	A gene can set different alleles	The Value of every parameter
Locus	The position of a gene	The position of the parameter

The generation of the initial population is made according to two stages: evaluation of the parameters of the initial population and Application of the genetic operators.

Evaluation Of The Parameters Initials

To estimate a specific parameter for a task of Web service, we choose first of all parameters Pi with i € [1, n]. Every parameter will have a value Vij with j € [1, m], determining its degree of importance by contribution to the other parameters chosen. This value is calculated as follows:

$$Vij = Xi / \sum_{i=1}^{n} NbrPi \qquad (1)$$

Vij**:** value j of the parameter i;
Xi: The number of position of the parameter i;
NbrPi : The number of parameters Pi chosen as a task;

For every parameter, we attribute a weight of reliability Pfi with 0 < Pfi < 1 and i € [1, n], corresponding at the request of access of the users (0 corresponds to 0 % and 1 corresponds to 100 %). This means that this weight of reliability Pfi, changes according to the demand of access of the users.

In a general way, the function of adaptation measures the performance of an individual in the resolution of the problem posed. In the precise context of the problem of access controlling, the adaptation of every specific parameter to a task of Web services, is expressed in our approach as follow:

$$F(Pi) = Vij \sum_{i=1}^{n} Pfi \qquad (2)$$

According to the previous example of the travel agency, table2 illustrates the values of the function of adaptation of the parameters of the task “booking of a ticket".

Table 2: Adaptation values of parameters of The ticket booking task

Parameters	Individuals	Values of F(Pi)
Family name	I_1	0.18
First name	I_2	0.20
Country of destination	I_3	0.33
Date of departure	I_4	0.42
Date of exit	I_5	0.56
Number of the ID card	I_6	0.75
Number of passport	I_7	0.80

Genetic Operators

Selection

After calculation of the value of adaptation of every specific parameter to a task of Web services, the operation of selection which is going to allow us to select the potential parameters for access controlling intervenes.

For our approach, the method of chosen selection is elitism. This method of selection allows selecting the best individuals of the population. It is thus the most potential parameters which are going to participate in the improvement of our population.

In our frame of application, the method of selection is translated as follow: the value of the function of adaptation of every parameter will be compared with a called indication "Indication of reliability" Pa, to attribute to every task of Web services.

We fix Pa to a value equal to 0.5, corresponding to 50%. All the parameters which will have a value of adaptation superior or equal to this value (Pa=0.5), will be selected.

According to table 2, the selected parameters are:

$\{(I_5,0.56) ; (I_6,0.75) ; (I_7,0.80)\}$

When the potential parameters are selected, we proceed in their mutation.

Mutation

For a better generation, we used another genetic operator “The Mutation ". The method of mutation which we propose consists of a permutation of two parameters. We are certain that the mutated parameters will always have the shape of a potential solution because we change only the order of the parameters. This permutation is made by basing itself on the value of adaptation of both parameters in the new intermediate generation. What means that among the selected parameters, the one which has the best value of adaptation will be generated.

According to the selected parameters, the mutation is as follows:

{(I_6,0.75) ; (I_5,0.56) ; (I_7,0.80)}.

APPLICATION OF GENETIC PROCESS

According to the example of the travel agency, we make a case study for a hostile customer who would like to make a booking of a ticket.

The customer calls upon Web service of the booking of ticket by authenticating. There is a start of the genetic process. Two cases appear:

First Case (At Least A Parameter In Wsdl)

Among the generated potential parameters, there is at least a parameter, which is at the level of the WSDL elaborated by the supplier. For example for the task of booking of tickets, a generated potential parameter is the date of exit. Thanks to this parameter we control the user.

Here also there are two cases which appear:

The user reaches for the first time thus there are no other parameters which are displayed except those in WSDL. The user is going to make his booking.

The user is authentic, but he wants to modify his date of departure. Since this parameter is potential, another potential parameter, which is not in WSDL, will be displayed, for example, the number of passport.

- If the number of passport is valid: the modification is accepted and the access is authorized.

Let us consider a hostile user who wants to reserve the ticket using the name of somebody else who has already reserved.
For example if he wants to modify the date of departure and it is a potential parameter, then there is a display of another potential parameter which is not in WSDL. For example number of passport. Two cases exist:

- If the number of passport is not valid: the modification is rejected, and so the access is interrupted.
- If the number of passport is valid: the modification is accepted, and the access is authorized.

If in this last case, the customer is suspicious we can control this thanks to another Web service. For example, when he wants to pay the expenses of the ticket (with the banking service), potential parameters appropriate for this service, will be displayed. For example if the entered number of bank card is not valid and if it is a potential parameter, then, there will be a display of another parameter which is the digital signature.

The second case (no potential parameters in WSDL): The control is made by the others Web services.

The last case: if the customer enters with another login, and he does not make his reservation using the name of another person we consider him as an authentic user until the first mistake which he will make.

CONCLUSION

In this article, we proposed an approach of access control of the users in Web services, based on the genetic algorithms. While trying to generate potential parameters among those supplied by the supplier, so that we can demonstrate that from these last ones, the tasks of Web service, can themselves control the access: compound Web services and that of the users.

The purpose of WS-Authorization is to describe how access policies for a Web service are specified and managed. In particular, the goal is to describe how claims may be specified within security tokens and how these claims will be interpreted at the end point.

Thus with WS-authorization, the access control in Web service is made by verifying constraints attributed to the specific parameters of Web services tasks.

With regard to our approach, the access control is made without attribution of constraints for the parameters, the

customer just has to touch a potential parameter generated by the genetic process and described in the file WSDL so that the control begins.
The second advantage is that in WS-authorization, if the supplier adds a parameter, he has to specify ways of access to this new parameter. For our approach, if the supplier adds parameters, it is enough to activate the genetic process to generate new potential parameters and the security starts.

As a perspective of this work, it would be interesting to take into account the profiles of the users.

REFERENCES

Atkinson .B, S. Hada and P. Hallam-Baker. April 2002. “Web Services Security (WS-Security)Version1.0”.

Bajaj .S, B. Dixon and C. Kaler. July 2003. “Web Services Federation Language” .

Bartel .M, J. Boyer, B. Fox and B. LaMacchia. 2002. “Signature Syntax and Processing”.

Chinnici .R, J. Moreau, A. Ryman and S. Weerawarana. June 2007. “Web Services Description Language (WSDL) Version 2.0”.

Clément .L, A. Hately, C. Riegen and T. Rogers, february 2005. “Universal Description Discovery and Integration (UDDI) Version 3.0”.

Curbera .F, D. Langworthy and M. Nottingham. May 2003. “Web Services Policy Framework” .

Della-Libera .G, M. Hondo and P. Hemma. December 2002. “Web Services Trust Language”.

Dixon .B, H. Maruyama and R. Zolfonoon. December 2002. “Secure Web Services Conversation Language”.

Freier .O and P. Karlton. March 1996. “The SSL Protocol Version 3.0”.

Gudgin .M, M. Hadley, N. Mendelsohn, J. Moreau, HF. Nielsen, A. Karmarkar and Y. Lafon. April 2007. “Simple Object Access Protocol (SOAP) Version 1.2”.

Merrells .J. November 2004. “Web Services Security : Access Control”.

McCabe .F, E. Newcomer and M. Champion. February 2004. “Web Services Architecture”.
Nagaratnam .N, M. Hondo and A. Nadalin. June 2003. “Securing Web Services”.

Takeshi .I, D. Blair and E. Simon. 2002. “Encryption Syntax and Processing”.

ENABLING COLLABORATION IN THE SEMANTIC GRID: SURVEY OF WEB SERVICE COMPOSITION APPROACHES

Taha Osman, Dhavalkumar Thakker, David Al-Dabass
School of Computing & Informatics,
Nottingham Trent University,
CIB, Nottingham NG11 8NS, United Kingdom.
Email: taha.osman@ntu.ac.uk, dhavalkumar.thakker@students.ntu.ac.uk, david.al-dabass@ntu.ac.uk

KEY WORDS

Web Services, Service Composition, Semantic Web, Business Process Management.

ABSTRACT

Web Service is loosely coupled highly accessible distributed computing technology that can expose applications beyond the firewall. Composition of Web Services has received much attention from the business and the research community. Composition techniques are classified as static, dynamic and semi-automatic composition, each addressing different application areas and requirements. In this contribution we analyze workflow-based and semantic-based composition approaches, primarily focusing on the facilitation to the service participants and the scalability required in a Grid-based environment.

1. INTRODUCTION

The last decade has witnessed an explosion of application services delivered electronically, ranging from e-commerce to information service delivered through the World Wide Web (WWW) to the services that facilitate trading between business partners, better known as Business-to-Business (B2B) relationships. Traditionally these services are facilitated by distributed technologies such as RPC, CORBA and more recently RMI. Web Services is the latest distributed computing technology. It is a form of remote procedure call like other distributed computing technology, but uses XML extensively for the messaging, discovery and description. The use of XML messaging makes Web Services platform and language neutral. Web Services use SOAP (Simple Object Access Protocol) for XML messaging, which in turn uses ubiquitous HTTP for the transport mechanism. HTTP is considered as a secure protocol thus it allows the Web Services to be exposed beyond the firewall. The Web Service messages and operations with invocation details are described using a platform-independent language WSDL (Web Services Description Language). Web Services can be published and discovered using UDDI (Universal Description Discovery and Integration) protocol. The Web Services architecture centred on WSDL, UDDI and SOAP is an instance of Service Oriented Architecture (SOA). Using this architecture services can be published using UDDI, with WSDL based description, and can be searched, called and bind at run time making it loosely coupled and highly accessible.

To take advantage of these features of Web Services, network applications services have to be developed as Web Services or converted into Web Service using the wrapping mechanism (Osman T et al, 2005). Moreover, multiple Web Services can be integrated either to provide a new, value-added service to the end-user or to facilitate co-operation between various business partners. This integration of Web Services is called "Web Services composition" and is feasible to achieve because of the Web Services advantages of being platform, language neutral and loosely coupled. The composition is particularly apt for grid environments, where internet-wide computing resources are available for application services to interoperate and collaborate.

The logic for the composition mainly involves two activities: selection of the candidate Web Services that fulfil the requirement in accumulation and flow management. Flow management is further categorized into control and data flow, where control flow is the order in which Web Services operations are invoked, while the data flow is the order in which the messages are passed between the Web Services operations. The level of automation provided in performing selection of services and flow management classifies composition into static, semi-automatic and dynamic. Static composition involves prior hard coding of the service selection and flow management. Performing selection and flow management on the fly, in machine-readable format leads to dynamic composition. In semi-automatic composition, service composer is involved at some stage.

This study shows that these approaches can be divided into two categories. The first category largely endorsed by the industry, borrows from business processes' workflow management theory to achieve the formalization necessary for describing the data flow and control flow in the composition scheme. The second category mainly promoted by the research community, aspires to achieve dynamic composition by semantically describing the process model of Web service and thus making it comprehensible to reasoning engines or software agents.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

The structure of the paper is as follows: sections 2 and 3 discuss workflow-based and semantic-based composition techniques respectively. Section 4 provides evaluation of the surveyed composition techniques and in section 5 we conclude the paper.

2. WORKFLOW MANAGEMENT THEORY-BASED APPROACHES

Workflow is the movement of documents and/or tasks through a work process. More specifically, workflow is the operational aspect of a work procedure: how tasks are structured, who performs them, what their relative order is, how they are synchronized, how information flows to support the tasks and how tasks are being tracked (van der Aalst 2003).

Workflow management systems are a class of information systems that make it possible to correlate people's work and computer applications. Such systems deal with the control flow (invocation sequence of applications) and data flow (information flow between applications) while control flow is important for achieving overall system objective, data flow is essential for the successful operation of individual applications.

In the information systems domain, workflow is being used since seventies for the office automation systems (Zisman 1977). This work has lead to identifications of workflow patterns for control and data flow (van der Aalst 2003).

One of the applications of workflow management in information systems domain is to address the Business Process Management (BPM) problem. Business process can be considered as workflow of business activities to carry out business goals (Leymann 2002). The examples of business activities for customer order fulfilment business process are: customer placing an order, checking account status, verifying order and despatch. Using Workflow management, BPM deals with achieving the integration of these individual applications.

Business process can have scope within inter and intra organization relations. Enterprise Application Integration (EAI) is the BPM solution to achieve intra-organization business applications integration, while Business-to-Business (B2B) integration software addresses the problem for inter organization business application integration. Traditional EAI and B2B integration solutions are very complex, proprietary and presume many details about the participating applications making them tightly coupled. For instance, these solutions assume the use of homogeneous service interfaces and implementation technology, which is a substantial limitation considering that different organizations will make independent decisions about what technology to use for the construction and deployment of their part; these decisions made over time accrete different hardware and software technologies (High 2004). Tightly coupled systems are difficult to manage and re-engineering business rules and requirements in such systems is also challenging. To overcome these limitations, the business applications are now being developed using Web services while the BPM problems (EAI, B2B) are being addressed with the workflow based integration of Web services, mainly to utilize SOA based Web services features.

The main industrial standards to achieve workflow based integration of Web services are WS-BPEL (Web Services Business Process Execution Language, shortened to BPEL) (Andrews 2003), WS-CDL (Web Services Choreography Description Language, shortened to CDL) (Kavantzas 2004) and BPML (Business Process Modelling Language) (Van der Aalst 2003). These approaches use WSDL extensively and build workflow based on WSDL operations and messages with the data types. The workflow based process model for these approaches also addresses requirements for describing flow management in composition, handling business transaction with roll back facility, state management for business interaction support, and also handling exception and errors. The category of process model and the extent to which these features are provided differentiates these standards.

The following sections outline two prominent workflow based industrial standards for Web services composition.

2.1 Composing services using BPEL

The BPEL specification - enhances and replaces existing standards Web Services for Business Process Design (XLANG) (Thatte 2001) from Microsoft and Web services Flow Language (WSFL) (Leymann 2002) from IBM. The specification uses workflow management as process model to achieve the control and data flow formalization for WSDL defined data and operations. All the participant services in BPEL process are modelled as partners. The WSDL files of such partners are required to create BPEL process. The partners contribute to the total processing capability of the BPEL process. BPEL process also has its own processing capability for dataflow, control flow, data manipulation, fault and event handling and state management. The significance of BPEL architecture is that the process itself is published as a Web Service. This composed BPEL service can be treated as a single Web service and can be used for further composition hence facilitating recursive composition.

2.2 Composition using WS-CDL

BPEL process model deals with B2B integration from a single party viewpoint i.e., the requirement specified for the travel agent scenario discussed here is from the viewpoint of travel agent business logic. Contrary to the BPEL process model, real world B2B integrations are peer-to-peer in place of being centralized, where the collaborating business applications agree to provide certain functionality in receipt of complimentary functionality from other business applications highlighting requirement for a description language documenting peer-to-peer viewpoint since natural B2B integrations are peer-to-peer collaborative relationships and not governed by a single party. The W3C recommendation WS-CDL from W3C Web services choreography working group confirms aforementioned conclusions that more work on BPEL is required to make it adoptable for B2B integration.

WS-CDL is a description language using which the B2B integration partners can first describe the collaborative functionality. This description document is considered as a contract and each party can implement their own part. The WS-CDL document describes common and complementary behaviour of all the parties involved, making the viewpoint global and peer-to-peer (Kavantzas 2004). The other aspect of WS-CDL process model is that the internal business logic of each party remains hidden from the business partners. i.e., for the travel agent application after receiving price quote from all airlines can have internal business logic for air line selection based on some criteria totally hidden from other partners as the external detail described in WS-CDL document is just an operation to make reservation at particular airline.

2.3 Facilitation provided to the service participants

In order to evaluate the facilitation provided to the service participants we consider a scenario based on travel agent service, which manages the reservation of airline and hotel for a customer trip. The travel agent can be implemented as BPEL process, which cans becomposition of four Web services: AirFrance service, AirUSA service, HotelRating service and HotelService service. The process logic for the travel agent is: to check the availability of flight service from two competing airlines AirFrance and AirUSA, make flight reservation, and then retrieve hotel ratings from the HotelRating service at the destination city and make the reservation using HotelService Web service at the selected hotel.

For a new service provider to make their service available for composition they need to provide minimum functionality consistent with the business logic outlined by the travel agent which is essentially composer. Considering a new AirUK service for travel agent composition, AirUK has the following options:

a) If the AirUK has Web service but does not implement required functionality then the service needs to be modified to accommodate the required functionality.

b) If the AirUK has a non-Web service application with the required functionality already built-in then only a WSDL file is required to be created without modifying existing application. BPEL execution engine uses Web Services Invocation Framework (WSIF 2005) for the Invocation of such non web-services.

c) However, BPEL specification does not address a case where the Web service provider has a service available with conceptually similar but syntactically different parameter structure. The service provider in this case needs to apply option (a) to be part of the composition.

Considering the case of service composer who for the most part encounter problems in parameter mismatch during the flow management, i.e., a service operation has different output format from the input of next service operation in the flow logic, BPEL in its current form delegate the responsibly with the service composer to address such parameter mismatch.

The travel agent BPEL process could be published using JSP technology. This way the service can be retrieved using simple web page or WSDL file for the composed Web service can be retrieved from the public UDDI registry. In such B2C interactions it is totally transparent from the end-user that the service is a Web service with the possibility of composition of multiple Web services or could be implemented on heterogeneous platforms using heterogeneous programming languages. However, there is a limited level of language expressiveness available to the service requestor to outline the constraints and preferences on the outputs and quality of service parameters.

To conclude this section, BPEL is widely-used specification for composing intra-organization Web services. The business analysts and developers can collaborate and can compose enterprise Web services manually using BPEL. The composition is hard coded and the developers should have the explicit knowledge of all the details of participating business services which is a major limitation considering the growth of Web services within and outside organizations.

3. SEMANTIC WEB -BASED COMPOSITION

Commercial institutions are focusing their efforts on standardizing the static composition techniques in preparation for their wider adoption amongst the business community. In contrast, the research community efforts concentrate on exploiting semantic web for the semi-automatic and automatic composition of Web services.

3.1 Semantic Web services

With respect to automation, the limitation of workflow-based approaches is that they rely on WSDL based description for the Web services selection. WSDL is a static interface and XML grammar which has no notion of machine interpretable semantics. In Web services protocol stack, the task of meaningful Web services discovery was the functionality of UDDI implementations so that service provider can describe the capability of their service using the WSDL descriptions and service requester can use these descriptions to retrieve exactly what they are looking for. The search in UDDI is based on keywords and based on human readable descriptions in WSDL, leaving the selection based on the requestor's interpretation and ultimately the solution static.

The problem of automatic Web services discovery and integration can benefit from the semantic web machine readable descriptions. The fundamental premise of the semantic web is to extend Web's currently human-oriented interface to a format that is comprehensible to software programmes. Applied to Web services composition, this can lead to the automation of services selection and execution.

The WSDL file of Web services describe the operations provided; request message format required for invoking operations and the format of response messages produced by the Web services. The interpretation of these details results in understanding of the service capability. The automation required for the service composition can be achieved by describing the WSDL elements semantically, thus allowing software agents to reason about the service capability, and make all the decisions related to the composition on behalf of the user or developer. The decisions include the selection of appropriate services, their actual composition and close examination of how they meet the criteria specified by the user. In contrast, in the static composition approach, the user or developer manually interprets the requirements for the required composition and the available service capability or functionality and makes decisions regarding how services can be interweaved to make a value-added service.

The WSDL specification is part of the base Web services protocol stack and has been already widely accepted and implemented to describe Web services. Considering this, the general scenario will be to annotate individual WSDL elements with corresponding OWL elements. OWL-S (Dean 2005) is such ontology specification for describing Web services semantically. OWL-S ontology provides a mechanism to describe the capability of Web services in machine-readable form, which makes it possible to discover and integrate Web services automatically. OWL-S defines three interrelated subontologies, known as the profile, process model and grounding. In brief, the profile is used to express "what a service does", for the purpose of advertising, constructing service requests and matchmaking; the process model describes "how it works", to enable invocation and composition; and the grounding maps the constructs of the process model onto detailed specifications of message formats, protocols and so forth (Martin 2004). Figure 1 outlines these subontologies.

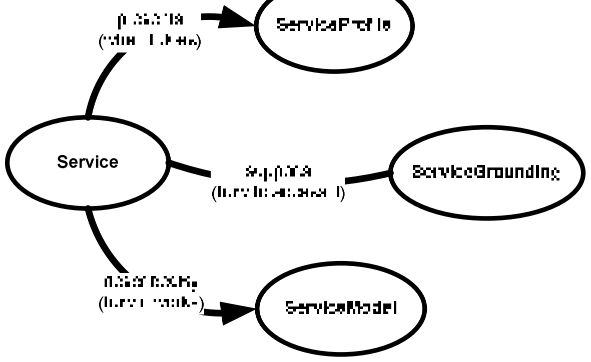


Figure 1. OWL-S subontologies

The OWL-S based approach facilitates the meaningful searches with the advantage of (IOPE) in profile and process based service model hence user can perform in-depth analysis of multiple services to perform a specific task.

3.2 Reasoning of the Service Semantics

Ontology based descriptions provides a mechanism to describe Web services functionality and the information useful for composition to be encoded in unambiguous machine understandable form. In order to perform the automated composition, an intelligent layer is essential that can interpret semantic descriptions and can order, combine and execute Web services to achieve the desired functionality or user goals. In other words, the intelligent layer should comprehend the descriptions in order to decide the possible services and build flow management for those services.

The semantics based approaches can be categorized based on the intelligent layer employed to achieve Web services discovery and composition. AI planning and case based reasoning are some of the methodologies employed as intelligent layer.

Artificial Intelligence Planning

This section discusses the relevancy of AI planning for the Web services composition problem and presents the literature survey on the subject.

Planning is a task of discovering a sequence of actions that can achieve a goal (Russell 2003). A planning problem can be described as a five-Tuple problem ***(S,s0,G,A,T)*** where ***S*** is the set of all possible states of the world, ***s0*** denotes the initial state of the planner, ***G*** denotes the set of goal states the planning system should attempt to reach, ***A*** is the set of actions the planner can perform in attempting to reach a goal state, and the transition relation ***T*** defines the semantics of each action by describing the state (or set of possible states if the operation is non-deterministic) that results when a particular action is executed in a given world state.

Web services composition is similar to planning problem evident from the following mapping.

S is the set of possible Web services, i.e. Web services available from the service registry
s0 is the initial state where some or none services are pre-selected for composition
G is the composition of Web services which satisfies the user requirements.
A is the Web services operations (I) or preconditions (P) available to planner to reach from the initial to goal state
T is the outputs (O) and effects (E) of invoking Web services operations.
AI planning dependent approaches use IOPE based OWL- S profile and process model to achieve required automation for the Web services composition. For example, if one starts with composition as goal (some desired outputs and effects), and matches it to the outputs and effects of a Web service (modelled as process), the result is an instantiation of the process, plus descriptions of new goals to be satisfied based on the inputs and preconditions of that process. The new goals (inputs and preconditions) then naturally match other processes (outputs and effects), so that composition arises naturally(Martin 2004).

Consistent with the above theory, Wu *et al* (Wu 2003) utilize DAML-S based descriptions, the previous version of OWL-S with SHOP2 planner (Kuter 2005). The SHOP2 is a Hierarchical Task Network (HTN) planner that creates plan by task decomposition - a process in which the planning system decomposes tasks into smaller and smaller subtasks, until primitive tasks are found that can be performed directly. The authors stress similarity between the concepts of task decomposition in HTN with the process decomposition in DAML-S.

Sirin *et al* in (Sirin 2004) describe another approach which couples OWL reasoner with AI planner to reason about the world state (effects and pre-condition) during planning. The reasoning is achieved by describing pre-condition and effects of the Web services using OWL.

Case Based Reasoning

Experience based learning using CBR is a relatively old branch of artificial intelligence and cognitive Science and is being used (Hammond 1986) as an alternative to rule-based expert system for the problem domains, which have knowledge captured in terms of experiences rather than rules. Case based reasoning for Web services were initially documented in (Limthanmaphon 2003), where the developed framework uses CBR for Web services composition. In their approach, the algorithm for Web services discovery and matchmaking is keyword based and has no notion for semantics. This affects the automation aspects for Web services search and later for composition. Similarly approach described in (Diaz 2006) proposes an extension of UDDI model for web services discovery using category-exemplar type of CBR, where web services are categorized in domains and stored as exemplar (Porter 1986) of particular domain. Their implementation of CBR reasoner facilitates UDDI registry by indexing the cases based on the functional characteristics of Web services. However, the approach does not take into consideration the importance of non-functional parameters in service selection and the use of semantics at CBR level is peripheral as they primarily use the UDDI based component for service discovery. UDDI is text-based leaving little scope for automation.

There is also a number of existing approaches which applies CBR for workflow modelling. (Madhusudan 2004) proposes an approach to support workflow modelling and design by adapting workflow cases from a repository of process models where workflow schemas are represented as cases and are stored in case repositories. The cases are retrieved for a problem which requires similar business process to solve the problem. The description and implementation language of framework is based on XML and main focus is on assisting workflow designer in creating business process flows. In similar line, (Cardoso 2005) represents adaptive workflow management system based on CBR and targets highly adaptive systems that can react themselves to different business and organization settings. The adaptation is achieved through the CBR based exception handling, where the CBR system is used to derive an acceptable exception handler. The system has the ability to adapt itself over time, based on knowledge acquired about past execution experiences that will help solve new problems.

3.3 Potential Facilitation to the composition participants

Despite the enthusiasm of the research community about the semantic web, there is still some way to go for creating a unifying framework facilitating the interoperation of intelligent agents or reasoning engines attempting to make sense of semantic Web services. However the workflow based approaches address here-and-now practical problem of Web services composition while dynamic Web services composition approaches holds better futuristic potential that can serve a great range of business domains. Automatic Web services composition has the potential to reduce development time and effort for the development of new applications. This is due to automatic re-configuration of changing or unavailable services in the integration.

Semantics assisted dynamic composition can serve all business domains for the possible B2B, EAI and B2C integrations. User can specify parameters for the successful composition and the composition can be performed at the run-time. The automatic Web services composition solution can address the problems of identifying candidate services, composing them, and verifying closely that they satisfy the request.

The service providers will be able to participate in the composition to their benefit with minimal effort as the development effort will be essentially reduced. The human developer will be taken out of the loop.

4. EVALUATION

For our research objectives, we have chosen the following criteria to study existing Web services composition approaches.

1. Service matchmaking

Using this evaluation criterion we compare various approaches based on how the service matchmaking is performed. The possible options are discovery using WSDL, UDDI, free-text or OWL-S (previously DAML-S) profile and process.

Workflow-based approaches use WSDL files to interpret the capability of a service coupled with the communications with the service provider or manual analysis of service parameters. AI planning, CSP, and agent-based approaches use different algorithms that utilize semantic web services profiles to match-make with semantically-encoded problem requests. CBR based approaches are so fare using UDDI to match-make web services.

2. Composition

We use this criterion to compare existing approaches to evaluate them based on the how they employ intelligent layers to achieve composition of Web services.

Workflow-based approaches use web services workflow languages such as BPEL and WS-CDL to outline the workflow of Web services. AI planning-based approaches utilize AI planner to form composition plans using existing planners such as SHOP2 (Levesque 1997) or GOLOG (Kuter 2005). CSP based approaches utilize existing standards WSDL, UDDI and BPEL to achieve the required composition. CBR based approaches use bespoke XML based workflow languages to write composition schema. Agents-based approaches model web services as agents so that the problem of web services composition translates to agent collaboration problem so that it is possible to utilize existing agent-infrastructure for composition.

3. Automation

Automation criterion is used to measure the level of automation achieved by various Web services composition approaches. We measure this using level of automation in the process of service discovery, composition and execution.

Most of these approaches support execution of composition schemes by providing execution engines, i.e., BPEL approaches use Oracle BPEL PM execution engine (Oracle 2005) or IBM BPWS4J (BPWS4J 2005), AI planners use OWL-S execution engines similar to the OWL-S API (Sirin 2004) provided by the University of Maryland.

Workflow-based approaches are static web services composition approaches involving manual intervention for discovery and composition of services. Semantic web based approaches achieve varying degree of automation in the process of composition (automatic discovery, semi-automatic composition).

4. Transparency

This criterion measures how transparent the process of composition (discovery, integration and execution) is from the composition participants.

For workflow-based approaches, end-user is transparent from the fact that the service presented to them in response to their request is a composed service, however the provider and composer has to work closely to integrate services in the workflow hence making the process opaque to them.

For AI planning based approaches, service requestor is transparent to the intelligent process of composition; however the process is semi-transparent to other participants. For example, the composer needs to be involved in the process of domain knowledge development and maintenance while tools assist them in converting semantic web services processes into planner domains. This knowledge is supplied to the planner in terms of operators and methods of services in order for planner to build composition plans. The service provider has to provide semantically enabled service but is transparent from the process of composition. Similarly, other semantic web based approaches offer complete transparency to end-users while requires some level of attention from service providers and composers.

5. Scalability of composition

Composing two services, however, is not the same as composing 10 or 100. In a real-world scenario, end users will typically want to interact with many services — consider the classic holiday booking scenario — while enterprise applications will invoke chains of possibly several hundred services Milanovic 2004).. Therefore, one of the critical issues is how the proposed approaches scale with the number of services involved. In BPEL, multiple service composition is somewhat tedious because XML files start to grow offering the approaches relying on BPEL as final composition scheme limited scalability (CSP based approach). OWL-S has similar issues and is propagated to the approaches that rely on using OWL-S process as final composition scheme (i.e., AI planning, software agent). Approaches that utilize bespoke XML schemas for final composition scheme (i.e., software synthesise approaches output synthesized XML schemas) also face similar challenges.

5. CONCLUSIONS

This contribution provides survey of two prominent categories of Web services composition approaches. The first approach, largely endorsed by the industry, borrows form business processes' workflow management theory to achieve the formalization necessary for describing the data flow and control in the composition scheme. The second approach, mainly promoted by the research community, aspires to achieve more dynamic composition by semantically describing the process model of Web service and thus making it comprehensible to reasoning engines or software agents.

The comparison made in this paper has shown that workflow based approaches are preferred by organizations as here-and-now and practical, albeit static, composition technique that robustly supports their business needs; while dynamic Web services composition approaches holds better futuristic potential that can serve a great range of business domains. In such kind of composition participating services can be external and public. User can specify parameters for the successful composition and the composition is performed at the run-time. The solution addresses the problems of identifying candidate services, composing them, and verifying closely that they satisfy the request.

At the end of this literature survey we concluded that despite the enthusiasm of the research community about the semantic web, there is still some way to go for creating a unifying framework facilitating the interoperation of intelligent agents or reasoning engines attempting to make sense of semantic Web services.

REFERENCES

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S. 2003. Business Process Execution Language for Web Services Version 1.1.

BPWS4J, "Business Process Execution Language for Web Services Java Run Time". http://www.alphaworks.ibm.com/tech/bpws4j.

Cardoso, J., Sheth, A. 2005. Adaptation and Workflow Management Systems. International Conference WWW/Internet. Lisbon, Portugal: 356-364.

Dean, M., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. F., Stein, L. A. 2005. Semantic Markup for Web Services, OWL-S version 1.1.

Hammond, K. 1986. Learning to anticipate and avoid planning problems through the explanation of failures. Fifth Conference on Artificial Intelligence, AAAI86. Philadelphia, USA, Morgan Kaufmann. 1: 556-560.

High, R., Kinder, S., Graham, S. 2004. An Architectural Introduction and Overview. IBM's SOA Foundation

Kavantzas N et al, 2004. Web Services Choreography Description Language (WS-CDL) Version 1.0, http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/

Kuter, U., Sirin, E., Nau, D., Parsia, B., Hendler, J. 2005. "Information gathering during planning for web service composition." Journal of Web Semantics 3(2-3): 183–205.

Levesque, H., Reiter, R., Lespérance, Y., Lin, F., Scherl, R. 1997. "GOLOG: A logic programming language for dynamic domains." Journal of Logic Programming 31: 59-84.

Leymann F at al. 2002. "Web Services and business process management". *IBM Systems Journal*, Volume 41-2, 2002, 198-211.

Limthanmaphon, B., Zhang, Y. 2003. Web service composition with case-based reasoning. the Fourteenth Australasian database conference on Database technologies. K. Schewe, Zhou, X Adelaide, Australia, Australian Computer Society, Inc. Darlinghurst, Australia. 143: 201 – 208.

Madhusudan, T., Zhao, L. J., Marshall, B. 2004. "A case-based reasoning framework for workflow model management." Data & Knowledge Engineering archive 50(1): 87-115.

Martin D et al, 2004, "Bringing Semantics to Web Services: The OWL-S Approach", *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*.

Milanovic, N., Malek, M .2004. "Current solutions for Web service composition." IEEE Internet Computing 8(6): 51 - 59.

Oracle BPEL Process Manager (PM), http://www.oracle.com/technology/products/ias/bpel/index.html.

Osman T et al, 2005. An Integrative Framework for Traffic Telematics Web Services, to be appeared in the *Parallel and Distributed Computing Networks Conference(PDCN 2005).*

Peer, J. 2004. PDDL based Tool for Automatic Web Service Composition. In Principles and practice of semantic web reasoning, Springer Verlag, Berling, Germany: 15.

Porter, B. W., Bareiss, R. E. 1986. PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. First International Meeting on Advnces in Learning (IMAL) Les Arcs, France: 159-174.

Russell, S., Norvig, P. 2003. Artificial Intelligence: A Modern approach, Prentice Hall.

Sirin, E., Parsia, B. 2004. Planning for Semantic Web Services. In Semantic Web Services Workshop at 3rd International Semantic Web Conference. Hiroshima, Japan, Springer Verlag, Berlin, Germany.

Thatte, S. 2001. XLANG Web Services for Business Process Design.

van der Aalst, W. M. P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P. (2003). "Workflow Patterns." Distributed and Parallel Databases 14(3): 45.

WSIF (2005). WSIF Apache Software Foundation Web Services project.

Wu, D., Parsia, B., Sirin, E., Hendler, J., Nau, D. (2003). Automating DAML-S web services composition using SHOP2. 2nd International Semantic Web Conference (ISWC2003).

Zisman, M. D. (1977). Representation, Specification and Automation of Office Procedures. Warton School of Business, University of Pennsylvania. PhD.

AUTHOR BIOGRAPHIES

DHAVALKUMAR THAKKER took his Masters degree in Data Communication Systems at Brunel University, London-UK. He also received his bachelors degree in engineering from India. He is a PhD student of Nottingham Trent University, School of computing and Informatics. His current research interests are distributed computing technologies, Web Services composition, Semantic Web and Ontologies. His email address is dhavalkumar.thakker@students.ntu.ac.uk.

TAHA OSMAN is senior lecturer at the Nottingham Trent University, UK. He received a B.Sc honours degree in Computing from Donetsk Polytechnical Institute, Ukraine in 1992. He joined the Nottingham Trent University in 1993 where he received an MSc in Real-time Systems in 1994 and a PhD in 1998. His current research peruses the utilisation of semantic web technologies in web services composition and information retrieval. His email address is taha.Osman@ntu.ac.uk.

DAVID AL-DABASS is visiting professor at the Nottingham Trent University, UK. His research work explores algorithm and architecture for machine intelligence. His email address is david.al-dabass@ntu.ac.uk and web page is http://ducati.doc.ntu.ac.uk/uksim/dad/webpage.htm.

Special Session on Pattern Analysis and Recognition (PAR'08)

Robust Recognition of Checkerboard Pattern for Deformable Surface Matching in Multiple Views

Weibin Sun Xubo Yang Shuangjiu Xiao Wencong Hu
DALAB Shool of Software
Shanghai Jiao Tong University, Shanghai, China 200240
Email: {skyend, yangxubo, xiaosj, ycguyue}@sjtu.edu.cn

KEYWORDS

Recognition checkerboard deformable surface matching

ABSTRACT

Checkerboard pattern can be used in many computer vision areas by matching the pattern as a surface, such as camera calibration, stereo vision, projector-camera system and even surface reconstruction. However, most existing checkerboard pattern recognition methods only work in planar and fine illuminating circumstances. A robust recognition method for checkerboard pattern is proposed in this paper to deal with those arbitrary surface deformation and complex illumination problems. Checkerboard internal corners are defined as special conjunction points of four alternating dark and bright regions. A candidate corner's neighbor points within a rectangular or a circular window are treated as in different one-point-width layers. By processing the points layer by layer, we transform the 2D points distribution into 1D to detect corners, which simplifies the regions amount counting and also improves the robustness. After corner detection, the pre-known checkerboard grids rows and columns amounts are used to match and decide the right checkerboard corners. Regions boundary data produced during the corner detection also assist the matching process. We compare our method with existing corner detection methods, such as Harris, SUSAN, FAST and also with the widely adopted checkerboard pattern recognition method, FindChessboardCorners function in OpenCV, to show the robustness and effectiveness of our approach.

INTRODUCTION

Checkerboard pattern(fig. 1) can be useful in many vision systems. Its alternating bright and dark grids and grid corners features can be a very strong features to detect and recognize. In most checkerboard applications, the internal corners(marked by circles in fig. 1), which are conjunctions of every four close grids, are the main features to use. Each checkerboard grid is a solid intensity region where the internal corner is surrounded by four alternating bright and dark regions, so we call these internal corners the region corners in this paper. By detecting and locating region corners of checkerboard pattern surfaces in two scenes, the mapping between those

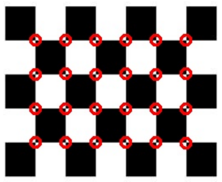


Figure 1: Checkerboard pattern with internal corners marked by circles

corresponding corners on the deformed surfaces of different scenes can be calculated. The mapping between those two deformed surfaces actually consists of many homographies between the approximate planar quadrangle grids of checkerboard patterns in different scenes. If both of the two scenes are camera views of a static checkerboard pattern on an object, the mapping can be used to model the object in this case, which is actually a modeling technique in stereo vision. If one of those scenes is a camera view of a planar checkerboard pattern and the other is the original checkerboard image, the mapping can be used in camera calibration case (Zhang, 2000). If the checkerboard pattern in the camera view in the second case is projected onto a surface by a projector, the mapping can be the foundation of geometry registration (Brown et al., 2005) in projector-camera augmented reality system (Bimber and Raskar, 2006; Raskar et al., 2002) case. What's more, in the geometry registration case, by treating each checkerboard grid as a quadrangular patch, we can reconstruct the projection surface, then checkerboard pattern can be a special structure light (Aggarwal and Wang, 1988; Stockman and Hu, 1986) type.

However, to recognize the checkerboard pattern is very difficult on deformable surfaces under natural lighting condition, which are very common environment problems in various computer vision tasks. This paper will deal with these problems.

RELATED WORKS

The recognition process is actually the locating process of checkerboard region corners. All checkerboard region corners should be detected and located without false corners and corners that do not belong to the checkerboard.

To detect corners, (Harris and Stephens, 1988) developed (Moravec, 1977)'s idea into the famous Plessey corner detector. This method is based on the first-order derivatives and has good behavior with respect to detection. (Smith and Brady, 1997) applied a circular mask to detect corners, that is, the so called SUSAN detector, which has the advantages of being robust to noises and yielding accurate results at a reasonable computation speed. The SUSAN principle is based on the fact that the center pixel should be a corner point if the number of the pixels that have the same brightness as the center pixel in the circular mask is below a threshold. (Rosten and Drummond, 2006)developed their FAST method for corner detection based on their previous works(Rosten and Drummond, 2005). FAST can perform efficient corner detection at high speed, however, its threshold to decide the dark and bright areas in a circle around a candidate corner weaken the robustness against complex illumination. Apart from this method, there are various other ones being proposed in (Trajkovic and Hedley, 1998; Zheng et al., 1999) to find corners or features. But when those methods were applied for checkerboard recognition under complex illumination and deformation conditions, many noise and redundant corner points were detected. Both problems would bring great trouble to identify the real checkerboard corners.

Besides those corner detection approaches, there are some specialized methods for checkerboard recognition. In (Zhang, 2000), the corners are found by intersecting lines. The drawback of this approach is that edges may be in general curved due to radial distortions or deformation. Furthermore, the subsequent ordering of the corners into a regular grid can be complex and unreliable. (Bouguet, 2000) proposed an interactive method to find the internal corners of planar checkerboard pattern image. This tool is especially convenient when working with a large number of images. However, the user needs to click on the four extreme corners on each rectangular checkerboard pattern image in order to calculate the data of the corners. The OpenCV function, cvFindChessBoardCorner, which is widely used, can do automatic corner extraction, but the algorithm fails rather often under complex illumination and deformation. (Wang et al., 2007) proposed an approach to automatically recognize and locate the internal target corners of the planar checkerboard pattern image. The proposed approach is based on the characteristics of local intensity and the grid line architecture of the planar checkerboard pattern image. (Shu et al., 2003) proposed a method, which is based on the algorithm of (Watson, 1981), that exploits the topological structure of the checkerboard pattern. The main idea is to use Delaunay triangulation (Bern and Eppstein, 1992) to connect the corner points. It can deal with different lighting conditions but also only planar checkerboard pattern.

In this paper, we propose a robust automatic method that can recognize checkerboard pattern under complex illumination and deformation. The main idea is to treat a certain point's neighbor points within a rectangular or a circular window as in different one-point-width layers and transform the 2D points distribution into 1D to detect regions. Corners are correlated and clustered by the region boundary data to recognize the checkerboard corners.

NOTATIONS AND MOTIVATION

Let $\mathbb{I}$ be the image containing the checkerboard pattern.

$$\mathbb{I} = \{\vec{p}\}, \quad \vec{p} = [x, y] \text{ denotes the point in } \mathbb{I}. \tag{1}$$

For a vector point $\vec{p}$, we use $(x_{\vec{p}}, y_{\vec{p}})$ to represent its Cartesian coordinates, $(\rho_{\vec{p}}, \theta_{\vec{p}})$ to represent its polar coordinates. In a grayscale image, we make $I(\vec{p})$ or $I(x_{\vec{p}}, y_{\vec{p}})$ be the intensity of point $\vec{p}$, $I_B(\vec{p})$ or $I_B(x_{\vec{p}}, y_{\vec{p}})$ be the intensity of point $\vec{p}$ in a binary image.

A checkerboard internal corner, which we call it the region corner(fig. 2(a)), has four alternating dark and bright regions around it. We define a rectangular and a circular window covering the four regions feature surrounding a candidate corner $\vec{p}$ to be $\mathbb{R}(\vec{p}, w)$ and $\mathbb{C}(\vec{p}, w)$ in eq. 2.

$$\begin{aligned} \mathbb{R}(\vec{p}, w) = &\quad \{\vec{p}_i | \, |x_{\vec{p}_i} - x_{\vec{p}}| \leq w, |y_{\vec{p}_i} - y_{\vec{p}}| \leq w\} \\ \mathbb{C}(\vec{p}, w) = &\quad \{\vec{p}_i \mid \|\vec{p}_i - \vec{p}\| \leq w\} \end{aligned} \tag{2}$$

w here is always an non-negative integer limit the window's scope. $\mathbb{R}(\vec{p}, w)$ is a set of points that are in a rectangular, actually a $2w + 1$ points width square window(we still call it rectangular to be more general in the rest of this paper). $\mathbb{C}(\vec{p}, w)$ is a set containing the points around $\vec{p}$ within a circular window whose radius is w points and the center is $\vec{p}$. The circular window can be generated by a Bresenham (Bresenham, 1977) circle when implementing to be efficient and accurate.

The four alternating dark and bright regions around a corner can be deformed seriously on arbitrary surface. The isotropy against deformation can be achieved if points within a window are iterated by circumambulating the corner from the outer to the inner. The layer is defined in eq. 3 to represent the points being checked in each circumambulating iteration.

$$\begin{aligned} \mathbb{L}_r(\vec{p}) = < \vec{p}_1, \ldots, \vec{p}_i, \ldots, \vec{p}_{n_r} >, i \in [1, n_r] \\ \text{for circular window } \vec{p}_i \in \mathbb{C}(p, r) - \mathbb{C}(p, r-1) \\ \text{for rectangular window } \vec{p}_i \in \mathbb{R}(p, r) - \mathbb{R}(p, r-1) \\ \theta_{\vec{p}_j} < \theta_{\vec{p}_{j+1}}, j \in [1, n_r - 1] \end{aligned} \tag{3}$$

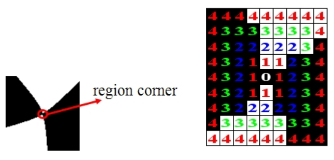
In a window $\mathbb{R}(p, w)$ or $\mathbb{C}(p, w)$, there are w layers from $\mathbb{L}_1(\vec{p})$ to $\mathbb{L}_w(\vec{p})$. n_r is the points amount in $\mathbb{L}_r(\vec{p})$. For a $\mathbb{L}_r(\vec{p})$ in $\mathbb{R}(p, w)$, n_r can be calculated by eq. 4.

$$n_r = 4(2r + 1) - 4 = 8r \tag{4}$$

For circular window layers $\mathbb{L}_r(\vec{p})$ in $\mathbb{C}(p, w)$, the Bresenham circle algorithm will decide the n_r. To be used later, we define the first order derivative in a layer $\mathbb{L}_r(\vec{p})$ to be:

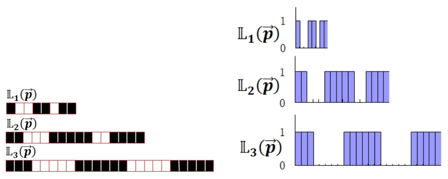
$$I'_L(\vec{p}_i) = I(\vec{p}_{i+1}) - I(\vec{p}_i) \tag{5}$$

Note that the coordinate index in $\mathbb{L}_r$ should be i in $\vec{p}_i$, this would make the coordinates in $\mathbb{L}_r$ be 1D. A layer



(a) Region corner (b) Layers

Figure 2: A region corner and its layers within a window $\mathbb{R}(\vec{p}, 4)$, from $\mathbb{L}_1(\vec{p})$ to $\mathbb{L}_4(\vec{p})$, points are labeled with their layer indices.



(a) 1D layers image (b) 1D layers brightness values

Figure 3: 1D form of layers from $\mathbb{L}_1(\vec{p})$ to $\mathbb{L}_3(\vec{p})$ in fig 2(b).

is an ordered tuple of points which are sorted by the sequence in which points are iterated when circumambulating the corner. The circumambulating can be performed in the order that the polar coordinates $\theta_{\vec{p}}$'s incremental direction(note that within each window, we set the polar coordinates origin $(0, 0)$ be the center of the window). fig. 2(b) shows the layers in a rectangular window around a corner. A layer $\mathbb{L}_r(\vec{p})$ is actually a n_r points long 1D sequential array representing a 1D image signal(fig. 3). It transforms the 2D point distribution within a window to 1D, then regions in a layer will be line segments after binarization. That will simplify the detection of regions around a corner when corners are detected by recognizing the four alternating regions feature in our method.

To be used by the *ring-morphology* defined in (Section CORNER DETECTION), we define some operators for set operations:

$$\begin{aligned} \text{Set cardinality:} \quad & |A| \\ \text{Translation:} \quad & (A)_p = \{c \mid c = a + p, a \in A\} \\ \text{Complementary set:} \quad & A^c = \{w \mid w \notin A\} \end{aligned} \tag{6}$$

RECOGNITION

Checkerboard recognition has two steps, corner detection and checkerboard match. The first step finds all candidate checkerboard corners. The second step does the checkerboard pattern matching with the help of corners.

CORNER DETECTION

Region corners are detected by checking whether there are four alternating dark and bright regions around a candidate corner within a window scope. To be efficient and robust, firstly, the image $\mathbb{I}$ is resized to a range $[IS_{min}, IS_{max})]$ without changing the width and height ratio. According to the common camera resolutions from 320×240 to 2048×1536, the range is set to be $[(300, 200), (2100, 1600)]$. There is no difference between the rectangular window and the circular one to achieve isotropy against deformations when circumambulating the points in layers around the candidate corner. We will use the rectangular one in rest of this paper.

The window size parameter, w in $\mathbb{R}(\vec{p}, w)$ is decided by the checkerboard grid size. To detect the four alternating dark and bright regions, the window should not cover more than four regions, which are actually four grids surrounding a corner. So window width $2w + 1$ should be less than two times of the grid width(we only consider checkerboard with square grid) g_{width}. To be efficient, we decide the two windows of two neighbor corners belonging to a same grid should not intersect too much. So $2w + 1 < g_{width}$ and $w \approx g_{width}/2$. We also define a a window size range $[WS_{min}, WS_{max}]$ to limit the window size for extreme large or small grid. g_{width} can be calculated by the amount of checkerboard grids rows g_r and columns g_c and the image size $width_{\mathbb{I}}$ and $height_{\mathbb{I}}$ according to eq. 7 to be a probable value.

$$g_{width} = \min\{\frac{width_{\mathbb{I}}}{g_c}, \frac{height_{\mathbb{I}}}{g_r}\} \tag{7}$$

Here g_r and g_c should be known before recognition. eq. 8 shows the calculating of the window size parameter w.

$$w = \frac{w'}{2}, w' \text{ is calculated by eq. 9.} \tag{8}$$

$$w' = \begin{cases} WS_{min} & \text{if } g_{width} < WS_{min} \\ g_{width} & \text{if } g_{width} \in [WS_{min}, WS_{max}] \\ WS_{max} & \text{if } g_{width} > WS_{max} \end{cases} \tag{9}$$

The window size parameter does not need high precision, we just ignore the 1 in $2w + 1$ window width in eq. 8. The window size range $[WS_{min}, WS_{max}]$ reflects the minimum scope in which a region corner can be identified. The region borders around a corner may be smoothed or extended to blocks rather than border lines to lose sharpness because of noise, low camera capturing quality and complex lighting. To get robustness, the window should be able to cover all four regions to detect them. However, too large window size will cause low performance. We set this range be $[11, 21]$ to tolerate the low quality corners in most common camera cases. The $[WS_{min}, WS_{max}]$ with value of $[11, 21]$ limits the window widths from 11 to 21 of $\mathbb{R}(\vec{p}, w)$. This range may be changed when implementing, however, a points block with over 21 points long width should not be recognized to be a border line even in human's eyes, so for most applications, the range can be fixed to our values without manual change.

For each layer, a mean value of the layer points is calculated to do thresholding as in eq. 10. Let $\mathbb{L}_r^B(\vec{p})$ be the

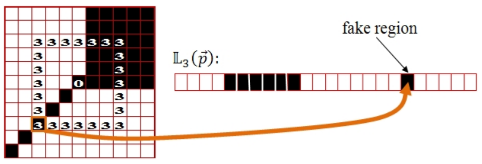


Figure 4: A fake region caused by a line

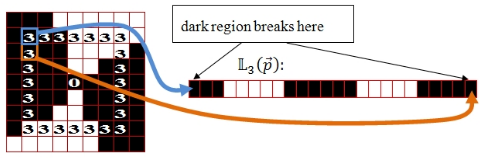


Figure 5: Broken region in $\mathbb{L}_3(\vec{p})$ of the region corner in fig. 2

binarization result of $\mathbb{L}_r(\vec{p})$.

$$\mathbb{L}_r^B(\vec{p}) = \{\vec{p'}_i \mid \vec{p'}_i = \vec{p}_i, \forall \vec{p}_i \in \mathbb{L}_r(\vec{p}) \text{ and }$$
$$I_B(\vec{p'}_i) = \begin{cases} 0 & \text{if } I(\vec{p}_i) < t \\ 1 & \text{if } I(\vec{p}_i) \geq t \end{cases}, t = \frac{\sum_{\vec{p}_i \in \mathbb{L}_r(\vec{p})} I(\vec{p}_i)}{|\mathbb{L}_r(\vec{p})|}\} \tag{10}$$

This layer-scope threshold can reduce the negative effects caused by noises to achieve a locally adaptive thresholding result. After thresholding, a noise reduction is performed on the binary 1D image signal to remove fake regions, which are mainly caused by lines or blob points. fig. 4 shows an example of this fake region caused by a line. These fake regions are actually regions with very short widths. We do the binary morphology (Haralick et al., 1987) operations, opening and closing, both of which are with a same 1D structure elements(SE) SE_{cd}, to remove noises. The SE_{cd}'s length is determined by the layer $\mathbb{L}_r^B(\vec{p})$'s length n_r. A proportion κ of the SE_{cd} length l_{se} and the layer $\mathbb{L}_r^B(\vec{p})$'s length n_r is used to calculate l_{se}(eq. 11).

$$l_{se} = n_r \cdot \kappa \tag{11}$$

SE_{cd} is defined to be:

$$SE_{cd} = \{1, \ldots, l_{se}\} \tag{12}$$

A checkerboard corner should have four regions, so κ should be less than $\frac{1}{4}$. To be robust, we decide to tolerate the deformed region covering about $\frac{1}{8}$ layer. To produce more redundant corners to avoid missing corners, we tolerate the minimum region taking up only $\frac{1}{10}$ layer. So we set κ to $\frac{1}{10}$. To avoid incorrect noise reduction, layers with small width parameter $r < 2$ will not do this operation. What's more, because the last point and the first point of a layer in 1D form are actually continuous in 2D form, a region may break here because of the 2D to 1D transformation. An example is shown in fig. 5. These broken regions may be treated as fake regions when noise reducing because of their short widths. To avoid this problem, we modify the binary morphology operations by elongating the layer to repeat it one more time behind its last point to be cycle-compatible, these modified operations are called the *ring-morphology* as in eq. 13.

$$\begin{aligned}
\text{Dilation:}\quad & A \oplus_{ring} B = \{\, x \textbf{ mod } |A| \mid \\
& \quad (B)_x \cap (A \cup (A)_{|A|}) \neq \emptyset\} \\
\text{Erosion:}\quad & A \ominus_{ring} B = \{\, x \textbf{ mod } |A| \mid \\
& \quad (B)_x \cap (A \cup (A)_{|A|})^c \neq \emptyset\} \\
\text{Opening:}\quad & A \circ_{ring} B = (A \ominus_{ring} B) \oplus_{ring} B \\
\text{Closing:}\quad & A \bullet_{ring} B = (A \oplus_{ring} B) \ominus_{ring} B
\end{aligned} \tag{13}$$

The *ring-morphology* treats a 1D image as a ring that the head and the end are joined. By performing *ring-morphology* opening and closing on the binary layers $\mathbb{L}_r^B(\vec{p})$ after thresholding, we can reduce the fake regions. Then regions count can be calculated by summing absolute value of the first order derivatives $I'_L(\vec{p}_i)$ of this binary(only 0 and 1 values) layer. To avoid the similar region breaking problem, we let the first order derivative $I'_L(\vec{p}_{n_r})$ at $\vec{p}_{n_r}$ of a layer $\mathbb{L}_r^B(\vec{p})$ to be:

$$I'_L(\vec{p}_{n_r}) = I_B(\vec{p}_1) - I_B(\vec{p}_{n_r}) \tag{14}$$

In checkerboard pattern, one layer should have 4 regions. If all layers of a candidate corner have 4 regions, this candidate should be a corner. However, to get high robustness, we define an acceptance threshold value α to allow some noises. If there are α or more portions of layers containing 4 regions, the candidate corner is accepted to be a checkerboard corner. Too large α will cause too many noise corners while too small α also reduces the robustness. This acceptance threshold value can be determined by the acceptance degree of human's eyes and image noise degree. We set it to be 0.7 to tolerate 30% noise layers in a window for common camera images.

Till now the corner detection result including the region corners is produced. Let the result be $\mathbb{N}$. Moreover, the regions boundaries positions of a corner are recorded for the later checkerboard match step. Since there are more than one layers within a corner's surrounding window, only the most outer layer is recorded. We represent the region boundary of a corner $\vec{p}$ within its window $\mathbb{R}(\vec{p}, w)$ to be(note that the derivative $I'_L(\vec{p})$ is on the thresholded binary layer $\mathbb{L}_r^B(\vec{p})$):

$$\begin{aligned}
\mathbb{B}_{\vec{p}} = \{\vec{p}_i \mid |I'_L(\vec{p}_i)| = 1, \forall \vec{p}_i \in \mathbb{L}^B_{w_{max}}(\vec{p})\} \\
w_{max} = \max\{w_j \mid w_j \in [1, w], \\
\sum_{\vec{p}_k \in \mathbb{L}^B_{w_j}(\vec{p})} |I'_L(\vec{p}_k)| = 4\}
\end{aligned} \tag{15}$$

CHECKERBOARD MATCH

The corner detection step can produce noise corners and redundant corners because of the acceptance threshold value and image noises. To reduce those noise, we perform a window neighbors checking on the corner detection result. To remove the redundant corners, firstly we cluster the result points, then in each cluster, the mass center or the point with minimum distances sum to other points of the same cluster is calculated to be the new checkerboard corner, other result corners are removed.

When doing noise reduction on the corner detection result $\mathbb{N}$, each corner $\vec{c}$'s neighbors within a window $\mathbb{R}(\vec{c}, w)$ is checked to see whether there are enough corners in the window. If there are enough(more than a given τ) corners, all corners within the window are reserved, otherwise the checked corner is removed. This operation is similar to binary erosion, we also need a $(2w_{se}+1) \times (2w_{se}+1)$ structure element SE_{cr} to perform the neighbors checking. By defining SE_{cr} in eq. 16, the refined result $\mathbb{N}'$ can be calculated by eq. 17, in which we also use the translation operation and set cardinality defined in eq. 6.

$$\begin{aligned} SE_{cr} = \{\vec{p}_{i,j} \mid i \in [1, 2w_{se}+1], j \in [1, 2w_{se}+1], \\ x_{\vec{p}_{i,j}} = i - w_{se} - 1, y_{\vec{p}_{i,j}} = j - w_{se} - 1\} \end{aligned} \tag{16}$$

$$\mathbb{N}' = \{\vec{p} \mid |(SE_{cr})_{\vec{p}} \cap \mathbb{N}| \geq \tau, \forall \vec{p} \in \mathbb{N}\} \tag{17}$$

The τ to reduce noise in eq. 17 should be no larger than w_{se}. We define w_{se} to be 2 to check a 5×5 neighbors patch and τ to be 2 to eliminate the isolated noise corner, which is the only one corner within that neighbors patch when τ is 2.

Redundant corners are reduced by clustering. Corners result $\mathbb{N}'$ after noise reduction is clustered according to their distances between each other. The result $\mathbb{N}'$ is treated to be a nodes set and each corner in $\mathbb{N}'$ is a node. If two corners have a distance no larger than SE_{cr}'s width $2w_{se}+1$, there is an edge connecting them. By defining nodes and edges, we get an undirected graph $\mathbb{G}'$ (eq. 18).

$$\begin{aligned} \mathbb{G}' = (\mathbb{N}', \mathbb{E}') \\ \mathbb{E}' = \{(\vec{p}_i, \vec{p}_j) \mid \|\vec{p}_i - \vec{p}_j\| \leq 2w_{se}+1, \forall \vec{p}_i, \vec{p}_j \in \mathbb{N}'\} \end{aligned} \tag{18}$$

Each connected component $\mathbb{G}'_c(\mathbb{N}'_c, \mathbb{E}'_c)$ in $\mathbb{G}'$ is a cluster. For each cluster $\mathbb{G}'_c$, the mass center or the point $\vec{p}_c$ with minimum distances sum to other points of the same cluster is calculated and made to be the new corner to replace others in the cluster. $\vec{p}_c$ is calculated by eq. 19. There may be more than one $\vec{p}_c$, just pick a random one when implementing.

$$\begin{aligned} \vec{p}_c : \vec{p}_c \in \mathbb{N}'_c \text{ and } \textstyle\sum_{\vec{p}_j \in \mathbb{N}'_c} \|\vec{p}_j - \vec{p}_c\| = \\ \min\{\textstyle\sum_{\vec{p}_k \in \mathbb{N}'_c} \|\vec{p}_k - \vec{p}_i\| \mid \forall \vec{p}_i \in \mathbb{N}'_c\} \end{aligned} \tag{19}$$

Because mass centers of some clusters may be not in $\mathbb{N}'$, the region boundary positions of those mass centers are not recorded in corner detection. This problem can be solved by calculating the mean region boundary positions of corners in a cluster. $\vec{p}_c$ does not have this problem although the mass center is always more accurate in position than $\vec{p}_c$ to represent the cluster's position. So when implementing, $\vec{p}_c$ will be an efficient selection. We let $\mathbb{N}''$ be the corners result after redundant corners reduction.

After noise and redundance reduction, the region boundary positions data of each found corner are used to calculate the connectedness of corners. We also use the graph theory to assist this process. Each corner in $\mathbb{N}''$ is a node. An edge connecting two corners will exist if those two corners are on a same region boundary line. An edge actually connects two checkerboard corners sharing the same boundary of one checkerboard grid. These two corners are neighbor corners of a grid. The edge set is defined by eq. 20.

$$\begin{aligned} \mathbb{E}'' = \{(\vec{p}_i, \vec{p}_j) \mid \exists \vec{p}_{i_k} \in \mathbb{B}_{\vec{p}_i}, \vec{p}_{j_l} \in \mathbb{B}_{\vec{p}_j} : \\ \frac{(\vec{p}_{i_k} - \vec{p}_i) \cdot (\vec{p}_{j_l} - \vec{p}_j)}{\|\vec{p}_{i_k} - \vec{p}_i\| \|\vec{p}_{j_l} - \vec{p}_j\|} \in [-1, \varepsilon]\} \end{aligned} \tag{20}$$

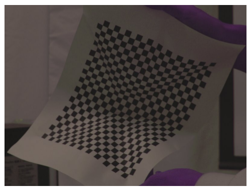
ε here should be very close to -1. $[-1, \varepsilon]$ defines an acceptance range to tolerate the inaccurate region boundaries. The inaccuracy is mainly caused by the serious deformation within a checkerboard grid. If two corners $\vec{p}_i$ and $\vec{p}_j$ are on a same boundary of a checkerboard grid, $[-1, \varepsilon]$ defines the cosine value(calculated by $\frac{(\vec{p}_{i_k} - \vec{p}_i) \cdot (\vec{p}_{j_l} - \vec{p}_j)}{\|\vec{p}_{i_k} - \vec{p}_i\| \|\vec{p}_{j_l} - \vec{p}_j\|}$) range of the angle between two region boundaries vectors: $(\vec{p}_{i_k} - \vec{p}_i)$ and $(\vec{p}_{j_l} - \vec{p}_j)$, here $\vec{p}_{i_k}$ and $\vec{p}_{j_l}$ are on the same grid boundary on which $\vec{p}_i$ and $\vec{p}_j$ locate. eq. 20 ignores the relative position of the two corners when deciding edges. In most application cases such as geometry registration, surface reconstruction, checkerboard usually has large amount of grids and a grid is treat to be a plane without deformation and distortion, deformation and distortion only occur among different grids. However, considering other cases that can tolerate the non-planar grid and to be robust under serious deformation, we decide ε to be -0.8 to tolerate a $\pm \arccos(-0.8) \approx \pm 36°$ bending deformation of straight boundaries within a grid.

Now the graph $\mathbb{G}'' = (\mathbb{N}'', \mathbb{E}'')$ is produced. With the assistance of checkerboard grids rows g_r and columns g_c, we can find the checkerboard pattern corners by finding the connected component $\mathbb{G}''_c$ that has exact $(g_r - 1) \cdot (g_c - 1)$ nodes. If there is the only one $\mathbb{G}''_c$ having $(g_r - 1) \cdot (g_c - 1)$ nodes, it is the set that contains all the right checkerboard corners.

RESULT AND COMPARISON

We compare our method with Harris, SUSAN and FAST corner detector to evaluate our corner detection of checkerboard region corners. We also compare our method with FindChessboarCorner function in OpenCV to evaluate our recognition of the checkerboard pattern.

Our comparison focuses on the corner detection robustness and checkerboard pattern recognition correctness. To evaluate the robustness against complex illumi-


(a) A printed checkerboard with serious deformation and bad illumination

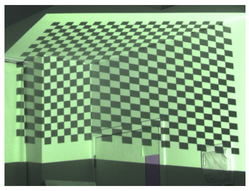

(b) A projected checkerboard pattern by a common projector with deformation and incorrect color capturing

Figure 6: Checkerboard images by common camera.(You may enlarge them to see more clearly.)

nation and deformed surface, we use the low-price general cameras to capture checkerboard patterns(in fig. 6) on serious deformed surfaces in a room without enough environment light. General cameras always have serious color incorrectness that they can not restore the exact color of the object it captured, which will cause various noises. fig. 7 shows our comparison results.

For Harris and SUSAN, both of them fail to deal with the darker area in fig. 7(b) and 7(a) while too many noises and redundance are produced in fig. 7(g) and 7(f).

FAST method's threshold weakens the robustness against complex illumination significantly. We use both the default value 20 and an adjusted value 10 in the FAST demo supplied by the authors of FAST. Under complex illumination in our test images(fig. 6), FAST with threshold 20 will miss many region corners(fig. 7(e) and 7(j)). We adjust this value to 10, however, it will produce many redundant and noise corners(fig. 7(d) and 7(i)). It is difficult to distinguish the four different corners on the boundaries of a grid.

OpenCV's FindChessboardCorner will do thresholding on the global image to transform the dark and bright grids into black and white. This thresholding can always fail even with adaptive method under complex illumination. The results in fig. 7(c) and 7(h) show the failed detection.

fig. 7(k) and 7(l) show that our method can find all checkerboard region corners and match them within the checkerboard pattern successfully while other methods fail. fig. 6(a) and 6(b) are just two representative images of a large number of checkerboard images captured by general cameras in our test. We test our method with plenty of checkerboard images captured by general cameras under complex illumination and deformation and find it is robust to recognize the checkerboard pattern correctly in those images. We also present some of them(fig. 7(m), 7(n), 7(o)) in this paper. The circumambulating iteration around a corner can get isotropy against deformation. Thresholding in 1D layers locally will get robustness against various illumination. The fake region reduction can ensure the correctness of corner detection. The acceptance threshold of layers and other parameters with the values defined in our method according to most general cases will reduce the effects caused by noise or deformation. Noise reduction in checkerboard match and the redundant corner reduction by clustering can ensure the corners without redundance and noise. Then corners connected by region boundaries will produce the right checkerboard corners set.

CONCLUSION AND FUTURE WORKS

We present a robust recognition method for checkerboard pattern. This method addresses the problem of recognizing checkerboard pattern under complex illumination and deformation in computer vision systems to do camera calibration, stereo vision, geometry registration or surface reconstruction. The checkerboard internal corners(region corners) surrounded by four alternating dark and bright regions and their boundaries are the features to find in our recognition. By detecting corners with the help of our own robust region corner detection approach and then do checkerboard matching mainly by clustering, our method can be automatic and robust with the parameters values we defined according to most general cases rather than manually-setting parameters. The experiment results in (Section EVALUATION) show that our method can deal with most common checkerboard images captured by general cameras.

To achieve the most robustness, we detect the region corners and match checkerboard pattern by recognizing and using almost all their features which can ensure the robustness but costs much time. Now our algorithm is $O(mn)$ time complexity(m for window size, n for image size). In future, we will focus on the speed. Some features may be simplified to save time without losing robustness.

ACKNOWLEDGEMENTS

This work is sponsored by 863 National High Technology R&D Program of China (No.2006AA01Z307) and National Natural Science Foundation of China (No.60403044).

REFERENCES

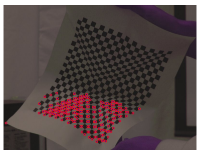
Aggarwal, J. and Wang, Y. (1988). Inference of object surface structure from structured lighting — an overview. *Machine Vision — Algorithms, Architectures, and Systems, (Proceedings of a workshop 'Machine Vision: Where are we Going")*.

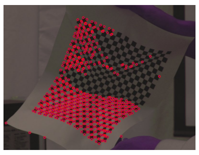
Bern, M. and Eppstein, D. (1992). Mesh generation and optimal triangulation. *Computing in Euclidean Geometry*, pages 23–90.

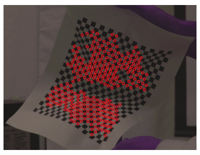
Bimber, O. and Raskar, R. (2006). Modern approaches to augmented reality. *ACM SIGGRAPH*.

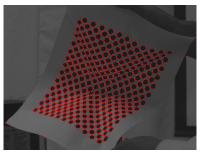
Bouguet, J. (2000). Matlab camera calibration toolbox. .

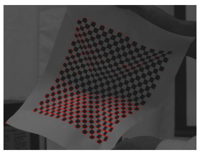
Bresenham, J. (1977). A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, 20:100–106.

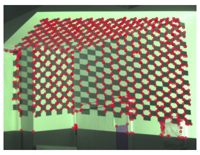

(a) SUSAN result, fail to find all region corners.

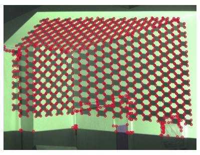

(b) Harris result, fail to find all region corners.

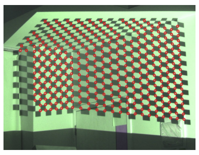

(c) OpenCV FindChessboardCorner result, fail to find all region corners and match checkerboard.

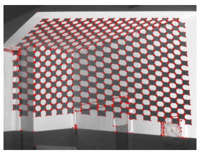

(d) FAST with threshold 10. Too many redundant and noise corners to match checkerboard.

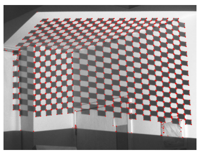

(e) FAST with default threshold 20, fail to find all region corners.

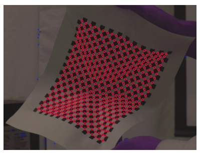

(f) SUSAN result, fail to find all region corners.

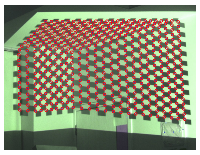

(g) Harris result, fail to find all region corners.

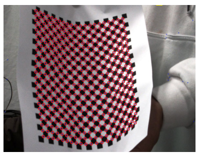

(h) OpenCV FindChessboardCorner result, fail to find all region corners and match checkerboard.

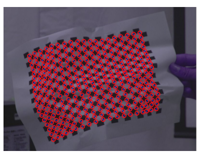

(i) FAST with threshold 10. Too many redundant and noise corners to match checkerboard.


(j) FAST with default threshold 20, fail to find all region corners.


(k) Our method result, success.


(l) Our method result, success.


(m) Our method result on other image, success.


(n) Our method result on other image, success.

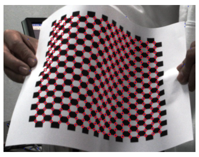

(o) Our method result on other image, success.

Figure 7: Comparison Result on fig. 6(a), 6(b) and some of our method results on other images. Corners are marked by blue cross. Red circles mark the checkerboard pattern's region corners. SUSAN and Harris corners are marked by red circle. FAST corners are marked by red pixels.(Enlarge images to see detail.)

Brown, M., M., A., and Yang, R. (2005). Camera-based calibration techniques for seamless multiprojector displays. *IEEE Transactions on Visualization and Computer Graphics*.

Haralick, R., Sternbergn, S., and Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. *Proc 4th Alvey Vision Conf*, pages 189–192.

Moravec, H. (1977). Towards automatic visual obstacle avoidance. *Fifth International Joint Conference on Artificial Intelligence*.

Raskar, R., van, B. J., and Chai, J. (2002). A low-cost projector mosaic with fast registration. *ACCV*.

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. *ICCV*.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. *ECCV*, 3951:430–443.

Shu, C., Brunton, A., and Fiala, M. (2003). Automatic grid finding in calibration patterns using delaunay triangulation. .

Smith, S. and Brady, J. (1997). Susan - a new approach to low level image processing. *IJCV*, 23:45–78.

Stockman, G. and Hu, G. (1986). Sensing 3-d surface patches using a projected grid. *CVPR*.

Trajkovic, M. and Hedley, M. (1998). Fast corner detection. *Image and Vision Computing*, 16.

Wang, Z., Wu, W., Xu, X., and Xue, D. (2007). Recognition and location of the internal corners of planar checkerboard calibration pattern image. *Applied Mathematics and Computation*, 185:894–906.

Watson, D. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2):167.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22.

Zheng, Z., Wang, H., and Teoh, E. (1999). Analysis of gray level corner detection. *Pattern Recognition Letters*, 20(2):149–162.

AUTHOR BIOGRAPHIES

Weibin Sun was born in 1984's China. His email is skyend@sjtu.edu.cn and his personal webpage at http://dalab.se.sjtu.edu.cn/ endysun.

Xubo Yang's email is yangxubo@sjtu.edu.cn.
Shuangjiu Xiao's email is xiaosj@sjtu.edu.cn.
Wencong Hu's email is ycguyue@sjtu.edu.cn.

FURTHER OPTIMIZATIONS FOR THE CHAN-VESE ACTIVE CONTOUR MODEL

Zygmunt L. Szpak and Jules R. Tapamo
School of Computer Science
University of KwaZulu-Natal
Durban, 4062, Republic of South Africa
Email: zygmunt.szpak@cs.ukzn.ac.za and tapamoj@ukzn.ac.za

KEYWORDS

Fast level-set, Optimization, Active Contours Without Edges, Chan-Vese, Shi-Karl, Segmentation

ABSTRACT

When a Chan-Vese active contour model is implemented in a framework that does not solve partial differential equations, we show how the mean pixel intensity inside and outside the curve can be updated efficiently. We reduce each iteration of the Chan-Vese active contour by O(n), when compared to an approach whereby the mean pixel intensities are recalculated for each iteration by looping over the entire image. After implementing the Chan-Vese active contour in the Shi-Karl level-set framework that does not solve partial differential equations, we show that the active contour may be trapped in an idempotent cycle, and we introduce a new stopping criterion to deal with this situation, thereby eliminating wasted computation cycles. Finally, we optimize the regularization cycle in the Shi-Karl framework, by detecting when an additional execution of the regularization cycle has no effect on the active contour, and breaking out of the loop.

1. INTRODUCTION

In (Lakshmanan et al., 2006) and (Pan et al., 2006a), the authors propose efficient implementations for the Chan-Vese *Active Contours Without Edges* model (Chan and Vese, 2001), in a level-set framework that does not require the calculation of partial differential equations. These implementations reduce computation time by using lists of points to represent the contours, and by evolving the contours by adding and removing points from these lists; while concurrently updating a level-set function. Previous work focused on modelling the evolution of the curve, and no mention was made of how to efficiently calculate the mean pixel intensity inside and outside the curve. This is a crucial calculation in the Chan-Vese model. We show how the mean intensity inside the curve and outside the curve can be efficiently updated for each iteration of the curve evolution, and we compare the resulting speed increase against an approach where the mean intensities are recalculated over the entire image at each iteration. Our proposed calculation scheme can be used whenever the mean intensities inside and outside the curve are required, and a list of points representing the curve is used. We implement our solution in the Shi-Karl (Shi and Karl, 2005a) fast level-set framework, and emphasize the speed increase of our method by testing it on large 1024×768 images.

Additionally, we propose a simple modification to the gaussian filtering curve regularization method, used in (Shi and Karl, 2005a), which eliminates redundant cycles and hence decreases the time to convergence. We also show that for certain images, a gaussian filtering regularization method will prevent the active contour from terminating. To overcome this problem we introduce an additional stopping criterion.

The rest of our paper is organized as follows. In Section 2 we summarize the Shi-Karl fast level-set framework, that does not require the calculation of partial differential equations. We review the Chan-Vese piecewise-constant active contour model, and discuss its use and limitations in Section 3. Our proposed optimization for calculating the mean pixel intensity, inside the curve and outside the curve, is presented in Section 4, and our new stopping criterion and optimization for the gaussian filtering regularization method, is introduced in Section 5.

2. SHI-KARL FAST LEVEL-SET METHOD

The level-set method is a numerical technique, for tracking a propagating interface which changes topology over time. It is often used in image segmentation (Paragios and Deriche, 2000). A segmentation is achieved by placing a closed curve on an image, and by evolving the curve according to internal, external and user defined forces. Snakes (Kass et al., 1987) are used in a similar way to segment an image, but in the level-set method, the curve can split and change topology whereas snakes cannot. The traditional level-set method (Sethian, 1999), requires the calculation of partial differential equations, that govern the evolution of the curve. This is a very time consuming calculation. In (Shi and Karl, 2005a), the authors proposed a fast implementation of the level-set method, that does not require the calculation of partial differential equations. Their framework resembles the traditional level-set method, because the curve C is still represented implicitly as the zero level-set of a function ϕ. The function ϕ is defined as the signed distance function, which is positive outside C and negative inside C. From this definition, when a point on C moves inward, its neighboring

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

point that previously was inside the curve C, will lie outside the curve, and so the value of ϕ of that neighboring point will change from negative to positive. Similarly if a point on C moves outwards, the value of ϕ of its outside neighboring point will change from positive to negative. This means that the evolution of the curve C can be controlled without solving partial differential equations, by manipulating the values of ϕ for a list of neighboring points outside C (L_{out}), and a list of neighboring points inside C (L_{in}). Formally, the two lists of neighboring points can be defined as:

$$L_{in} = \{x|\phi(x) < 0 \ \ and \ \ \exists \, y \in N_4(x), \ \phi(y) > 0\} \tag{1}$$

$$L_{out} = \{x|\phi(x) > 0 \ \ and \ \ \exists \, y \in N_4(x), \ \phi(y) < 0\}, \tag{2}$$

where $N_4(x)$ is the discrete 4-connected neighborhood of a pixel x.

To approximate the signed distance function, ϕ is defined as:

$$\phi(x) = \begin{cases} 3, & \text{if } x \text{ is outside } C \text{ and } x \notin L_{Out}; \\ 1, & \text{if } x \in L_{Out}; \\ -1, & \text{if } x \in L_{In}; \\ -3, & \text{if } x \text{ is inside } C \text{ and } x \notin L_{In}. \end{cases} \tag{3}$$

The curve is evolved by switching neighboring pixels between the two lists L_{in} and L_{out}, based on an external speed function F_{ext}, an internal speed function F_{int}, and by updating the level set function ϕ. The external speed is used to attract the curve to the regions of interest, while the internal speed is used to regularize the evolution of the curve so that the curve remains smooth.

The evolution of C is split into two different cycles. In the first cycle, C is evolved according to the external speed; a positive value of F_{ext} moves a point on C outwards (by switching the point from L_{out} to L_{in}), while a negative value of F_{ext} moves a point on C inwards (by switching the point from L_{in} to L_{out}).

The external speed is synthesized from the image and there are many ways to define F_{ext}. For example, the function F_{ext} could be based on the response of an edge detector applied to the image; it could be based on a range of pixel intensities only, or as in the case of the Chan-Vese *Active Contours Without Edges*, the speed could depend on the mean intensities of the regions inside and outside the curve C.

In the second cycle, C is evolved according to the internal speed. The most common way to define F_{int}, is to apply a gaussian filter on the level-set function ϕ for each point on the curve C. The gaussian filter is a weighted sum of a neighborhood of ϕ, centered at a point on the curve C. If the majority of pixels in the neighborhood are inside C, switching a point from L_{out} to L_{in} has a smoothing effect on the curve. Otherwise, if the majority of pixels in the neighborhood are outside C, then switching a point from L_{in} to L_{out} smoothes the curve. This observation can be summarized into the following rule: a point is switched from L_{out} to L_{in} or vice versa, if the sign of ϕ for that point before the gaussian filter is applied to it, is different from the sign of ϕ after the gaussian filter is applied to it.

The evolution of the curve stops when one of two following conditions is satisfied: *(a)* The speeds at each neighboring grid point satisfy:

$$\begin{array}{ll} F(x) \leq 0, & \forall x \in L_{out} \\ F(x) \geq 0, & \forall x \in L_{in}; \end{array} \tag{4}$$

(b) a pre-defined maximum number of iterations N_a, is reached. The pre-defined maximum number of iterations needs to be specified for a noisy image, because the curve may fail to converge to the stable state specified in *(a)*.

This fast level-set framework has been used to solve image processing problems in real time. Before we show how this framework can be used to calculate and update the mean intensities inside and outside the evolving curve, we first review the Chan-Vese model in the next section.

3. CHAN-VESE ACTIVE CONTOUR MODEL

The advantage of the Chan-Vese *Active Contours Without Edges* model, is that it is able to segment an image that has smooth boundaries. It can do this because the evolution of the curve does not depend on gradient information, so weak edges do not affect the final segmentation. The Chan-Vese model suffers from initialization problems (Pan et al., 2006b). The final segmentation is dependent on the placement of the initial curve; sometimes this behavior is desirable

The original formulation of the *Active Contours Without Edges* model (Chan and Vese, 2001), focused on bi-modal images. This was later extended to multiphase images (Chan and Vese, 2002). In the bi-modal model, it is assumed that an image I consists of two regions, c_f and c_b, of approximately piecewise-constant distinct intensity values. If the region to be segmented is represented by c_f, then a curve C can be evolved to reach the boundary of c_f by minimizing the energy:

$$F_1(C) + F_2(C), \tag{5}$$

where F_1 and F_2 are defined as follows:

$$\begin{array}{rcl} F_1(C) & = & \displaystyle\int_{inside(C)} |I - c_1|^2 dxdy \\ & \text{and} & \\ F_2(C) & = & \displaystyle\int_{outside(C)} |I - c_2|^2 dxdy. \end{array} \tag{6}$$

C represents the curve, and the variables c_1 and c_2 represent the average intensities inside and outside the curve respectively. If the curve C is inside the region to be segmented, represented by intensity c_f, then $F_1(C) \approx 0$ and $F_2(C) > 0$. If the curve C is outside c_f, then $F_1(C) > 0$ and $F_2(C) \approx 0$. Only when the curve is on the boundary of the region of interest will $F_1 \approx 0$ and $F_2 \approx 0$.

To incorporate this energy minimization into the fast level-set framework, only the external speed function F_{ext} needs to be defined.

3.1 Definition of the External Speed Function

To incorporate the Chan-Vese model into the Shi-Karl level-set framework, the authors in (Lakshmanan et al., 2006) calculate the external speed with:

$$v_i = I(x,y) - c_i \quad i \in \{1,2\}, \tag{7}$$

where $I(x,y)$ is the pixel intensity of a point on the curve, c_1 is the average pixel intensity inside the curve and c_2 is the average pixel intensity outside the curve. The external speed F_{ext} is then defined as:

$$F_{ext} = \begin{cases} 1, & if \;\; v_1 \le 0 \;\; and \;\; v_2 > 0; \\ -1, & if \;\; v_1 > 0 \;\; and \;\; v_2 \le 0. \end{cases} \tag{8}$$

This definition of the external speed is not entirely correct. According to this definition, the curve will only expand outwards if the pixel intensity of a point on the curve is greater than the average pixel intensity of the area outside the curve. Figure 1 a) illustrates such an example and the rectangle is successfully segmented. However, by inverting the colors of the test image such that the average pixel intensity of the area outside the curve, is greater than the intensity of a point on the curve, the rectangle is not segmented because the curve fails to evolve outwards (see Figure 1 b)).

To solve this problem we define F_{ext} as:

$$F_{ext} = \begin{cases} 1, & if \;\; |v_1| < |v_2|; \\ -1, & if \;\; |v_1| > |v_2|, \end{cases} \tag{9}$$

where v_1 and v_2 are calculated with equation (7). In the next section we show how the average intensities c_1 and c_2 can be updated efficiently for each evolution iteration of the curve.

4. CALCULATING AND UPDATING THE MEAN PIXEL INTENSITIES

Previous work on fast implementations of the Chan-Vese model (Lakshmanan et al., 2006; Pan et al., 2006a), did not discuss how the mean pixel intensities, outside the curve and inside the curve, can be calculated. We show that the mean intensities c_1 and c_2 can be updated efficiently during the evolution of the curve, since each pixel on the curve in the Shi-Karl fast level-set framework, is moved outward or inwards sequentially, and the intensities of the pixels that are modified in the level-set function ϕ are known. In the traditional level-set framework, where partial differential equations are solved, the level-set function ϕ is evaluated over the entire image, or over a narrow band (Adalsteinsson and Sethian, 1995), and there is no direct knowledge of which pixels have moved outwards, which pixels have moved inwards and which pixels have remained stationary. As a consequence, there is no direct and obvious knowledge of how the mean intensities inside and outside the curve can be updated online. Hence, the mean intensities are calculated by iterating over the entire image, for each iteration of the curve evolution. The original paper on *Active Contours Without Edges* (Chan and Vese, 2001), presents the calculation of the mean intensities as such, and to our knowledge no faster way of calculating the mean intensities has been presented in previous works. In fact, even work that combines the fast Shi-Karl level-set method and the Chan-Vese *Active Contours Without Edges* for real-time contour tracking (Thida et al., 2006), still presents the calculation of the mean intensities inside and outside the curve, as the iteration over the entire image for each frame. For these reasons, we choose the calculation of the mean intensities inside and outside the curve C, by iterating over the entire image for each iteration of the curve evolution, as the baseline against which we compare our proposed method.

Our proposed calculation scheme is simple to implement. Given Ω the image domain, a value v, and Ω_v defined as follows:

$$\Omega_v = \{(x,y) \in \Omega \;/\; \phi(x,y) = v\}, \tag{10}$$

initial values of c_1 and c_2 can be obtained as follows:

$$c_1 = \frac{i_i(-3)}{tp_i(-3)} \tag{11}$$

and

$$c_2 = \frac{i_o(3)}{tp_o(3)}, \tag{12}$$

where for a certain value v, $i_i(v)$, $i_o(v)$, $tp_o(v)$ and $tp_i(v)$ are defined as follows:

$$i_i(v) = \int_{\Omega_v} I(x,y) H(\phi(x,y)) dxdy; \tag{13}$$

$$tp_i(v) = \int_{\Omega_v} H(\phi(x,y)) dxdy; \tag{14}$$

$$i_o(v) = \int_{\Omega_v} I(x,y)(1 - H(\phi(x,y))) dxdy; \tag{15}$$

$$tp_o(v) = \int_{\Omega_v} (1 - H(\phi(x,y))) dxdy; \tag{16}$$

and $H(x)$ is a step function defined as:

$$H(x) = \begin{cases} 1, & \text{if } x > 1; \\ 0, & \text{if } x < -1. \end{cases} \tag{17}$$

In summary, $tp_i(-3)$ is the total number of pixels for which $\phi = -3$; $tp_o(3)$ is the total number of pixels for which $\phi = 3$; $i_i(-3)$ is the sum of intensities of the pixels, for which $\phi = -3$ and $i_o(3)$ is the sum of the intensities of the pixels, for which $\phi = 3$.

Once the initial values for c_1 and c_2 have been calculated, they can be updated efficiently for the evolution of

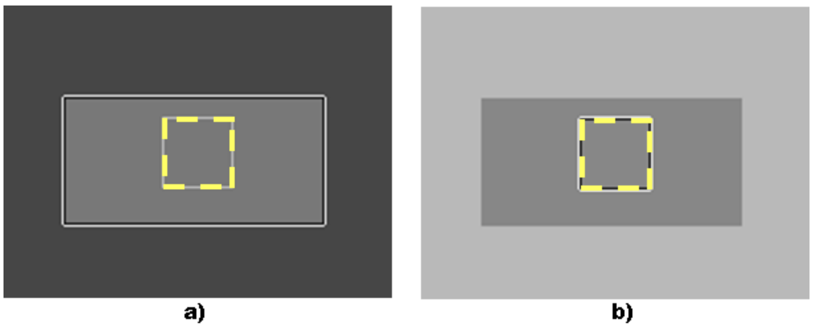


Figure 1: A Poorly Defined External Speed Function **a)** The contour successfully evolves and segments the rectangle. **b)** The contour fails to evolve at all. The initial placement of the curve is overlayed with dashed lines for clarity.

the curve, every time a pixel is switched from L_{in} to L_{out} or vice-versa, and anytime a pixel is removed from a list without placing it in the other list. With this in mind, we modified the two fundamental methods that govern the evolution of the curve in the Shi-Karl fast level-set framework: $switch\_in(x)$ and $switch\_out(x)$.

The purpose of the $switch\_in(x)$ procedure is to move a point on the curve outward by one pixel, while the $switch\_out(x)$ procedure moves a point on the curve inward by one pixel. Our modified procedure $switch\_in(x)$ for a point $x \in L_{Out}$ is defined as:

1. Delete x from L_{Out} and add it to L_{In}. Set $\phi(x) = -1$;

2. $\forall y \in N_4(x)$ satisfying $\phi(y) = 3$, add y to L_{Out} and set $\phi(y) = 1; tp_o \longrightarrow tp_o - 1$ and $i_o \longrightarrow i_o - y$,

while our modified procedure $switch\_out(x)$ for a point $x \in L_{In}$ is defined as:

1. Delete x from L_{In} and add it to L_{Out}. Set $\phi(x) = 1$;

2. $\forall y \in N_4(x)$ satisfying $\phi(y) = -3$, add y to L_{In} and set $\phi(y) = -1$; $tp_i \longrightarrow tp_i - 1$ and $i_i \longrightarrow i_i - y$,

where $N_4(x)$ is the discrete 4-connected neighborhood of x, and i_i, tp_i, i_o and tp_o are defined by equations (13)-(16) respectively.

By adding and subtracting from the variables i_i, tp_i, i_o and tp_o, we can keep track of the necessary information to recalculate the mean intensities c_1 and c_2, without having to iterate over the whole image I.

4.1 Evolving the Curve

Besides $switch\_in(x)$ and $switch\_out(x)$, two more methods $remove\_in(x)$ and $remove\_out(x)$ need to be defined, to explain in detail how the curve C is evolved. The procedure $remove\_in(x)$ is defined as:

1. if $\forall y \in N_4(x), \quad \phi(y) < 0$, delete x from L_{in} and set $\phi(x) = -3$;

2. $tp_i \longrightarrow tp_i + 1$ and $i_i \longrightarrow i_i + x$,

and the procedure $remove\_out(x)$ is defined as:

1. if $\forall y \in N_4(x), \quad \phi(y) > 0$, delete x from L_{out} and set $\phi(x) = 3$;

2. $tp_i \longrightarrow tp_o + 1$ and $i_o \longrightarrow i_o + x$

The purpose of these methods is to remove redundant points that may have been added to L_{in} or L_{out}, when the level-set function ϕ was modified with the $switch\_in(x)$ and $switch\_out(x)$ procedures.

An outline of the algorithm proposed by Shi and Karl, that evolves the curve C, is listed in Algorithm 1. The $\otimes$ symbol denotes convolution, and G is a gaussian kernel. The stopping condition is tested after the curve is evolved according to both the internal force and external force cycles. Some implementations check for the stopping condition immediately after the external force evolution cycle, and skip the internal force evolution cycle if the stopping condition is satisfied. In Section 5, we introduce an additional stopping condition to optimize the gaussian regularization cycle.

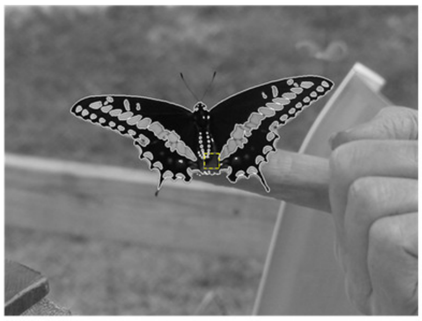
4.2 Experimental Results on the Fast Mean Intensity Calculation

We have reduced the time complexity of each iteration of our algorithm by $O(n)$, where n is the number of pixels in image I, by calculating c_1 and c_2 at each iteration based on updated values of i_i, tp_i, i_o and tp_o, instead of calculating c_1 and c_2 by iterating over the entire image I.

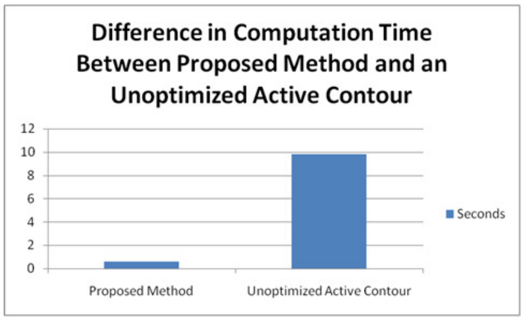
To demonstrate the optimization, we ran our algorithm on a large 1024×768 image taken from the Caltech database (Griffin et al., 2007), using an Intel Core 2, 6420 @ 2.13 GHZ with 2 Gigabytes of RAM, and compared it to the algorithm that calculates c_1 and c_2 by iterating over the whole image. We evolved our curve according to the external force F_{ext} only. Refer to Figure 2.

5. OPTIMIZING THE REGULARIZATION METHOD

By evolving a curve according to the external speed F_{ext} only, the curve often develops sharp boundaries due to



a)



b)

Figure 2: Comparison of Mean Intensity Calculation Methods in Time **a)** The dashed square represents the initial curve boundary. **b)** Difference in computation time between our proposed method, and an implementation in which the mean pixel intensities outside and inside the curve, are calculated by iterating over the entire image for each iteration of the curve evolution. No regularization was used.

Algorithm 1 Algorithm for Curve Evolution in the Shi-Karl Framework

```
1: Initialize the array $\phi$, $F$, and the two lists $L_{in}$ and $L_{out}$.
2: repeat
3:   for $i = 0$ to $n_{ext}$ do {Evolve according to external speed for $n_{ext}$ iterations}
4:     For each point $x \in L_{out}$, $switch\_in(x)$, if $F_{ext} > 0$.
5:     For each point $x \in L_{in}$, $remove\_in(x)$.
6:     For each point $x \in L_{in}$, $switch\_out(x)$, if $F_{ext} < 0$.
7:     For each point $x \in L_{out}$, $remove\_out(x)$.
8:   end for
9:   for $i = 0$ to $n_{int}$ do {Evolve according to internal speed (smoothness) for $n_{int}$ iterations}
10:    For each point $x \in L_{out}$, $switch\_in(x)$, if $(G \otimes \phi)(x) < 0$.
11:    For each point $x \in L_{in}$, $remove\_in(x)$.
12:    For each point $x \in L_{in}$, $switch\_out(x)$, if $(G \otimes \phi)(x) > 0$.
13:    For each point $x \in L_{out}$, $remove\_out(x)$.
14:  end for
15: until equation (4) holds, or $N_a$ iterations have elapsed.
```

noise. To smooth the curve and to make it less susceptible to noise, an internal speed F_{int} is usually introduced into the model. In the traditional curve evolution methods which are based on partial differential equations, the internal speed is some regularization parameter or function, that is introduced into an energy minimization framework. However, in the Shi-Karl level-set framework that does not solve partial differential equations, the most common approach to smooth the curve is to perform a gaussian filtering on the level-set function ϕ. We summarized the gaussian filter method in Section 2. In this Section we discuss a further optimization to this method and introduce an additional stopping criterion, after demonstrating that performing a gaussian filtering on the level-set function may in some cases prevent the active contour from terminating.

5.1 Gaussian Filtering and Idempotent Active Contours Without Edges

On certain images, the choices of the number of iterations for the external force cycle F_{ext}, and the internal force cycle F_{int}, can cause idempotents. For example, the curve C could be caught in a cycle where it expands outwards due to the external force, and shrinks back to its original shape because of the internal force. This usually happens when the curve has almost reached its optimum boundary, when evolving with the external force F_{ext}, and most pixels are stationary. Refer to Figure 3 for an example of an active contour trapped in such a cycle.

Related work that uses the Shi-Karl fast level-set framework (Shi and Karl, 2005b) and (Thida et al., 2006), presents the stopping criteria a) and b) in equation (4), whereby the parameter N_a is used to ensure the termination of the algorithm for noisy images.

Specifying a pre-determined maximum number of iterations N_a will ensure the termination of the algorithm, but the choice of an appropriate value is difficult. If the chosen value is too low, it can result in a premature termination, and if it is too high it will increase the computation time unnecessarily.

In (Shi and Karl, 2005b), the authors state that when noise in an image is low, one can choose a small value for n_{int} , or increase the parameter n_{ext}, to reduce the percentage of computation allocated for smoothness regularization, thereby speeding up the algorithm (refer to Algorithm 1). This may be true for certain images, but Figure 3 clearly demonstrates that an evolving curve can be trapped in a cycle, even when there is no noise in the image.

We solve this problem by testing for idempotents and stopping the evolution of the curve when a cycle is detected.

After n iterations of evolving the curve according to *both* the external and internal force cycles, the points contained in the outer list L_{out} are defined as $\overline{x}_n$. By evolving the curve C according to the external force (represented by function f; $f^2 = f \circ f$ is the composition of f by itself, and $f^n = \underbrace{f \circ f \circ \ldots \circ f}_{n\ times}$), we have

$$f^{n_{ext}}(\overline{x}_n) = \overline{x}_{n_{ext}}, \tag{18}$$

where $\overline{x}_{n_{ext}}$ are the points contained in L_{out}, after the external force evolution cycle. After evolving according to the internal force (represented by function $\hat{f}$), if we have

$$\hat{f}^{n_{int}}(\overline{x}_{n_{ext}}) = \overline{x}_n, \tag{19}$$

then a cycle has been detected.

Unfortunately, this type of cycle detection involves comparing each element in $\overline{x}_n$ with each element in $\overline{x}_{n-1}$, which is time consuming especially if the lists are not sorted. To avoid making these comparisons, we instead test to see if the number of points in $\overline{x}_n$ is approximately the same as the number of points in $\overline{x}_{n-1}$. In other words, we detect a cycle if

$$|\overline{x}_n| = |\overline{x}_{n-1}| - \epsilon, \text{where } \epsilon \in \mathbb{Z}. \tag{20}$$

We choose ϵ to be a small integer, usually $\epsilon \in \{-2, -1, 0, 1, 2\}$. This is necessary because sometimes the difference between the two lists is only one or two pixels, which have no real impact on the final segmentation; without the ϵ a lot of computation is wasted on insignificant pixels.

5.2 Removing Redundant Cycles During Regularization

We have already mentioned that the number of iterations for the external speed evolution cycle and the internal speed evolution cycle, have to be chosen empirically. Sometimes the number of iterations for the internal speed evolution cycle (regularization) can be set too high, resulting in wasted computation cycles. When the curve C is evolved according to the internal force (represented by function $\hat{f}$), there may be a value of k such that

$$\hat{f}^{n_{int}-k}(\overline{x}_{n_{ext}}) \quad = \quad \overline{x}_n \tag{21}$$

and

$$\hat{f}^{n_{int}-k+1}(\overline{x}_{n_{ext}}) \quad = \quad \overline{x}_n, \tag{22}$$

where $k \in \mathbb{N}$ and n_{int} is the empirically chosen number of iterations for the internal speed evolution cycle. Using equations (21) and (22), we can exit the regularization cycle when we detect that a cycle has not changed $\overline{x}_n$, and avoid unnecessary computations. In practice, to sidestep comparing each element in $\overline{x}_n$ with each element in $\overline{x}_{n-1}$, we instead exit the regularization cycle when equation (20) is true for $\epsilon = 0$.

5.3 Choosing Parameters

In (Shi and Karl, 2005a), the authors mention that the choice of the parameter n_{int}, should normally be set to equal the size N_k of a gaussian kernel. The size of the gaussian kernel (N_k), is geometrically related to the elimination of small holes in the final segmentation; to eliminate holes with a radius smaller than r, $N_k = 2r$. However, in our experiments we found that when n_{int} is greater than the size of the gaussian kernel, segmentation results can sometimes be affected in a positive way (refer to Figure 4 for an example). Hence, we prefer to choose $n_{int} \approx 3N_k$, and allow our algorithm to remove redundant regularization cycles. In this way, the amount of smoothing can vary for each iteration.

The number of iterations devoted to external speed evolution (n_{ext}), is usually greater than n_{int}, since it attracts the curve to the regions of interest.

Finally, the choice of ϵ for the detection of idempotents, is related to the size of the objects in the scene that are to be segmented. For example, if one of the objects is a needle, then it is possible that from one complete curve evolution cycle to the next (using both the external and internal force), the curve expands by only one pixel and so we should choose $\epsilon > 1$. If the objects are large, than it is unlikely that the curve should expand by only one pixel after one complete curve evolution cycle, and so we set $\epsilon \leq 1$.

5.4 Experimental Results on the New Stopping and Regularization Criteria

Figure 5 shows a comparison in running time of an active contour on a test image, with and without our new criteria. The test was conducted on a 640×480 image taken from the Caltech database (Griffin et al., 2007), using an Intel Core 2, 6420 @ 2.13 GHZ with 2 Gigabytes of RAM. The running time of our algorithm was considerably less, because it removed redundant regularization cycles and detected idempotents. The same method of calculating the mean pixel intensities was used throughout the experiment.

In Figure 6 we compare the running time on a large 1600×1600 image of Mars, also taken from the Caltech database. Once again the running time of our algorithm is less. In this image, the difference between the running time is not so great, because the active contour was not trapped in an idempotent cycle, and only 5 iterations were devoted to regularization per evolution cycle. This means that there are at most only 5 redundant regularization cycles, per evolution cycle. By increasing the amount of iterations devoted to regularization, the difference in computation time becomes more noticeable. Nonetheless, even fractions of a second are important in real-time image processing.

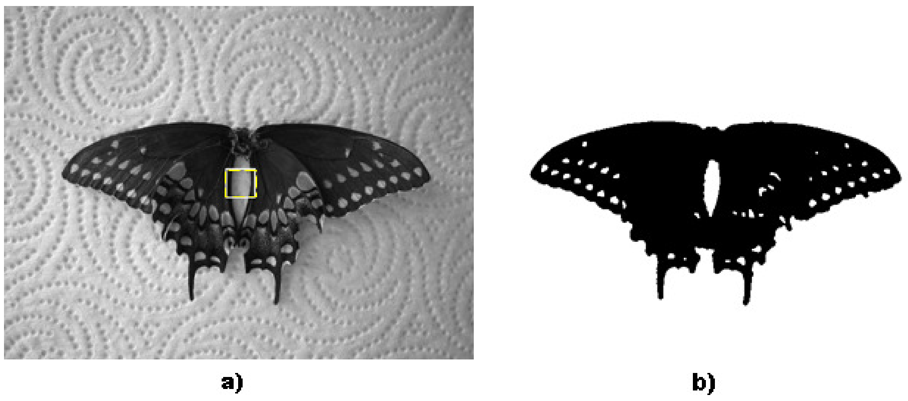
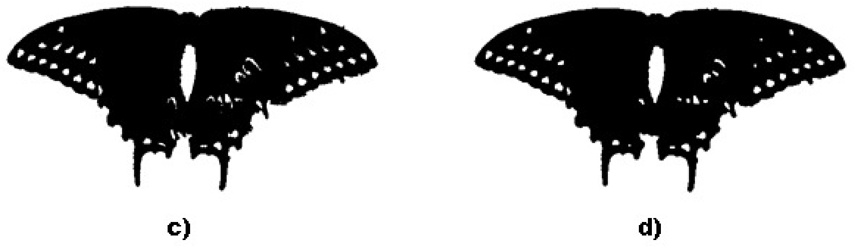



Figure 3: Example of an Active Contour Trapped in an Idempotent Cycle **a)** The initial contour (0 Iterations). **b)** Region inside the curve after 502 iterations. **c)** Region inside the curve after the external speed evolution cycle (522 iterations). **d)** Region inside the curve after the regularization cycle (532 iterations). Notice that **b)** and **d)** are exactly the same. This cycle could continue indefinitely. For example, iteration 562 will be the same as **b)** and **d)**.

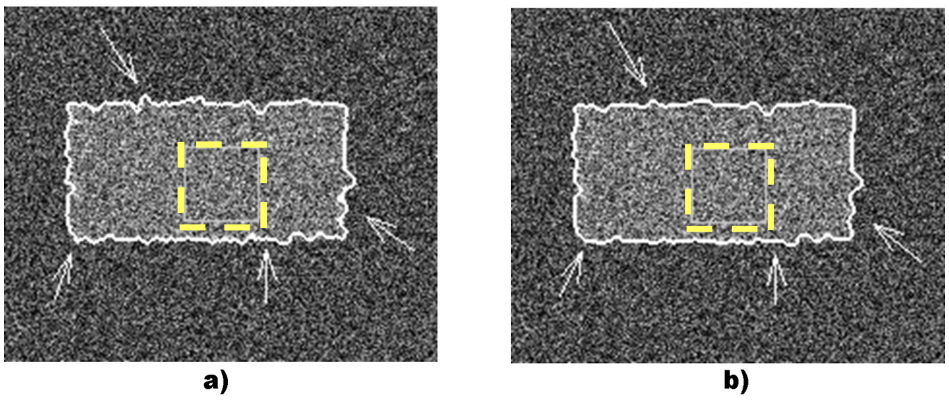


Figure 4: Difference in Regularization Quality **a)** Regularization, by choosing n_{int} as suggested by Shi and Karl (n_{int} equals the size of the kernel). $n_{ext} = 12$, $n_{int} = 5$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization. **b)** Regularization, by choosing a large value for n_{int}, and allowing our algorithm to remove redundant regularization cycles. $n_{ext} = 12$, $n_{int} = 10$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization. The arrows point to parts of the curve that are smoother. The initial placement of the curve is overlayed with dashed lines for clarity.

6. EXPERIMENTAL RESULTS ON THE COMBINATION OF OUR PROPOSED ALGORITHMS

In Figure 7, we compare both the quality of segmentation and the running time, between an active contour using a standard regularization implementation and using the baseline method of calculating the mean intensities inside and outside the curve, against an active contour using our proposed regularization method and our fast mean intensity calculation scheme. We use a synthetic image of size 512×512, so that a segmentation can be unambiguously evaluated. Clearly, the segmentation result is the same for both methods, while the running time of our method is less.

7. CONCLUSION

We have presented several optimizations for the Chan-Vese *Active Contours Without Edges*, when the active contours are implemented without solving partial differential equations. Using the Shi-Karl framework, we have introduced a fast scheme for updating the mean pixel intensity inside and outside the evolving curve, and we have explained why we chose the baseline for our comparison as the calculation of the mean intensity, by iterating over the entire image for each iteration. Additionally, we have shown that the choice of parameters for the external speed evolution loop and the regularization loop, can trap the active contour in an idempotent cycle, and we have developed a new stopping criterion to break out of that cycle. Finally, we have optimized the regularization loop by exiting the loop, when a further execution of the regularization loop has no impact on the active contour.

REFERENCES

Adalsteinsson, D. and Sethian, J. (1995). "A Fast Level Set Method for Propagating Interfaces". Journal of Computational Physics, No. 118(2), 269-277.

Chan, T. and Vese, L. (2001). "Active Contours Without Edges". IEEE Trans. Image Processing, No.14(10) (Feb.), 266–277.

Chan, T. and Vese, L. (2002). "A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model". International Journal of Computer Vision, No.50(3), 271–293.

Griffin, G.; Holub, A. and Perona, P. (2007). "Caltech-256 Object Category Dataset". California Institute of Technology, 7694, http://authors.library.caltech.edu/7694

Kass, M.; Witkin, A.; and Terzopoulos, D. (1987). "Snakes - Active Contour Models". International Journal of Computer Vision No. 1(4), 321-331.

Lakshmanan, A.; Thida, M.; Chan, K. L. and Zhou, J. (2006). "Incorporation of Active Contour Without Edges in the Fast Level Set Framework for Biomedical Image Segmentation". In *International Conference on Biomedical and Pharmaceutical Engineering* (Singapore, Dec.), 296–300.

Pan, Y.; Birdwell, D. J. and Seddik D. M.(2006). "Efficient Implementation of the Chan-Vese Models Without Solving PDEs". In *Proceedings of International Workshop On Multimedia Signal Processing* (Victoria, BC, Canada, Oct. 03-06), 350–353.

Pan, Y.; Birdwell, D. J.; and Seddik, D. M. (2006). "An Efficient Bottom-up Image Segmentation Method Based on Region Growing, Region Competition and the Mumford Shah Functional". In *Proceedings of International Workshop On Multimedia Signal Processing* (Victoria, BC, Canada, Oct.), 344–348.

Paragios, N. and Deriche, R. (2000). "Coupled Geodesic Active Regions for Image Segmentation: A Level Set Approach". In *Proceedings of ECCV* (Dublin), 224-240.

Sethian, J. (1999). *Level Set Methods and Fast Marching Methods*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press

Shi, Y. and Karl, W. (2005). "A Fast Level Set Method Without Solving Pdes. In *Proceedings of ICASSP* (Philadelphia, PA, USA, Mar.), 97–100.

Shi, Y. and Karl, W. (2005) "Real-time Tracking Using Level-sets". In *Proceedings of CVPR* (San Diego, CA, USA, June) Vol. 2, 34–41.

Thida, M.; Chan, K. L. and Eng, H. L (2006) "An Improved Real-time Contour Tracking Algorithm using Fast Level Set Method". In *Proceedings of the First Pacific Rim Symposium* (Hsinchu, Taiwan, Dec.), 702–711.

AUTHOR BIOGRAPHIES

ZYGMUNT L. SZPAK is a Master's student at the School of Computer Science at the University of KwaZulu-Natal, South Africa. His general research interests include Artificial Intelligence, Image Processing, Computer Vision and Pattern Recognition. Currently, the central theme of his research is on real-time tracking and modelling of the behavior of ships, in a maritime environment. His email is `zygmunt.szpak@gmail.com`.

JULES R. TAPAMO is Associate Professor at the School of Computer Science at the University of KwaZulu-Natal, South Africa. He completed his PhD degree from the University of Rouen (France) in 1992. His research interests are in Image Processing, Computer Vision, Machine Learning, Algorithms and Biometrics. He is a member of the IEEE Computer Society, IEEE Signal Processing Society and the ACM. He maintains a Computer Vision, Image Processing and Data Mining research webpage at `http://www.cs.ukzn.ac.za/cvdm`. His email is `tapamoj@ukzn.ac.za`.

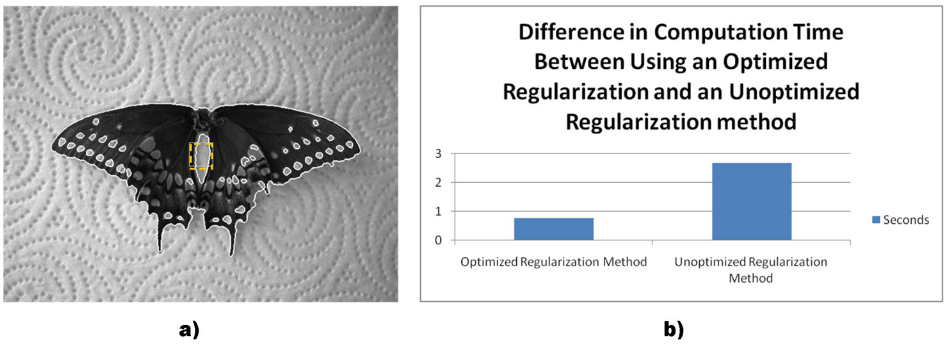


Figure 5: Comparison of Regularization Methods in Time **a)** The dashed square represents the initial curve boundary. **b)** Difference in computation time between our proposed optimized regularization method, and a standard regularization implementation that does not remove redundant regularization cycles, and that does not check for idempotents. $n_{ext} = 10$, $n_{int} = 5$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization.

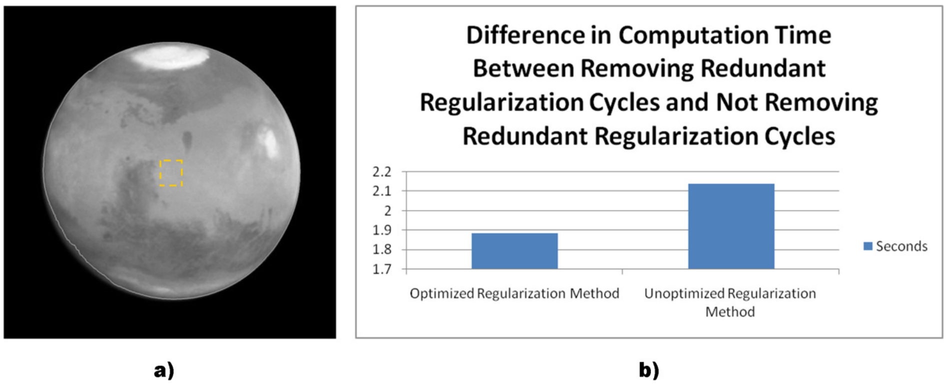


Figure 6: Comparison of Regularization Methods in Time **a)** The dashed square represents the initial curve boundary. **b)** Difference in computation time between our proposed optimized regularization method, and a standard regularization implementation that does not remove redundant regularization cycles, and that does not check for idempotents. $n_{ext} = 10$, $n_{int} = 5$ and $N_a = 2000$. A 5×5 discrete gaussian kernel was used for regularization.

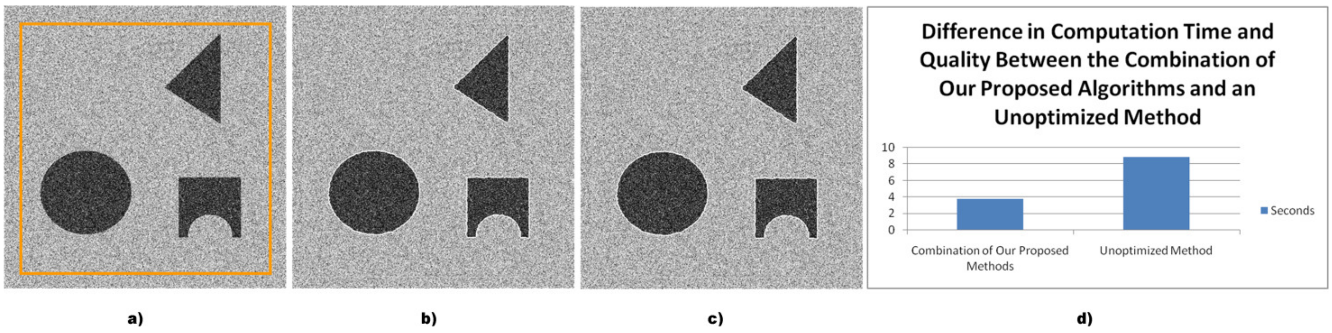


Figure 7: Comparison of the Combination of our Proposed Optimizations against an Unoptimized Method in Time and Quality **a)** Synthetic image corrupted by gaussian noise, with $\mu = 0$ and $\sigma = 80$. The square represents the initial curve boundary. **b)** Segmentation result of our proposed regularization methods, together with our fast mean intensity calculation scheme. **c)** Segmentation result by calculating the mean intensities inside and outside the curve, for each iteration, and using a standard regularization implementation that does not remove redundant cycles nor check for idempotents. **d)** Difference in computation time between the combination of our proposed methods (b), and an approach that calculates the mean intensities inside and outside the curve, for each iteration, and uses a standard (Shi-Karl) regularization implementation, that does not remove redundant regularization cycles, and that does not check for idempotents (c). $n_{ext} = 10$, $n_{int} = 5$ and $N_a = 1000$. A 5×5 discrete gaussian kernel was used for regularization.

Workshop on Optimization Issues in Grid and Parallel Computing Environments (Optim’08)

ENERGY MINIMIZATION-BASED CROSS-LAYER DESIGN IN WIRELESS NETWORKS

Le Thi Hoai An, Nguyen Quang Thuan
Laboratory of Theoretical and Applied Computer Science,
Paul Verlaine Metz University, Metz, FRANCE,
Email: lethi, thuan@univ-metz.fr.

Phan Tran Khoa
Department of Electrical and Computer Engineering,
University of Alberta, Edmonton, AB, CANADA,
Email: khoa@ece.ualberta.ca.

Pham Dinh Tao
Laboratory of Modelling, Optimization & Operations Research,
National Institute of Applied Sciences, Rouen, FRANCE,
Email: pham@insa-rouen.fr.

KEYWORDS

Cross-layer optimization, TDMA, Difference of Convex functions Algorithm (DCA)

ABSTRACT

In this paper, a cross-layer optimization framework is proposed for multi-hop time division multiple access (TDMA) networks. Particularly, given a set of quality-of-service (QoS) constraints on the network flows, we study a centralized controller that coordinates the routing process, link scheduling and power control to minimize the energy consumption in the network. The aforementioned design can be formulated as a mixed integer-linear program (MILP) in which finding optimal solution is well-known to have worst case exponential complexity. Realizing this inherent difficulty in computational complexity, our main contribution is to propose a novel approach to solve the cross-layer design problem which is based on a so-called *Difference of Convex functions Algorithm* (DCA). The proposed approach is able to provide either optimal or near-optimal solutions with finite convergence. The preliminary numerical results demonstrate the effectiveness of the proposed design.

INTRODUCTION

Wireless networks, for example mesh, ad hoc or sensor networks have recently emerged as essential means of communications to provide reliable data communication among many users. In such networks, wireless nodes usually self-configure to exchange information without the aid of any established infrastructure. However, due to the random deployment and mobility of wireless nodes, multi-hop transmission is necessary where nodes can forward other nodes' information. Due to interference between links, in this research, time division multiple access-based (TDMA) MAC is adopted to allocate communication resources to links/nodes. Note that the problem of optimal scheduling in TDMA-based networks is NP-complete (5) and is somehow similar to the vertex coloring problem in graph theory (13). Furthermore, in a multi-hop network, power allocation, link scheduling, routing, and rate control interact with each other. Thus, a cross-layer design across all layers (see, e.g., (3) for an overview) is shown to outperform the method of designing each layer by itself which is popular in wireline networks. Recently, cross-layer optimization with different design objectives and constraints has received much attention from the academia (2), (4), (15), (16).

In this work, we consider a cross-layer design problem to allocate communication resources, i.e., time and power to links in an interference-limited TDMA wireless network. Generally, nodes in a wireless network are battery-powered devices and energy is consumed when a node transmit or receive data to/from other nodes. Moreover, since nodes participate in the network operation by either generating or relaying information that needs to be communicated to a base station, we aim at minimizing the energy consumption for all nodes. The proposed design objective is helpful to estimate the energy expenditure for optimal network operation. We show that the proposed design can be formulated as a mixed integer-linear program (MILP) which is well-known to be computationally expensive. By employing the *exact penalty method* theory, we are able equivalently recast the proposed MILP as a *concave minimization* problem with only continuous variables without losing optimality. Next, we reformulate the concave minimization problem in the form of a DC (Difference of Convex functions) program that consists of minimizing a DC function on the whole space. We propose a technique which combines *DC Algorithm* (DCA) and the traditional *branch and bound* (BnB) to solve the resulting DC problem. Generally, DCA has linear convergence and achieves near-optimal solution. One of the powerful and distinct advantage of the DCA-BnB approach is its ability to solve very large-scale problems.

SYSTEMS DESCRIPTION

Consider a multi-hop network with node set $\mathcal{N}$. Uplink transmission is assumed where there is one common traffic destination (not included in $\mathcal{N}$) for all the nodes. Each node $n \in \mathcal{N}$ generates traffic at a rate r_n which is a integer number of unit rate. Let $\mathcal{L}$ denote the set of unidirectional links.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

TDMA-based MAC and Flow Conservation Model

In a multi-hop network, in general, all the links may not be scheduled to transmit concurrently since they contend and/or interfere with each other. In addition, due to primary interference, each node cannot transmit and receive simultaneously, and thus, a node's outgoing and incoming links cannot be active at the same time. Further, we assume unicast network in which a transmitter cannot transmit data to more than one receivers. In addition, any two simultaneous transmissions with a common receiver are not allowed due to collision in packet reception.

In the considered TDMA network, time is partitioned into fixed-length frames, and each frame is further divided into J time slots with unit duration. Since the resource allocation is the same in all frames, we concentrate our design on a single frame. A node may need to transmit in one or more slots for its own traffic and/or relay traffic from other nodes. If a node transmits in a slot, while its transmission power can be varied from [0, $P_{\max}$], its transmission rate is fixed at a unit rate. In the TDMA-based network, a channel is specified by two elements (j, l), $j \in \mathcal{J}$, $l \in \mathcal{L}$, where $\mathcal{J} = \{1, 2, ..., J\}$. For the channel, the resource allocation is denoted by (s_j^l, P_j^l), where $s_j^l = 1$ means link l is active at slot j while $s_j^l = 0$ otherwise, and $P_j^l > 0$ denotes the transmission power of link l at slot j if $s_j^l = 1$, $P_j^l = 0$ otherwise.

At each node, the difference of its outgoing traffic and its incoming traffic should be the traffic generated by itself, i.e.,

$$\sum_{l \in \mathcal{O}(n)} \sum_{j=1}^{J} s_j^l - \sum_{l \in \mathcal{I}(n)} \sum_{j=1}^{J} s_j^l = r_n, \quad n \in \mathcal{N} \tag{1}$$

where $\mathcal{O}(n)$ and $\mathcal{I}(n)$ are the set of outgoing links and incoming links at node n, respectively. The values of s_n for the non-source nodes are set to zero.

The energy consumption at node n can be written as

$$\mathcal{E}_n = \sum_{l \in \mathcal{O}(n)} \sum_{j=1}^{J} P_j^l + \sum_{l \in \mathcal{O}(n)} \sum_{j=1}^{J} \epsilon_l s_j^l + \sum_{l \in \mathcal{I}(n)} \sum_{j=1}^{J} \varepsilon_l s_j^l \tag{2}$$

where ϵ_l, ε_l denote the energy needed to transmit, receive a unit of traffic over link l, respectively. Note that ϵ_l, ε_l include the energy consumed by the signal processing blocks at the link ends.

Interference Model

Interference relations among the nodes and/or links in a wireless networks can be modeled in various ways, for example by using contention-based model (15) or the signal-to-interference-plus-noise-ratio (SINR)-based model (11), (1). The latter model is adopted in this research. Specifically, if the link $l \in \mathcal{L}$ is active at slot j (i.e., $s_j^l = 1$), the following inequality should hold so as to guarantee the transmission quality of the link

$$\mathrm{SINR}_j^l = \frac{P_j^l h_{ll}}{\sum_{k \neq l} P_j^k h_{kl} + \eta_l} \geq \gamma^{\mathrm{th}} \tag{3}$$

where SINR_j^l is the SINR for link l at slot j, h_{kl} is the path gain from the transmitter of link k to the receiver of link l, η_l is the noise power at receiver of link l, and γ^{th} is the required SINR threshold for accurate information transmission.

We assume that all wireless nodes are low-mobility devices and/or the topology of the network is static or changes slowly allowing enough time for computing the new scheduler. An example of such networks is a wireless sensor network for environmental monitoring with fixed sensor locations. In this case, the need for distributed implementation is not necessary.

PROBLEM FORMULATION

As discussed above, energy consumption is an important design criterion for a multi-hop wireless network. From the preceding discussions, the energy minimization-based cross-layer design, i.e., joint rate control, routing, link scheduling, and power allocation problem can be mathematically posed as

$$\min_{r_n, P_j^l, s_j^l} \quad \sum_{n \in \mathcal{N}} \mathcal{E}_n \tag{4a}$$

subject to:

$$\sum_{l \in \mathcal{O}(n)} \sum_{j=1}^{J} s_j^l - \sum_{l \in \mathcal{I}(n)} \sum_{j=1}^{J} s_j^l = r_n, \ n \in \mathcal{N} \tag{4b}$$

$$r_n \geq r_n^{\min}, \ n \in \mathcal{N} \tag{4c}$$

$$\sum_{l \in \mathcal{I}(\hat{n})} \sum_{j=1}^{J} s_j^l = \sum_{n \in \mathcal{N}} r_n \tag{4d}$$

$$\sum_{l \in \mathcal{O}(n)} s_j^l + \sum_{l \in \mathcal{I}(n)} s_j^l \leq 1, \ \forall n \in \{\mathcal{N} \cup \hat{n}\}, \ \forall j \tag{4e}$$

$$h_{ll} P_j^l \geq \gamma^{\mathrm{th}} \sum_{k \neq l} P_j^k h_{kl} + \gamma^{\mathrm{th}} \eta_l + D(s_j^l - 1), \quad \forall l \in \mathcal{L}, \ j = 1, \ldots, J \tag{4f}$$

$$0 \leq P_j^l \leq P_{max} s_j^l, \ \forall l \in \mathcal{L}, \ j = 1, \ldots, J \tag{4g}$$

$$s_j^l \in \{0, 1\}, \quad \forall l \in \mathcal{L}, \ j = 1, \ldots, J \tag{4h}$$

where $\hat{n}$ denotes the common sink node for all data generated in the network, D is a very large positive constant. The objective function is the energy consumption in the network. Constraints (4b) ensure that the data generated by source nodes are routed properly. Constraints (4c) guarantee that the rate for each node is no less than a minimum rate. The minimum rates are possibly different for nodes and are usually determined by the network QoS. Nodes which do not generate traffic have $r_n = r_n^{\min} = 0$. Constraint (4d) is the flow conservation at the traffic destination for all the sources. Constraints (4e) state that a node can not receive and transmit simultaneously in one particular time slot. Constraints (4f) make sure the SINR requirement is met: if a link l is active in time slot j, then the SINR at receiver of link l must be larger than the given threshold γ^{th} which also depends on the system implementation. Constraint (4f) is automatically satisfied if link l is not scheduled in time slot j. Constraint (4g) states that if a link l is scheduled for time slot j, i.e., $s_j^l = 1$, then the corresponding power value P_j^l must be less than P_{max}. Otherwise, P_j^l obviously equals to zero. We also impose binary integer constraints on s_j^l.

It can be seen that the cross-layer optimization problem (4a)–(4h) belongs to a class of well-known mixed-integer linear programs (MILPs). The combinatorial nature of the optimization (4a)–(4h) is not surprising and it has been shown in some previous works, albeit with different objective functions and formulations (11), (13), (1). Theoretically, MILPs are NP-hard which is clearly inviable for practical scenarios when the dimension is large. The following theorem is in order.

LEMMA 1: At optimality, the source rate constraints (4c) must be met with equalities for all sources.

PROOF: It is clear that at one node, the transmit power is an increasing function with respect to the node's transmission rate. Therefore, in order to minimize the transmit power, nodes should transmit at their minimum rate requirements or only relay data for other nodes. □

Since the proposed design aims at minimizing the total energy consumption, it may cause some particular nodes spending more energy than the other nodes, and thus, running out of energy quicker. Therefore, equal energy distribution among nodes is not optimal. In this context, the proposed design can be performed, for example during the stage of network planning. In such scenarios, the network designer needs to assign each wireless node a certain amount of energy (e.g., a number of AAA batteries) according to the network topology and QoS constraints of the nodes. Therefore, the proposed design helps to determine which nodes need to be equipped with more and/or less energy than the others. Moreover, it quantifies the minimum amount of energy needed in a TDMA frame to satisfy the QoS demands. Obviously, depending on a particular context, this energy value is closely related to the network lifetime depending how the network lifetime is defined.

As discussed, the routing algorithm resulted from the proposed design may cause some nodes spending more time than the others. Therefore, another design objective which may help to prevent such situation is as follows

$$\min_{r_n, P_j^l,\, s_j^l} \quad \max_{n\in\mathcal{N}} \mathcal{E}_n \tag{5a}$$

$$\text{subject to:} \quad \text{The constraints (4b)–(4h)}\,. \tag{5b}$$

The optimization problem (5a)–(5b) aims at minimizing the maximum energy consumed at nodes(s). As a result, more nodes are likely to be involved in the routing algorithm, i.e., relaying information for other nodes. Hereafter, for simplicity, we only consider the optimization problem (4a)–(4h).

The cross-layer optimization problem (4a)–(4h) has worst case exponential complexity when BnB methods are used to compute the solution. Moreover, when modeling practical networks and depending on the number of links, nodes and time slots, problem with large sizes may arise. As a result, it is extremely difficult to schedule links optimally. Most research in literature is based on heuristic at the cost of performance degradation, for example, see (11), (13). Here, we propose a method to solve the mixed 0-1 linear program (4a)–(4h) efficiently. To this purpose, we first apply the theory of exact penalization in DC programming (7) to reformulate the MILP as that of minimizing a DC function over a polyhedral convex set. The resulting problem is then handled by DCA which was introduced and extensively developed over the last decades (6), (8), (9), (10). The mentioned approach has been applied successfully in several large scale problems (see (6), (8), (9), (10) and reference therein). The details are provided in the following section.

AN EFFICIENT ALGORITHM FOR CROSS-LAYER DESIGN IN TDMA NETWORKS

DC Reformulation via Exact Penalty Method

Using an exact penalty result, we can reformulate the aforementioned MILP (4a)–(4h) in the form of a concave minimization program. The exact penalty technique aims at transforming the original MILP into a more tractable equivalent problem in the DC optimization framework. Let S be the feasible set of the problem MILP (4a)–(4h) which does not include the binary constraints. For notational simplicity, we group all the power variables and link scheduling variables in column vectors $P = [P_1^1 \dots P_1^J P_2^1 \dots P_L^J]^T$, $s = [s_1^1 \dots s_1^J s_2^1 \dots s_L^J]^T$ respectively where T denotes the transpose operator. We denote a new set $K := \{(P,s) \in S : s \in [0,1]^{LJ}\}$, and assume that K is a nonempty, bounded polyhedral convex set in $\mathbb{R}^{LJ} \times \mathbb{R}^{LJ}$. The cross-layer optimization problem (4a)–(4h) can be expressed in the general form

$$(P_{\text{opt}}, s_{\text{opt}}) \quad = \quad \arg\min \Big\{ e^T P + \eta^T s : (P,s) \in S, \; s \in \{0,1\}^{LJ} \Big\}. \tag{6}$$

where e is the column vector with all elements being 1, $\eta = [\eta_1^1, \dots, \eta_1^J, \eta_2^1, \dots, \eta_L^J]$, $\eta_l^j = \epsilon_l^j + \varepsilon_l^j$. Let us consider the function $p(P,s)$ defined by

$$p(P,s) = \sum_{l\in\mathcal{L},\, j\in\mathcal{J}} \min\{s_j^l, 1 - s_j^l\}. \tag{7}$$

It is clear that p is concave and finite on K, $p(P,s) \geq 0$ for all $(P,s) \in K$, and

$$\Big\{(P,s) \in S : s \in \{0,1\}^{LJ}\Big\} = \Big\{(P,s) \in K : p \leq 0\Big\}.$$

Hence problem (6) can be rewritten as

$$(P_{\text{opt}}, s_{\text{opt}}) \quad = \quad \arg\min \Big\{ e^T P + \eta^T s : (P,s) \in K, \; p(P,s) \leq 0 \Big\}. \tag{8}$$

The following theorem is in order.

THEOREM 2: (Theorem 1, (7)) Let K be a nonempty bounded polyhedral convex set, f be a finite concave function on K and p be a finite nonnegative concave function on K. Then there exists $\tilde{t}_0 \geq 0$ such that for $\tilde{t} > \tilde{t}_0$ the following problems have the same optimal value and the same solution set

$$(P_t) \qquad \alpha(t) = \min\Big\{f(x) + \tilde{t}p(x) : x \in K\Big\} \quad (9)$$

$$(P) \qquad \alpha = \min\Big\{f(x) : x \in K, p(x) \leq 0\Big\}. \quad (10)$$

Furthermore

- If the vertex set of K, denoted by $V(K)$, is contained in $x \in K : p(x) \leq 0$, then $\tilde{t}_0 = 0$.
- If $p(x) > 0$ for some x in $V(K)$, then $\tilde{t}_0 = \min\Big\{\frac{f(x)-\alpha(0)}{S_0} : x \in K, p(x) \leq 0\Big\}$, where $S_0 = \min\Big\{p(x) : x \in V(K), p(x) > 0\Big\} > 0$.

PROOF: The proof for the general case can be found in (7). □

From Theorem 2 we get, for a sufficiently large number $\tilde{t}$ ($\tilde{t} > \tilde{t}_0$), the equivalent concave minimization problem to (8)

$$\min : \Big\{e^T P + \eta^T s + \tilde{t}p(P,s) : (P,s) \in K\Big\} \qquad (11)$$

which is a DC program

$$\min : \Big\{g(P,s) - h(P,s)\Big\} \qquad (12)$$

where

$$\begin{aligned} g(P,s) &= \mathcal{X}_K(P,s) \\ h(P,s) &= -e^T P - \eta^T s - \tilde{t} \sum_{l \in \mathcal{L},\, j \in \mathcal{J}} \min\{s_j^l, 1 - s_j^l\} \end{aligned}$$

and $\mathcal{X}_K(P,s)$ is 0 if $(P,s) \in K$, otherwise $+\infty$ (the indicator function of K).

We have successfully transform an optimization with integer variables into its equivalent form with continuous variables.

DCA for Solving (11)

In this section we investigate a DC programming approach for solving (11). In recent years, D.C. programming has been developed extensively, becoming an attractive topic of research in nonconvex programming. A DC program has the following form

$$\alpha := \min\Big\{f(x) := g(x) - h(x) : x \in \mathcal{R}^n\Big\} \qquad (13)$$

with g, h being lower semi-continuous proper convex functions on $\mathcal{R}^n$, and its dual is defined as

$$\alpha := \min\Big\{h^*(y) - g^*(y) : y \in \mathcal{R}^n\Big\} \qquad (14)$$

where $g^*(y) := \max\{x^T y - g(x) : x \in \mathcal{R}^n\}$ is the conjugate function of g.

Based on local optimality conditions and duality in DC programming, the DCA consists in the construction of two sequences $\{x^k\}$ and $\{y^k\}$, candidates to be optimal solutions of primal and dual programs respectively, in such a way that $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and their limits points satisfy the local optimality conditions. The idea of DCA is simple: each iteration of DCA approximates the concave part $-h$ by its affine majorization (that corresponds to taking $y^k \in \partial h(x^k)$) and minimizes the resulting convex function.

Generic DCA scheme:
Initialization Let $x^0 \in \mathcal{R}^n$ be a best guest, $0 \leftarrow k$.
Repeat

- Calculate $y^k \in \partial h(x^k)$
- Calculate $x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathcal{R}^n\} \quad (P_k)$
- $k + 1 \leftarrow k$

Until convergence of x^k.

Convergence properties of DCA and its theoretical basis can be found in (8), (9), (10), for instance it is important to mention that:

- DCA is a descent method (the sequences $\{g(x^k) - h(x^k)\}$ is decreasing) *without linesearch.*
- If the optimal value of problem (13) is finite and the infinite sequence $\{x^k\}$ is bounded then every limit point x^* of $\{x^k\}$ is a critical point of $g - h$.
- DCA has a *linear convergence* for general DC programs.
- DCA has a finite convergence for polyhedral DC programs ((13) is called polyhedral DC program if either g or h is polyhedral convex).

We now describe the DCA applied to the DC program (12). By the very first definition of h, a sub-gradient $(u,v) \in \partial h(P,s)$ can be chosen

$$(u,v) \in \partial h(P,s) \leftarrow u_j^l = -1; \qquad (15)$$
$$v_j^l = \eta_j^l + \tilde{t} \text{ if } s_j^l \geq 0.5, \text{otherwise } v_j^l = \eta_j^l - \tilde{t}.$$

Algorithm 1 (DCA applied to (11))

Let $\epsilon > 0$ be small enough and (P^0, s^0). Set $k = 0$, $er = 1$.
while $er > \epsilon$ **do**

- Compute $(u^k, v^k) \in \partial h(P^k, s^k)$ via (16).
- Solve the linear program: $\min\{-{u^k}^T P - {v^k}^T s : (P,s) \in K\}$ to obtain (P^{k+1}, s^{k+1}).
- Set $er = \|(P^{k+1}, s^{k+1}) - (P^k, s^k)\|$, $k = k + 1$.

endwhile

Regarding the complexity of the proposed DCA, besides the computation of the sub-gradients which is trivial, the algorithm requires one linear program at each iteration and it has a finite convergence. The linear program has polynomial complexity. The convergence of Algorithm 1 can be summarized in the next theorem (9).

THEOREM 3:

i) Algorithm 1 generates a sequence $\{(P^k, s^k)\}$ contained in $V(K)$ such that the sequence $\{g(P^k, s^k) - h(P^k, s^k)\}$ is decreasing.

ii) If at iteration r we have $s^r \in \{0,1\}^{LJ}$, then $s^k \in \{0,1\}^{LJ}$ and $f(P^{k+1}, s^{k+1}) \leq f(P^k, s^k)$ for all $k \geq r$.

iii) The sequence $\{(P^k, s^k)\}$ converges to $\{(P^*, s^*)\} \in V(K)$ after a finite number of iterations. The point $\{(P^*, s^*)\}$ is a critical point of Problem (11). Moreover such an (P^*, s^*) is almost always a strict local minimum of (11).

PROOF: i) is a convergence property of general DC programs (9), (10) while ii) and iii) can be deduced from Proposition 2 in (6). □

Since DCA works on the continuous problem (11), its solution may not be integer, i.e. not feasible to (MILP). For obtaining an integer solution we combine DCA with the branch and bound method in which a lower bound is computed by solving the corresponding relaxed linear problem. At each iteration we restart DCA from the optimal solution of the relaxed problem. We stop the combined algorithm when the solution furnished by DCA is feasible to (MILP).

Algorithm 2: DCA with starting points obtained by BnB

```
    Set $R_0 := [0,1]^{LJ}$, $k := 0$.
    Solve the linear relaxation problem of MILP to obtain an optimal solution $(P^0, s^0)$ and the optimal value $\beta(R_0)$.
    **If** $(P^0, s^0)$ is feasible of MILP **then** STOP
    **else**: solve (11) by DCA from the starting point $(P^0, s^0)$ to obtain $(\overline{P}, \overline{s})$.
    **If** $(\overline{P}, \overline{s})$ is feasible of MILP, **then** STOP
    **else** set $\Re = \{R_0\}$ and go to the iteration step.
**While** (stop = false) **do**
    • Set $k := k + 1$ and select a rectangle $R_k$.
    • Let $j*$ be the index to be separated. Divide $R_k$ in to two rectangles $R_{k_0}$ and $R_{k_1}$ such that
          $R_{k_i} = \{s \in R_k : s_{j^*} = i, i = 0, 1\}.$
    • For each $i = 0, 1$ solve the corresponding relaxed linear problem to obtain an optimal solution $(P^{k_i}, s^{k_i})$ and the optimal value $\beta(R_{k_i})$.
    • Launch DCA from $(P^{k_i}, s^{k_i})$ to obtain $(\overline{P^{k_i}}, \overline{s^{k_i}})$.
    • **If** $(\overline{P^{k_i}}, \overline{s^{k_i}})$ is feasible of MILP, **then** STOP **else**:
          $\Re \leftarrow \Re \cup \{R_{k_i}; i = 0, 1\} \setminus R_k$
**endwhile**
```

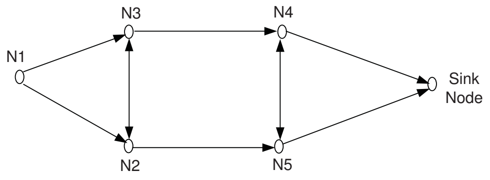


Figure 1: The network model used in Section V

We adopted an adaptive procedure for the choice of rectangle to be separated: choose the rectangle such as the optimal solution of the corresponding linear relaxed problem has one of the components s_j^l, for the links l connected to the sink node, is not integer; otherwise we choose the rectangle corresponding to the smallest lower bound.

COMPUTATIONAL EXPERIMENTS

In this section, we provide preliminary computational results of our approach. We have coded the **Algorithm 2** in C++ programming language and tested the instances using PC Pentium 4 3GHz, 1GB RAM. CPLEX 9.1 is used to solve the linear programs. The small-size network with 6 nodes and 10 links as in Figure 1 has been tested. It is worth mentioning that most of currently deployed wireless networks, for example sensor networks are of small scale which centralized synchronous TDMA is viable. Moreover, the implementation of centralized large scaled networks are extremely difficult, if not impossible. If that is the case, one likely approach is to partition the network into smaller clusters and our proposed design can be applied for each cluster. The node coordinates are showed in Table 1. The maximum transmit power is taken to be equal to $P_{max} = 5$. The noise variance $\eta = -20$ dB. The SNR threshold γ^{th} equals to 10 dB. Energy consumption for transmitting and receiving 1 unit data ϵ_l, ε_l is assumed to be 0.25. The link gains are computed using the path loss model as $h_{ij} = \frac{1}{10}[\frac{1}{d}]$ for $i \neq j$, and $h_{ii} = [\frac{1}{d}]$ where d is the Euclidean distance between nodes. The factor of $\frac{1}{10}$ can be viewed as the spreading gain in a CDMA system. We have tested this network with the different number of time slots $J = 10, 15, 20, 25, 30$.

In Table 2, we report the results of **Algorithm 2** (the number of iterations and the value of the objective function calculated by the algorithm). For evaluating the efficiency of **Algorithm 2** we indicate in this table the optimal value given by CPLEX 9.1 applied to (MILP). The following notations are used: J: the number of time slots; $VarC$: the number of continuous power variables $P_j^l,\ j = 1, \ldots, J,\ l = 1, \ldots, L$; $VarB$: the number of binary scheduling variables $s_j^l,\ j = 1, \ldots, J,\ l = 1, \ldots, L$; Con: the number of constraints in the optimization problem (4a)–(4h); $Value$: the computing ob-

Table 1: Node coordinates

Node	N1	N2	N3	N4	N5	Sink node
Coordinates	(-20,20)	(0,0)	(0,40)	(40,40)	(40,0)	(80,25)

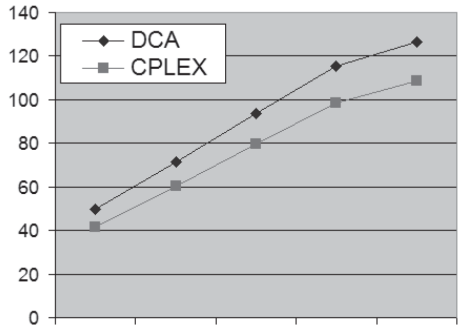


Figure 2: Comparative results of objective values between DCA and CPLEX

jective value by Algorithm 2; $iter$: the number of iterations of Algorithm 2. $OptVal$: the optimal value of (MILP) and $Gap = \frac{Value-OptVal}{Value}100\%$.

From Figure 2 and the column Gap in Table 2, it is clear that the solutions given by DCA are close enough to the optimal solutions. The ability to handle very large-scale problems makes the proposed method implementable for practical networks.

RELATED RESEARCH

There are numerous existing results in the areas of cross-layer design. Hereafter, we mention only the works which are mostly related to the research in this paper. In particular, we consider the system and interference model as in (1), (11). (11) presents a joint link scheduling and power control scheme for TDMA-based networks. Moreover, routing is assumed to be fixed and the network throughout, i.e., sum of links' throughput is maximized. A heuristic polynomial time algorithm to solve the proposed MILP is proposed. Our proposed formulation can be seen as an extension to the work in (11) where we also incorporate rate control, routing with quality-of-service (QoS) constraints on the end-to-end flows.

Routing algorithms have been designed to prolong the network lifetime (1), (12). In (1), a cross-layer design across physical, MAC and routing layers is proposed to maximize the network lifetime which is defined as the earliest time when the first node dies. Optimal TDMA scheduling to maximize the average transmission rate or to minimize the cross-link interference given fixed link transmission powers is considered in (14). Unsurprisingly, the resulting formulation is also a MILP but no efficient solution approaches are proposed. In (2), the authors investigate the problem of joint routing, link scheduling and power control in wireless multi-hop networks. The objective of the optimal policy is the minimization of the total average transmission power given that each link attains the minimum average data rate. The proposed approach via duality is applicable at low SINR regions since the capacity is assumed to be a linear function of SINR.

CONCLUSION

In this paper, we have studied the cross-layer design problem in an interference-limited TDMA wireless network. Particularly, the problem of joint rate control, routing, link scheduling and power control has been considered to minimize the energy consumption. The proposed design can be formulated as a mixed-integer linear program which has worst case exponential complexity to compute optimal solution. Our main contribution was to propose a computationally efficient approach based on DCA. The considered combinatorial optimization problem has been beforehand reformulated as a DC program with a natural choice of DC decomposition, and the resulting DCA then consists in solving a finite sequence of linear programs. DCA is original because it gives an integer solution while it works in a continuous domain. Preliminary numerical results were encouraging and demonstrated the effectiveness of the proposed method. Moreover, notice that most problem formulations arising in TDMA-based networks can be formulated as some sort of MILP problems, our proposed approach seems attractive and needs more investigation.

REFERENCES

[1] R. Madan, S. Cui, S. Lall, and A. Goldsmith, "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 5, no. 11, pp. 3142-3152, Nov. 2006.

[2] R. L. Cruz, and A. V. Santhanam, "Optimal routing, link scheduling and power control in multi-hop wireless networks," in *Proc. IEEE INFOCOM'03*, pp. 702–711, San Francisco, USA, Mar. 2003.

[3] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452-1463, Aug. 2006.

[4] H. Jiang, W. Zhuang, and X. Shen, "Cross-layer design for resource allocation in 3G wireless networks and beyond," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 120-126, Dec. 2005.

Table 2: Computational results of Algorithm 2

J	VarC	VarB	Con	iter	Value	OptVal	Gap(%)
10	100	100	266	9	49.5	41.57855	16.0
15	150	150	396	72	71.5	60.56757	15.3
20	200	200	526	205	93.5	79.55660	14.9
25	250	250	656	770	115.5	98.54560	14.7
30	300	300	786	369	126.5	108.53460	14.2

[5] S. Ramanathan, and E.L. Lloyd, "Scheduling algorithms for multi-hop radio network," *IEEE ACM Trans. Networking*, vol. 1, no. 2, pp. 166-177, Apr. 1993.

[6] H. A. Le-Thi, T. Pham Dinh, "A continuous approach for globally solving linearly constrained quadratic zero-one programming problems," *Optimization*, vol. 50, pp. 93-120, 2001.

[7] H. A. Le-Thi, T. Pham Dinh, and M. Dung Le, "Exact penalty in DC programming," *Vietnam Journal of Mathematics*, pp. 1216-1231, 1999.

[8] H. A. Le-Thi, and T. Pham Dinh, "The DC (Difference of Convex functions) Programming and DCA revisited with DC models of real world nonconvex optimization problem," *Annals of Operations Research*, pp. 23-46, 2005.

[9] T. Pham Dinh, and H. A. Le-Thi, "Convex analysis approach to DC programming : Theory, Algorithms and Applications," *Acta Mathematica Vietnamica*, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, pp. 289-355, 1997.

[10] T. Pham Dinh, and H.A. Le Thi, "DC optimization algorithms for solving the trust region subproblem," *SIAM Journal Optimization*, vol. 8, pp. 476-505, Feb. 1998.

[11] J. Tang, G. Xue, C. Chandler, and W. Zhang, "Link scheduling with power control for throughput enhancement in multihop wireless networks," *IEEE Trans. Vehicular Tech.*, vol. 55, no. 3, pp. 733-742, May 2006.

[12] J.-H. Chang, and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. IEEE INFOCOM'00*, pp. 21-31, Tel-Aviv, Israel, Mar. 2000.

[13] C. W. Commander, and P. M. Pardalos, "A combinatorial algorithm for the TDMA message scheduling problem," to appear *Computational Optimization and Applications*, 2008.

[14] R. Madan, S. Cui, S. Lall, and A. Goldsmith, "Mixed integer-linear programming for link scheduling in interference-limited networks," *Proc. of 1st workshop on Resource Allocation in Wireless Networks,* Italy, Apr. 2005.

[15] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proc. IEEE INFOCOM'06*, pp. 1-13, Barcelona, Spain, Apr. 2006.

[16] K. T. Phan, H. Jiang, C. Tellambura, S. A. Vorobyov, and R. Fan, "Joint medium access control, routing and energy distribution in multi-hop wireless networks," *under revision, IEEE Trans. Wireless Commun.*.

AUTHOR BIOGRAPHIES

LE THI HOAI AN received her Ph D degree and Habilitation degree in 1994 and 1997, respectively in Modelling, Optimization and Operations Research from University of Rouen, France. Since 2003 she has been full professor in the Department of Computer Science, University Paul Verlaine - Metz, France. Professor Le Thi is the director of the Laboratory of Theoretical and Applied Computer Science. Her research interest is in the area of optimization and operations research and their applications in data mining, bioinformatics, image analysis, cryptology, finance, telecommunication, transportation, supply chain and management.

NGUYEN QUANG THUAN received the BSc. and MSc. degrees from Hanoi University of Technology, Vietnam in 2004 and Metz University, France in 2007, respectively. He is currently working toward the Ph.D. degree at the Laboratory of Theoretical and Applied Computer Science, Paul Verlaine Metz University. France. His research interest is in the broad areas of global optimization, for example large scale combinatorial optimization, and its applications.

PHAN TRAN KHOA is currently a MSc. student at iCORE Wireless Communications Lab, University of Alberta, Canada. He will join California Institute of Technology (Caltech), USA as a PhD student in September 2008. He obtained his BSc. from the University of New South Wales, Australia in 2005. He has been the recipient of numerous prestigious fellowships including the Australian Development Scholarship, the Alberta Ingenuity Fund Fellowship, and the iCORE Graduate Student Award. His research interests include wireless communications, networking and optimization.

PHAM DINH TAO received the Doctor of Sciences degree in Numerical Analysis and Optimization in 1981 from the University Joseph-Fourier, Grenoble, France. He held positions up to 1989 as Associate Professor at the same University. Since 1989, he has been full professor with the Department of Mathematics Engineering at the National Institute of Applied Sciences, Rouen, France. He is currently Director of Laboratory Modelling, Optimization and Operations Research. His research interests include nonconvex programming: Local and global approaches, theory, algorithms, and their applications in transportation-logistics, finance, telecommunication, data mining, computer vision, pattern recognition, cryptology, bioinformatics, management science, structure mechanics.

Evaluation of Different Optimization Techniques in the Design of Ad Hoc Injection Networks

Bernabé Dorronsoro, Grégoire Danoy, Pascal Bouvry, and Enrique Alba

Abstract— **Injection networks arise as a way to deal with the network partitioning problem in ad hoc networks. In this kind of networks, it is assumed that devices might have other communication interfaces rather than Wi-Fi and/or Bluetooth that allow them to connect to remote devices, such as GSM/UMTS. The problem considered in this work is to establish remote links between devices (called bypass links) in order to maximize the QoS of the network by optimizing its properties to make it small world. Additionally, these bypass links are not free, so the number of this kind of links in the network should be minimized as well. We face the problem with six different GAs and compare their behaviors. These alorithms are two panmictic algorithms, two GAs with the population structured in islands and two cellular GAs. One of the island GAs (a simple distributed GA with steady-state GAs runing in the islands) and the two cellular GAs were applied here for the first time to this problem. The other island GA, a cooperative coevolutionary GA, is considered the current state-of-the-art algorithm for this problem. As a result, we conclude that the two cellular GAs outperform all the compared algorithms, including the CCGA, for the three studied network instances.**

I. Introduction

Mobile multi-hop ad hoc networks most often face the problem of network partitioning. In this work we consider the problem of optimizing *injection networks* which consist in adding long-range links (e.g., using GSM, UMTS or HSDPA technologies) that are also called *bypass links* to interconnect network partitions. To tackle this topology control problem, we use small-world properties as indicators for the good set of rules to maximize the *bypass links* efficiency. Small-world networks [1] feature a high clustering coefficient (γ) while still retaining a small characteristic path length (L). On the one hand, a low characteristic path length is of importance for effective routing mechanisms as well as for the overall communication performance of the entire network. On the other hand, a high clustering coefficient features a high connectivity in the neighborhood of each node and thus a high degree of information dissemination each single node can achieve. This finally motivates the objective of evoking small-world properties in such settings. In order to optimize those parameters (maximizing γ, minimizing L) and to minimize the number of required bypass links in the network, we relied on Evolutionary Algorithms (EAs) and more specifically on Genetic Algorithms (GAs) [2].

This optimization problem was first introduced in [3], where it was solved with two panmictic GAs (generational and steady-state) and a cooperative coevolutionary GA (CCGA), this latter one reporting the best results. However, in that study the CCGA was compared versus two simple panmictic algorithms. In this work we extend this preliminar study by proposing three aditional decentralized GAs, namely a GA distributed in islands running a steady-state GA in each island, and two cellular genetic algorithms: a canonical and a hierarchical one. We consequently compare two panmictic GAs and four decentralized GAs (two island and two cellular GAs, the main kinds of structured GAs) on this complex problem. One important contribution of this paper is the comparison we perform among the CCGA versus other decentralized GAs. As an additional contribution of this work, we found that the two cellular GAs generally outperformed all the compared algorithms, representing the new state of the art for the problem.

The remainder of this paper is organized as follows. In the next section we introduce the injection network problem. Section III provides a brief description of the studied genetic algorithms, as well as the representation used and the fitness function we defined. Then, section IV presents the experiments and discuss the results. The last section contains our conclusions and perspectives.

II. Problem Overview

The problem we study in this article consists in overcoming partitioning in ad hoc networks by optimizing the placement of long range links that we call *bypass links*.

Our initial motivation for the current investigation is based on the assumption that technologies like Bluetooth and Wi-Fi can be used to create ad hoc communication links within the transmission range at no charge. Additional cellular network links such as GSM/UMTS/HSDPA might be employed by appropriately equipped devices to establish supplementary communication links, that we call *bypass links*, between two capable devices. These links will induce additional costs, and they are typically used to connect distant (not in range) devices that are either far away (there is a many-hops communication between them) or not connected (belonging to different clusters) in the network. Practically, a bypass link can be built by using a cellular network as well as by using access points. Nevertheless, in our model a bypass link is counted as a single hop, thus simplifying the real topology behind that bypass link. Devices used for establishing bypass links are called injection points, and self-organizing communication networks based on bypass links and injections points as described here are called *injection networks* (see an example injection network in Fig. 1).

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

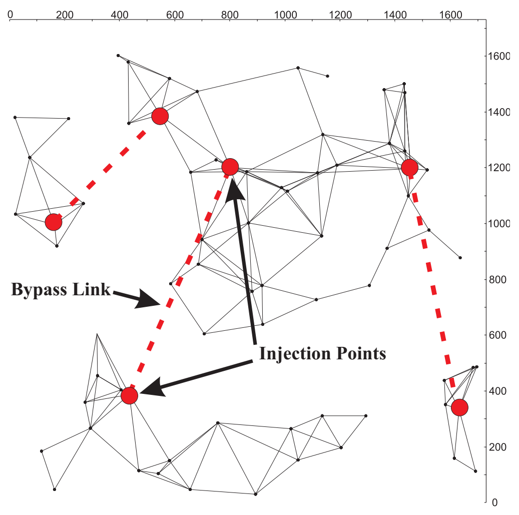


Fig. 1. Example of an Injection Network.

Injection points serve two different purposes: a point where information dissemination starts and where services are being placed (service placement, Herrmann et al. [4]). In the first case, the injection point is of essential importance at the moment of receiving information and passing this information to the neighborhood. The injection point might represent a bottleneck, depending on the amount of data passing through. In addition, injection points become particularly attractive when offering a service. In fact, information dissemination can be seen as such a service that is usable by devices in the injection points surrounding. Hence, the criterion for determining the injection point might highly influence the behavior of the network.

In order to optimize this kind of networks, we consider small-world properties as indicators for the good set of rules to maximize the bypass links efficiency. Small-World networks [1] are a class of random graphs that exhibit a small characteristic path length (L), indicating the degree of separation between the nodes in the graph, and a high clustering coefficient (γ), defining the extent to which nodes in the graph tend to form closely-knit groups that have many edges connecting each other in the group, but very few edges leading out of the group. The challenging aspect in using small-world properties is that small-world networks combine the advantages of regular networks (high clustering coefficient) with the advantages of random networks (low characteristic path length). In order to study the small-world properties of such hybrid networks, we had to rely on some ad hoc network simulator. In our case we used Madhoc [5], an application-level network simulator dedicated to the simulation of mobile ad hoc networks. The main motivation for using Madhoc is its ability to simulate hybrid networks, i.e., mixing different technologies (e.g., bluetooth/Wi-Fi for local connections and UMTS for long distance calls), and its graphical and batch modes of visualization, which greatly help in understanding the network design alternatives.

III. The Algorithms

We compare in this work several algorithms with different population structures (panmictic, cellular, and islands) on the complex problem of optimizing the design of injection ad hoc networks. On the one hand, panmictic algorithms do not consider any structure into the population; so any individual can mate with any other one in the population. On the other hand, in structured, or also called decentralized, populations (e.g., cellular and distributed GAs) individuals can only interact with a subset of the individuals in the whole population. In cellular GAs, a distance measure is defined among all the individuals in the population and only individuals that are close each other can interact. In the case of island (or distributed) GAs, the population is partitioned into several smaller subpopulations that independently evolve, exchanging some information among them during the run.

We present in this section the six algorithms evaluated in this study. Specifically, they are two panmictic GAs, the generational (genGA) and steady-state GAs (ssGA), two cellular ones, a canonical (cGA) and a hierarchical one (HcGA), and two island GAs, a canonical one (dGA) and a coevolutionary GA (CCGA). All these algorithms are briefly described in sections III-A to III-C, while the problem representation and the fitness function are presented in Section III-D.

A. Panmictic Genetic Algorithms

We present in this section the two GAs with centralized population we study in this paper, namely the steady-state (ssGA) and the generational (genGA) genetic algorithms. In panmictic algorithms, any individual in the population can mate with any other one during the breeding loop. These two algorithms perform in a similar way: they iterate a process in which two parents are selected from the whole population with a given selection criterion, they are then recombined, the obtained offsprings are mutated, and finally they are evaluated and inserted back into the population following a given criterion.

The difference between the ssGA and the genGA is the way in which the population is being updated with the new individuals generated during the evolution. In the case of the ssGA, new individuals are directly inserted into the current population (it is a (μ+1)-GA), while in the case of the genGA, a new auxiliary population is built with the obtained offsprings and then, once this auxiliary population is full, it completely replaces the current population (it is a (μ, λ)-GA, with $\mu = \lambda$). Thus, in ssGAs the population is asynchronously being updated with the newly generated individuals, while in the case of genGAs all the new individuals are updated at the same time, in a synchronous way.

B. Cellular Genetic Algorithms

Cellular genetic algorithms (cGAs) [6] are a kind of GA with a structured population in which individuals are spread in a two dimensional toroidal mesh, and they are only allowed

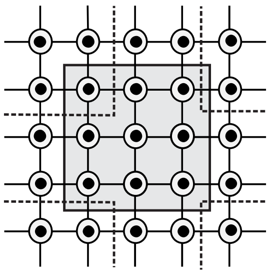


Fig. 2. Example 5×5 population of a cGA with C9 neighborhood

to interact with their neighbors. As an example, we show in Fig. 2 the disposition of the individuals in the population of a cGA, the neighborhood of the center individual (shadowed), and of another individual far from the center, in the upper left corner (dashed line).

A canonical cGA follows the pseudo-code included in Algorithm 1. In this basic cGA, the population is usually structured in a regular grid of d dimensions ($d = 1, 2, 3$), and a neighborhood is defined on it. The algorithm iteratively considers as current each individual in the grid (line 3), and individuals may only interact with individuals belonging to their neighborhood (line 4), so parents are chosen among the neighbors (line 5) with a given criterion. Crossover and mutation operators are applied to the individuals in lines 6 and 7, with probabilities P_c and P_m, respectively. Afterwards, the algorithm computes the fitness value of the new offspring individual (or individuals) (line 8), and inserts it (or one of them) instead of the current individual in the population (line 9) following a given replacement policy. This loop is repeated until a termination condition is met (line 2).

Algorithm 1 Pseudocode for a canonical cGA

```
1: proc Evolve(cga)      //Algorithm parameters in 'cga'
2: while ! StopCondition() do
3:    for individual ← 1 to cga.popSize do
4:       n_list←Get_Neighborhood(cga,position(individual));
5:       parents←Selection(n_list);
6:       offspring←Recombination(cga.Pc,parents);
7:       offspring←Mutation(cga.Pm,offspring);
8:       Evaluation(offspring);
9:       Add(position(individual),offspring,cga);
10:   end for
11: end while
12: end proc Steps_Up;
```

In addition to the previously described canonical cGA, we also study in this paper a hierarchical version of the algorithm, called hierarchical cGA (HcGA) [7]. HcGA is a simple extension of canonical cGAs in which the population structure is augmented with a hierarchy according to the current fitness of the individuals. The basic idea is to put the best current solutions all together in the same region of the population, so that high quality solutions are exploited quickly, while at the same time new solutions are provided by individuals outside this region that keep exploring the search space. This algorithmic variant is expected to increase the convergence speed of the cGA algorithm and to maintain the diversity given by the distributed layout. In [7], the HcGAs were proposed for the first time and were validated in a theoretical and empirical comparison versus an equivalent canonical cGA.

C. Distributed Genetic Algorithms

In addition to the cellular model, there is another common way for structuring the population of GAs. It consists of splitting the whole population into several subpopulations in which isolated GAs are evolving, and these subpopulations exchange some information among them during the run. We study in this paper two algorithms following this model, namely dGA, a simple distributed GA with an ssGA running in every island, and CCGA, a cooperative coevolutionary GA that represents current state of the art for this problem.

The main idea behind coevolutionary algorithms is to consider the coevolution of subpopulations of individuals representing specific parts of the global solution, instead of considering a population of similar individuals representing a global solution, like classical genetic algorithms do. The quality of this kind of algorithms have been reported in a large number of papers in the literature. As an example, two different coevolutionary GAs were applied in [8] on a number of test functions known in the area of evolutionary computation, and they were demonstrated to clearly outperform a sequential GA.

Cooperative (also called symbiotic) coevolutionary genetic algorithms (CCGA) involve a number of independently evolving species which together form complex structures, well-suited to solve a problem (see a pseudocode in Algorithm 2). The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals. The CCGA considered here is based in the model proposed by Potter and DeJong [9], in which a number of populations explore different decompositions of the problem. In this system, each species represents a subcomponent of a potential solution. Complete solutions are obtained by assembling representative members of each of the species (populations). The fitness of each individual depends on the quality of (some of) the complete solutions it participated in,

Algorithm 2 Pseudocode of the CCGA

```
1: gen = 0
2: for all $species_s$ do
3:    $Pop_s(gen)$ = randomly initialized population
4:    evaluate fitness of each individual in $Pop_s(gen)$
5: end for
6: while $termination\ condition = false$ do
7:    $gen = gen + 1$
8:    for all $species_s$ do
9:       select $Pop_s(gen)$ from $Pop_s(gen - 1)$ based on fitness
10:      apply genetic operators to $Pop_s(gen)$
11:      evaluate fitness of each individual in $Pop_s(gen)$
12:   end for
13: end while
```

thus measuring how well it cooperates to solve the problem. The evolution of each species is controlled by a separate, independent evolutionary algorithm. In the initial generation *(t=0)* individuals from a given subpopulation are matched with randomly chosen individuals from all other subpopulations. A fitness for each individual is evaluated, and the best individual in each subpopulation is found. The process of *cooperative coevolution* starts form the next generation *(t=1)*. For this purpose, in each generation a cycle of operations is repeated in a round-robin fashion. Only one current subpopulation is active in a cycle, while the other subpopulations are frozen. All individuals from the active subpopulation are matched with the best values of frozen subpopulations. When the evolutionary process is completed a composition of the best individuals from each subpopulation represents a solution of a problem.

D. Problem Encoding and Fitness Functions

Solution encoding is a major issue in this kind of algorithms since it will determine the choice of the genetic operators applied for exploring the search space. We have used a binary encoding of the solution in which each gene encodes an integer on 15 bits, that corresponds to one possible bypass link in the half-matrix of all possible links. For instance, if the maximum number of bypass links fixed a priori for the network that is optimized is 10, then a chromosome will have 10 genes of 15 bits. Figure 3 shows the example of a chromosome composed of 2 genes (thus the maximum number of created bypass links is 2) on a network of 5 stations. The 5×5 half-matrix represents all the possible links in the network including the already existing local links in the network (i.e. the existing Wi-Fi connections). In the example showed in Figure 3, the first gene (circled) with the integer value 2 stands for the connection between station 1 and station 3 in the corresponding half-matrix (also circled).

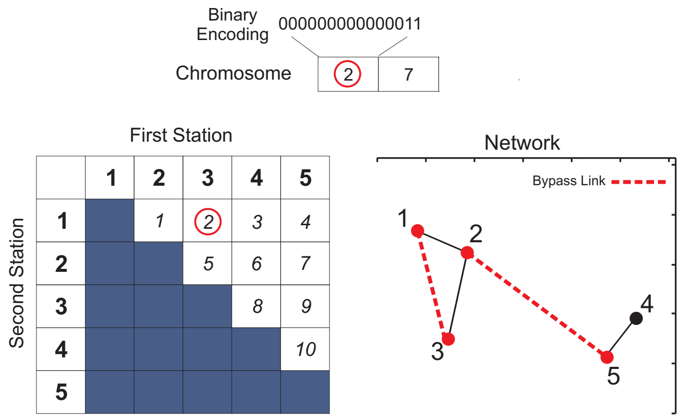


Fig. 3. Solution encoding example

In order to assign a fitness value to the candidate solutions (i.e. sets of possible bypass links) of our algorithms, we use a unique cost function F which combines the two small world measures (L and γ) and the number of created bypass links. The calculation of the characteristic path length L imposes that there exists a path between any given nodes a and b. Consequently, the computation of the fitness function requires that we first test if the network is partitioned.
If the optimized network is still partitioned (the bypass links defined do not achieve to connect all the partitions), the fitness value is assumed to be a weighted term of the number of partitions in the network.
On the contrary, if the optimized network is no longer partitioned, the fitness value is assumed to be a linear combination of the clustering coefficient, of the characteristic path length, and of the difference between the number of bypass links defined and the maximum number allowed.

The aim of the optimization process is to maximize the clustering coefficient, and to minimize both the characteristic path length and the number of bypass links. By using this fitness function we now face the maximization problem defined in Algorithm 3.

Algorithm 3 Fitness Function
```
1: if Graph connected then
2:     F = $\alpha$ * $\gamma$ - $\beta$ * (L - 1) - $\delta$ * ($bl$ - $bl_{max}$)
3: else
4:     fitness = $\xi$ * P
5: end if
```

With weights experimentally defined:
α = 1
β = 1 /(N -2)
δ = 2 / (N * (N-1)) - WifiConnections
ξ = 0.1

where *bl* is the number of bypass links created in the simulated network by one solution, bl_{max} (defined a priori) is the maximum number of bypass links that can be created in the network, *P* is the number of remaining partitions in the whole network after the addition of bypass links and *N* is the number of stations in the global network. Finally, WifiConnections is the number of existing Wi-Fi connections in the network.

IV. Experiments

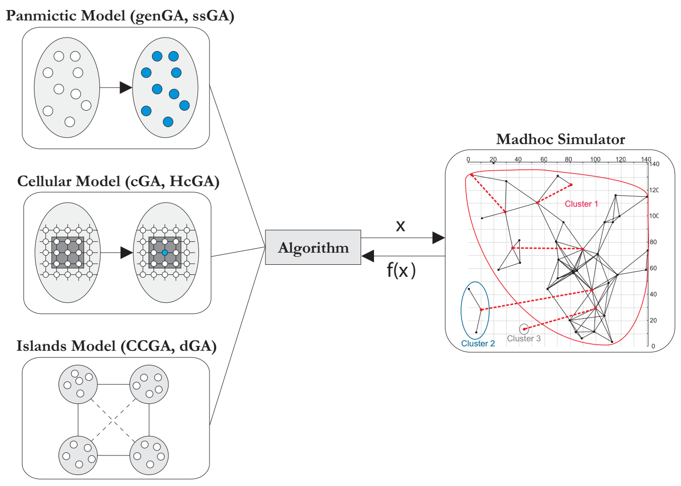
This section presents the results obtained on the injection network optimization problem using the different GAs presented in Section III. We first describe the parameters used for the genetic algorithms. Next, the configuration of the network simulator is introduced and, finally the results obtained using the six GAs are analyzed and compared.

A. Parameterization

In Table I, we show the parameters used for all the proposed algorithms. All of them have a single population of 100 individuals, except for the two distributed algorithms: CCGA (10 populations of 50 individuals), and dGA (5 subpopulations of 100 individuals), having both of them a total population of 500 individuals. The termination condition is achieving 50,000

TABLE I
PARAMETERS USED FOR THE STUDIED GAS

Number of Subpopulations	10 for CCGA
	5 for dGA
(Sub)Population size	100 (genGA, ssGA, dGA)
	10 × 10 (cGA, HcGA)
	50 (CCGA)
Termination Condition	50,000 function evaluations
Selection	Binary tournament (BT)
	Current individual + BT in cGA and HcGA
Neighborhood	C9 in cGA
	C13 in HcGA
Crossover operator	DPX, p_c=0.8
Mutation operator	bit flip, p_m = 1/chrom_length
Elitism	1 individual (not for ssGA)



Panmictic Model (genGA, ssGA)
Cellular Model (cGA, HcGA)
Islands Model (CCGA, dGA)
Algorithm
x
f(x)
Madhoc Simulator
Cluster 1
Cluster 2
Cluster 3

Fig. 4. Components of the experimental study

fitness function evaluations, common to all the algorithms, as well as the recombination (the two points crossover –DPX–) and mutation (bit-flip) operators, and their probabilities: $p_c = 0.8$ and $p_m = 1/\text{chrom\_length}$, respectively.

The two parents are selected using a binary tournament, except for the two cellular algorithms, for which one of them is considered to be the current individual itself. A specific parameter of these cellular models is the neighborhood. We used C9 (9 closest individuals measured in Manhattan distance –see Fig.2–) for cGA and C13 for HcGA. The reason of using a different neighborhood for HcGA is that it maintains a higher diversity in the population than the cGA, and thus we can use a more exploitative neighborhood. Finally, all the algorithms follow an elitist strategy, with the exception of ssGA.

B. Madhoc Configuration

As stated before, the Madhoc simulator was used for managing the complex scenarios posed by this injection network problem. Fig. 4 shows how the genetic algorithms interact with Madhoc.

The parameters we have used in Madhoc for defining our problem instances are shown in Table II. We have defined a square simulation area of 0.2 km^2 and tested three different

TABLE II
PARAMETERIZATION USED IN MADHOC

	1 Cluster	3 Clusters	5 Clusters
Surface	0.2 km^2	0.2 km^2	0.2 km^2
Node Density	350 nodes/km^2	210 nodes/km^2	150 nodes/km^2
Number of Nodes	70	42	30
Partitions	1	3	5
Possible Links	2189	745	400

densities of 150, 210 and 350 devices per square kilometer. Each device is equipped with both Wi-Fi (802.11b) and UMTS technologies. The coverage radius of all mobile devices ranges between 20 and 40 meters in case of Wi-Fi.

The studied networks, as presented in Fig. 5, here represent a snapshot of mobile networks in the moment in which a single set of users moved away from each other creating the clusters of terminals, that were obtained using the graphical mode of Madhoc. As an example, the network with 3 clusters (center of Fig. 5) consists in 42 stations located in three partitions, the first partition has 38 stations, the second one 3, and the third one has a single station. The number of possible connections in this 3-clusters network is $\frac{N*(N-1)}{2}$ = 861. The number of existing Wi-Fi connections in this network is 116, thus the number of possible bypass links is 861-116 = 745. The clusters are selected purposely to be different and thus challenging.

C. Results

In Table III we show the averaged results, the best ones, and the total computational time for all 30 runs for each algorithm. Additionally, we show the results of the statistical tests in the comparison of the different algorithms for each cluster instance in order to obtain concluding results from the comparison made. For performing these statistical tests, we first check whether the data follow a normal distribution or not using the Shapiro-Wilks test. Then, if the data are normally distributed we perform an ANOVA test. In the other case, we use the Kruskal-Wallis test. This statistical study allows us to assess if there are meaningful differences among the compared algorithms with 95% probability or not.

Symbol '+' in Table III stands for existing statistical differences in the comparison of the algorithms. Grey background means that the result is the best one with statistical confidence (if there are more than one result with grey background for the same problem it means that there are no statistical difference between them, but they are better than the others with statistical significance).

As we can see in Table III, the two cellular models and the CCGA are clearly the three best compared algorithms. The two panmictic GAs and dGA obtain the worst results with statistical significance with respect to the best performing algorithm for the three studied instances. If we now compare the CCGA and the two cellular models, we can see that the celullar alorithms outperform CCGA in the case of the 5 clusters instance with statistically significant differences. For the other two instances (1 and 3 clusters), CCGA obtains better

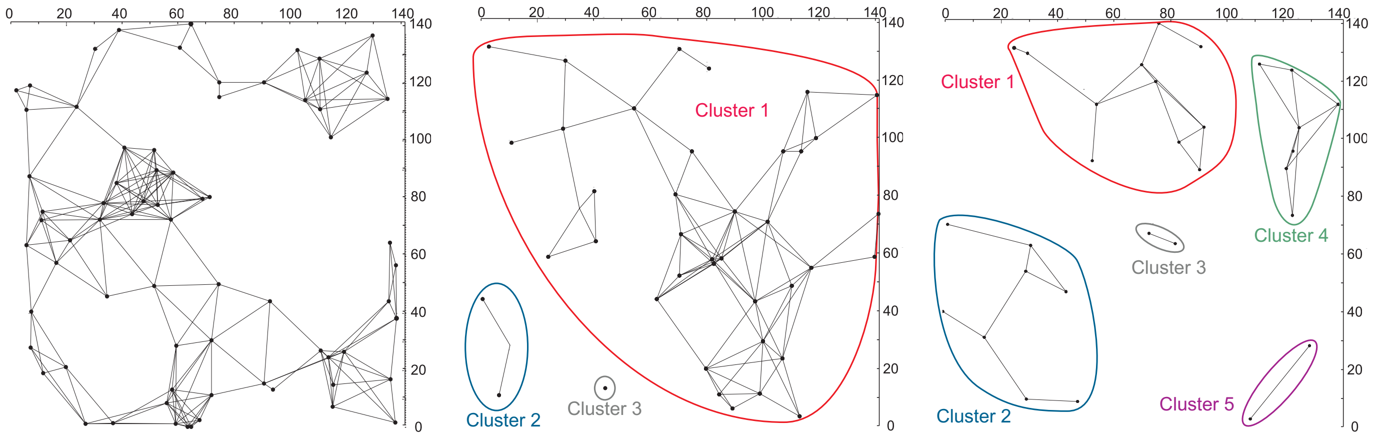


Fig. 5. Studied Networks with 1, 3 and 5 clusters

TABLE III
RESULTS OF ALL EXPERIMENTS

Network	GA	Avg. Result	Best Result	Time (s)	*p*-value
1 Cluster	genGA	0.6833	0.6920	**8256**	+
	ssGA	0.6736	0.6855	8395	
	dGA	0.6760	0.6856	11311	
	CCGA	**0.6911**	0.6925	17784	
	cGA	0.6887	0.6924	10760	
	HcGA	0.6893	**0.6926**	10366	
3 Clusters	genGA	0.6685	0.6782	4486	+
	ssGA	0.6489	0.6651	3289	
	dGA	0.6555	0.6664	5703	
	CCGA	**0.6739**	**0.6757**	6793	
	cGA	0.6727	**0.6757**	3563	
	HcGA	0.6724	0.6754	**2883**	
5 Clusters	genGA	0.5634	0.5848	1672	+
	ssGA	0.5527	0.5809	1717	
	dGA	0.5576	0.5783	3563	
	CCGA	0.5652	0.5864	8674	
	cGA	**0.5790**	0.5923	1713	
	HcGA	0.5779	**0.5931**	**1569**	

average results, but the differences with the cellular models are not statistically significant. If we now pay attention to the best solution found in the 30 runs, HcGA finds the best result for instances of 1 and 5 clusters, while in the case of the 3 clusters problem, both the cGA and CCGA achieve the same best value.

Regarding the computation time, the HcGA is the fastest algorithm among the three best ones. Indeed it is the fastest algorithm out of the six compared ones for instances of 3 and 5 clusters, being only improved by the two panmictic algorithms (wich find much worse results) in the largest instance.

With the goal of better understanding the behavior of the different compared algorithms, we now analyze the evolution of the population during the run for all the alorithms. So, we plot in figures 6 to 8 the evolution of the average of the best fitness values during the execution in the 30 runs for the six algorithms and the three studied problem instances. As it can be seen, the behavior of the algorithms is similar for the three problem instances.

As a general rule, all the algorithms experiment a high improvement of the fitness value during the first function evaluations, but then they get stuck and the improvement of the fitness is very low. The reason for this too slow evolution of the fitness value is that the population (or populations) of the

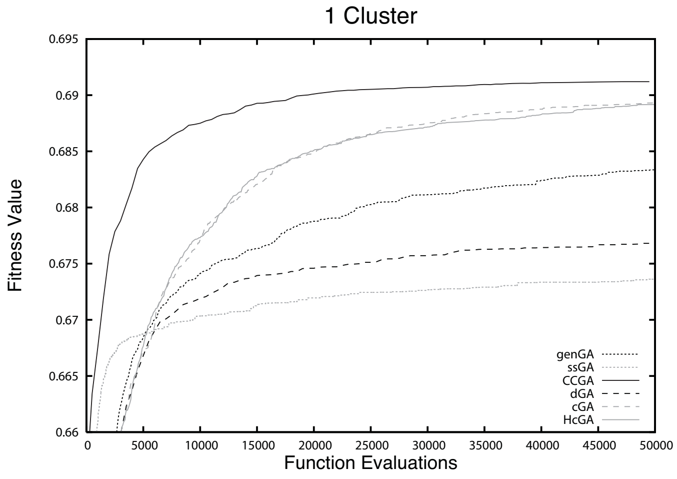


Fig. 6. Evolution of the average best fitness value (30 executions) during the run. One cluster instance

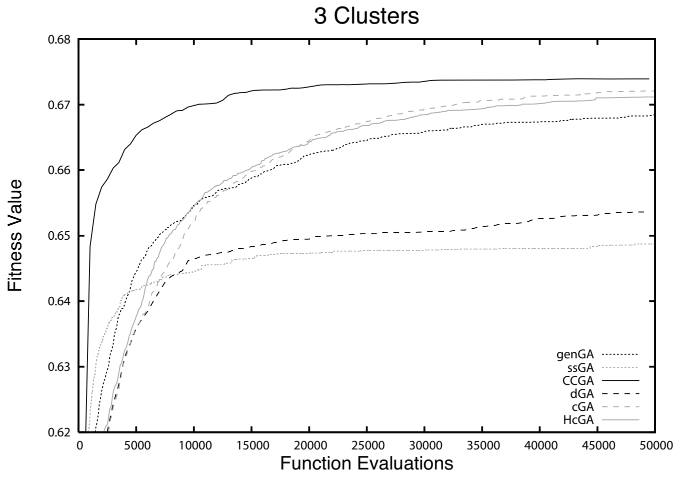


Fig. 7. Evolution of the average best fitness value (30 executions) during the run. Three clusters instance

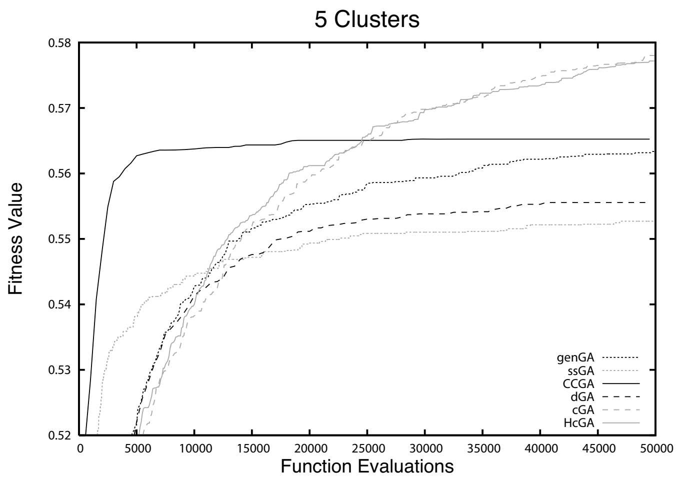


Fig. 8. Evolution of the average best fitness value (30 executions) during the run. Five clusters instance

algorithms prematurely converged in the first steps of the run, loosing the diversity of individuals, and thus making difficult the improvement of the current individuals with the application of the variation operators.

As an exception, the reader can see how the convergence of the cellular GAs is slower from the beginning of the evolution, and it does not get stuck as in the case of the other compared algorithms. The reason is that the cellular model preserves the diversity of the population for longer [6] with respect to the other compared GAs. It thus makes possible to improve the current solutions with the application of the variation operators to the individuals in the population during the breeding loop. We can clearly see this effect in Fig. 8: the fitness value is continuously growing during the whole evolution for the two cellular models, while the other algorithms hardly improve the fitness value in the second half of the run.

The CCGA experiments a really fast convergence during the first generations, but after that the convergence becomes much slower, slightly improving the solution. The two cellular models show a similar behavior in the three problem instances. The convergence of these two algorithms is slower than for the other ones, consequently they need more funtion evaluations to achieve good results, but they maintain the diversity of the population for longer avoiding local optimal solutions.

Finally, it can be seen how the island and the two panmictic GAs prematurely converge to local optimal solutions, from which they find difficulties to escape.

V. Conclusions and Further Works

We have compared in this paper six different GAs on the problem of designing ad hoc injection networks. The compared algorithms are two panmictic algorithms, the generational and steady-state GAs, two algorithms with population structured in islands, a distributed GA with steady-state GAs in the islands and a coevolutionary GA, and two cellular GAs: the canonical and the hierarchical cGA.

Our main conclusion from our comparison study is that the cellular models outperform the other compared ones for the three studied instance problems. The CCGA reports similar average results than the cellular algorithms for the two largest intances, but the cellular models are better in terms of the best solution found and the computational time required. As future works, we propose the study of more realistic instances of the problem. This can be achieved by either considerably increasing the size of the problems or by studying dynamic networks varying with time.

References

[1] D. J. Watts, *Small Worlds – The Dynamics of Networks between Order and Randomness*. Princeton, New Jersey: Princeton University Press, 1999.

[2] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[3] G. Danoy, E. Alba, P. Bouvry, and M. R. Brust, “Optimal design of ad hoc injection networks by using genetic algorithms,” in *GECCO ’07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 2256–2256.

[4] K. Herrmann and K. Geihs, “Self-Organization in Mobile Ad hoc Networks based on the Dynamics of Interaction,” Erlangen, Germany, 2003, frühjahrstreffen der GI-Fachgruppe Betriebssysteme. [Online]. Available: http://www.kbs.cs.tu-berlin.de/publications/fulltext/gi0403.pdf

[5] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, “Simulating Realistic Mobility Models for Large Heterogeneous MANETS,” in *Demo proceeding of the 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM’06)*. IEEE, October 2006.

[6] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*, ser. Operations Research/Compuer Science Interfaces. Springer-Verlag Heidelberg, 2008.

[7] S. Janson, E. Alba, B. Dorronsoro, and M. Middendorf, “Hierarchical cellular genetic algorithm,” in *Evolutionary Computation in Combinatorial Optimization (EvoCOP)*, ser. Lecture Notes in Computer Science (LNCS), J. Gottlieb and G. Raidl, Eds., vol. 3906. Budapest, Hungary: Springer-Verlag, Heidelberg, April 2006, pp. 111–122.

[8] F. Seredynski, A. Zomaya, and P. Bouvry, “Function optimization with coevolutionary algorithms,” in *Proc. of the International Intelligent Information Processing and Web Mining Conference*. Springer, 2003.

[9] M. Potter and K. De Jong, “A cooperative coevolutionary approach to function optimization,” in *Parallel Problem Solving from Nature (PPSN III)*. Springer, 1994, pp. 249–257.

BODYF – A Parameterless Broadcasting Protocol Over Dynamic Forest

P. Ruiz, B. Dorronsoro, D. Khadraoui, P. Bouvry, L. Tardón.

***Abstract*— Mobile Ad hoc Networks (MANETs) are composed of mobile devices which spontaneously communicate each other without any previous existing infrastructure. Thus, the resulting network is highly fluctuating and has a dynamically changing topology. Dealing with ad hoc networks, broadcasting is one of the main operations, since many other applications use it very often. In order to avoid the broadcast storm problem in wireless networks, many researchers work on the design of efficient broadcasting algorithms. In this contribution, we present BODYF, a broadcasting protocol that relys on a tree-based topology. The behavior of this algorithm does not depend on any parameter, what makes it challenging and suitable for dealing with MANETs. We compare here our proposal to DFCN, an efficient broadcasting neighbor topology based protocol, and also to Simple Flooding, the most simple broadcasting technique which does not require any knowledge.**

I. Introduction

Broadcasting constitutes one of the fundamental low-level network operations which serves as the basis of higher level applications, such as routing, in mobile ad hoc networks (MANETs). In MANETs, the limited radio range of the nodes, as well as node mobility, cause the unreachability of some nodes at a given time and a highly fluctuating topology. This is the reason why some researchers are focusing on optimizing the behavior of these algorithms, e.g., maximizing the number of nodes reached, and minimizing both the time required and the network overload [1].

Recently, there exists also a tendency in the ad hoc networks field focused on the development of new mobile networks composed of vehicles. In Vehicular Ad-hoc Networks (VANETs), vehicles can communicate each other (Car to Car communication) or with road-side units that allow access to backend systems which provide warnings, traffic information, etc. Vehicle communication is a major research topic, covered by many national and international research projects as CARLINK [2]. Applications promise to make our driving safer, more efficient, and funnier. This includes warning applications, e.g., cars are able to send warning messages to other cars alerting them of a danger ahead, weather and traffic conditions, etc. In VANETs, due to the high speed of the devices (car PCs), the topology of the network is even more dynamic than in MANETs, what difficulties the communication between them.

D. Khadraoui and P. Ruiz work at C.R.P. Henri Tudor, Luxembourg. B. Dorronsoro and P. Bouvry work at the Faculty of Sciences, Technology and Communications, University of Luxembourg. L. Tardón is with the Department of Communication Engineering, University of Málaga, Spain. (Email: {djamel.khadraoui, patricia.ruiz}@tudor.lu {bernabe.dorronsoro, pascal.bouvry}@uni.lu, lorenzo@ic.uma.es).

D. Khadraoui and P. Ruiz acknowledge that this work was partially supported by CELTIC CARLINK project financed by the Ministry of Luxembourg (CARLINK project website: `http://carlink.lcc.uma.es`)

In MANETs, we can typically differentiate broadcasting protocols into *heuristic-* and *topology-based protocol*. We are interested in *topology-based protocols* which are subcategorized into *neighbor topology based protocol*, *source-tree based protocol* and *cluster-based protocol* [3]. In this paper, we propose a new source-tree based broadcasting algorithm, called Broadcasting Over Dynamic Forest (BODYF). We will compare BODYF to a neighbor topology based protocol, Delayed Flooding with Cumulative Neighborhood (DFCN). In contrast to BODYF, which requires a tree-based topology established in the network, DFCN does not exchange any message for stablishing its topology; it just needs the one hop neighborhood knowledge obtained using the beacons (*hello messages* that devices send for notifying their presence). We will also compare BODYF to Simple Flooding, a well known broadcasting technique which does not require any knowledge about the neighborhood or the topology, and either makes any attempt to reduce the number of forwarded messages.

In addition to the comparison (in terms of coverage, bandwidth and elapsed time) of the behavior of these so different broadcasting protocols, which is already interesting by itself, with this study we can also check, whether the overload caused by the tree-based topology is worthy or not.

The rest of this paper is organized as follows: Section II describes DAGRS, the model for creating topologies used for the creation of our tree, and presents BODYF, the broadcast algorithm over this tree. Section III introduces DFCN the broadcast protocol with one hop neighborhood knowledge. The mobility model and the simulator used are presented in Section IV. After that, the results are shown in Section V and finally, Section VI concludes the paper.

II. Broadcasting Over Dynamic Forest, BODYF

In this section we present BODYF, a broadcasting protocol over a dynamic forest. Although broadcasting using a tree structure is a well known and widely used technique [4], it is typically claimed to be inappropriate for ad hoc networks, being the maintenance of the tree very sensitive to network changes. In this work, we built the tree using DAGRS (dynamicity aware graph relabeling systems) [5], a general model for creating dynamic topologies, and developed the broadcast algorithm on top of this topology.

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

A. Dynamicity Aware Graph Relabeling Systems, DAGRS

DAGRS is an extension of Graph Relabeling Systems, GRS [6]. It is a high level abstraction model that can be used to drastically improve the development of self-organized systems. The model supports the design of algorithms in which a computation step only involves direct neighbors and where a device can react to the appearance/disappearance of its neighbors [7]. The main advantage of DAGRS is that it makes the description of algorithms easier to understand and validate.

The network is represented as a graph, where the mobile devices are the set of vertices *(V)*, and the links between them are the edges of the graph, *(E)*. The dynamicity of the network is represented by the fact that both *V* and *E* can change at any time.

It is important to emphasize that DAGRS model does not itself model services or applications, it just models the mechanisms to handle with topology changes and interaction between devices.

A spanning tree of a graph is a connected cycled-free subgraph. In fact, in dynamic networks we should talk about spanning forest, since the network is typically partitioned. In this model we only use one-hop neighbor information, so it is a localized algorithm.

Initially, all devices are labelled *T*, what means they are tree themselves. The algorithm performs on the basis of four rules described after and represented in Figure 1; where *T* represents a node with token, *N* is a device without token, and *Any* means it can be both of them. The numbers on the edges are labels representing the route to the token.

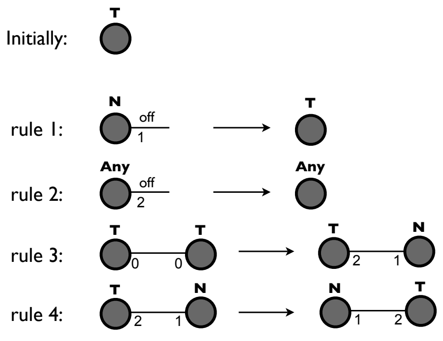


Fig. 1. DAGRS rules for creating spanning forest topologies.

Every tree has only one token, and the possible operations of this token are: circulation, merging, and regeneration. As explained before, every device is initially a tree itself, with only one element. Only two nodes with token can merge. When two nodes with token meet each other, rule 3 allows merging the two trees, deleting one of the tokens (there only can be one token in each tree). The circulation of the token is represented by rule 4. In our implementation, the token explores the tree in a depth-first manner. Both rules 1 and 2 deal with a broken link. In rule 1, the token must be regenerated (label 1 represents the route to the token, so if this link breaks the token is lost), meanwhile in rule 2, the node has nothing to do with the maintainance of the token (label 2 represents a different link from the route to the token). Applying these easy rules to every device, we can build a dynamic tree topology in the network, in a decentralized way.

It is very important to remark that there is only one token per tree in the forest, since it is the way for avoiding cycles. Only two nodes with token can merge, so since there is only one token in the tree, two nodes belonging to the same tree are not able to merge never (it is impossible both of them have the token at the same time).

As we explained above, we are using these simple rules of DAGRS for creating and maintaining the tree, but as we are dealing with high speed mobile environments and also with distributed systems, we exchange some messages between nodes for merging trees and also for circulating the token. DAGRS do not specify how to implement a token. In our protocol, the token is a message. For circulating the token, a device sends this message to one of its neighbors. When a device receives the token, it should check if there is any neighbor with token in its neighborhood just to merge their trees, but this behaviour leads in a high level number of messages interchanged. So we take advantage of the high mobility and use it: only when a change in the one hop neighborhood is detected, the device with token checks if there is any other device nearby with token.

B. The proposed broadcasting protocol, BODYF

BODYF is a broadcasting protocol specifically designed for communication dynamic networks based on spanning tree topologies. We suppose the tree-based topology is already established in the network, maybe for routing or any other necessity [8]. Once we have the tree (or the forest due to the network partitioning), we use it for broadcasting. Therefore, our main goal is not the tree itself, but the design of a new broadcasting protocol that can achieve the best possible coverage and broadcast time at a minimum cost, using the information of the network provided by the tree-based topology.

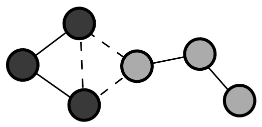


Fig. 2. Possible types of neighbors in a spanning forest algorithm.

Assuming we have the forest, we will distinguish between logical neighbors, which are the ones belonging to the same tree, and potential neighbors, which are those that do not belong to my tree yet but are in communication range, as it is shown in Figure 2. In this picture, devices in same color belong to the same tree. The links between devices are represented as a continuous line if the neighbors belong to the same tree and are connected (logical neighbors), or as a discontinuous line for neighbors either in communication range but which do not belong to the tree (potential neighbors) or belonging to the same tree but not connected in order to avoid cycles.

A pseudocode of our proposed broadcasting algorithm is given in Algorithm 1. We suppose that all the messages have an unique identifier, and when a device receives the same message more than once, it will directly discard it, no processing is done.

When a device wants to spread a message, instead of doing a multicast only to its logical neighbors (the neighbors in the same tree), it will broadcast the information, what supposes the same load for the network, allowing the potential neighbors to also receive the message. When a device receives the message from a logical neighbor, it will forward it if it was not received before, otherwise it is dropped (lines 2-4). But if it was not received from a logical neighbor (i.e., from its own tree), the device will wait until it receives the token (line 5). Once the device receives the token, it will forward the message if and only if, during the time it was waiting for the token, it did not receive the same message again from a neighbor belonging to its tree (lines 6-10). That is the way the message can spread through different trees, trying to avoid the dissemination of the same message more than once in the same tree.

Algorithm 1 Pseudocode of BODYF.

```
Data: m: the incoming broadcast message.
Data: d: the node receiving broadcast message.
Data: s: the node which sent m.
 1: if m is received for the first time then
 2:    if s and d belong to the same tree then
 3:       d → forward m;
 4:    else
 5:       wait until the token is received → d is token;
 6:       if d received m also from its tree then
 7:          d → discard m;
 8:       else
 9:          d → forward m;
10:       end if
11:    end if
12: else
13:    drop m
14: end if
```

III. Delayed Flooding with Cumulative Neighborhood, DFCN

DFCN [9], [10] is an efficient broadcasting protocol specially designed for MANETs. It needs the information about the one hop neighbors provided by the beacons. DFCN requires to embed into the broadcast message a list of all the neighbors of the sender nodes. This broadcasting algorithm focuses on different goals:

1) Minimize network overhead.
2) Provide a protocol dealing with the network density. More precisely, as illustrated in Figure 3, broadcasting in a low density network is difficult because the protocol needs to make a very good use of the mobility to achieve good coverage. When the density increases, the connectivity gets better, although the network may be partitioned. When the network is highly connected, the broadcasting protocol must be bandwidth-efficient in order to minimize the risk of packet collisions.
3) Provide a localized protocol which operates with 1-hop neighborhood information. One challenge is to achieve better performance by using less network information.

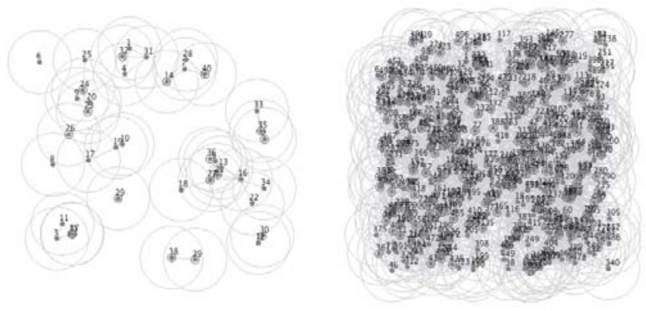


Fig. 3. Node density in MANETs. Picture on the left is a very sparse and partitioned network. The one on the right shows a very dense network.

In this section we explain how DFCN works [9]. First of all, we give a general idea of the problem and, after that, we explain the problem in detail.

When the network is sparse, it is quite difficult to spread a message, hence a node should forward the message as soon as another device is met. This would lead to good results since every re-emission proves to be useful because of the reduced number of meeting points between stations. However, in a dense environment, this strategy would lead to catastrophic results. As a consequence, DFCN sets a random delay (*RAD*) when a node receives a message for the first time (before forwarding it). If the density is low it will forward the message inmediately after a new neighbor is met, but if the density is high, it will wait before resending it (in order to avoid collisions). This perception of the density corresponds directly to its neighborhood and, in DFCN, it is managed with a threshold called *densityThreshold*.

DFCN attaches in the message a list with the neighbors of the sender, *T(m)*. The list *T(m)* is managed as follows: when a node *s* sends a message *m* to its neighbors, it knows that all of them will receive *m* (unless some collisions occur). *T(m)* is set with the neighbors of *s*, *N(s)*. DFCN uses this list to decide whether the received message will be forwarded or not, in terms of the neighbors that already received the message which are in the one hop neighborhood. For that, we set a threshold called *minBenefit* which is formally defined on the *benefit*, computed as the ratio between the neighbors of *s* which do not belong to *T(m)*, and the total neighbors of *s*, *N(s)*, see Equation 1. The higher the benefit, the higher the probability of re-emission.

$$benefit = \frac{|N(s)\text{-}T(m)|}{|N(s)|} \qquad (1)$$

The behavior of DFCN is driven by the following three events:

- the reception of a message, referred to as reactive behavior;
- the expiration of the *RAD* of a message;
- and the detection of a new neighbor, referred to as proactive behavior;

When one of these three events occurs, DFCN reacts by behaving in a specific manner, as described below.

1) Message reception event: if a message *m* is received for the first time, a *RAD* is then assigned to *m*. Otherwise the message is dropped. This behavior corresponds to Algorithm 2. All messages are univocally identified. If a device already received the message, it will be discarded. No processing is done.

Algorithm 2 Behavior of DFCN upon message reception.

```
Data: m: the incoming broadcast message
Data: s: the node which recives m
1: if m is received for the first time then
2:     rad(m) ← random ∈ [0, maxRAD];
3: else
4:     s drops m
5: end if
```

2) RAD *expiration event:* when the *RAD* of a message expires, its hosting node computes the ratio of neighbors that did not receive it yet. If the ratio is greater than the threshold *minBenefit*, the message is forwarded, otherwise it is dropped. If the message is emitted, then *T(m)* is set to *N(s)*. Algorithm 3 shows this behavior.

Algorithm 3 Behavior of DFCN when *RAD* expires.

```
Data: m: the message candidate to immediate emission
Data: s: the node that received m
1: benefit ← $\frac{|N(s)-T(m)|}{|N(s)|}$
2: if benefit ≥ minBenefit then
3:     T(m) ← N(s);
4:     s → forward m
5: end if
```

3) New neighbor event: each time a node *s* detects a new neighbor, the *RAD* for all messages is set to zero. Messages are hence immediately candidate to re-emission. If *N(s)* is higher than the threshold *densityThreshold* (our network is dense), this behavior is disabled, see Algorithm 4.

DFCN behaves depending on several parameters (e.g., *densityThreshold*, *minBenefit* and *RAD*) which need to be tuned for every different network. As an example, DFCN was optimized in [11] for three particular scenarios: mall, highway and metropolitan area, reporting very different parameter values. This optimization was made according to the coverage, the usage of the network and the broadcasting time, depending on the necessities of each application.

Algorithm 4 Behavior of DFCN when a new neighbor is detected.

```
Data: M(s): the set of messages received
Data: s: node with a new neighbor
1: if $|N(s)| <$ densityThreshold then
2:     for m ∈ M(s) do
3:         rad(m) ← 0;
4:     end for
5: end if
```

IV. Mobility Model and JANE Simulator

In this work, we evaluate BODYF and compare it to DFCN in a particular environment, Vehicular Ad hoc Networks (VANETs). So, we consider every device is a car, with communication capabilities. Our scenario for the simulations is the city center of Luxembourg, see Figure 4. The simulation area is 1323m x 1863m, what means 2.465 Km^2. The speed of cars oscillates between 30 and 75 km/h.

Many researchers are focusing on the realism of simulations, and are hence, creating their own mobility models to emulate how the protocols would behave in a real scenario [12]. We did the same and developed a mobility model where devices move along streets, stop at crossroads and also overtake each other.

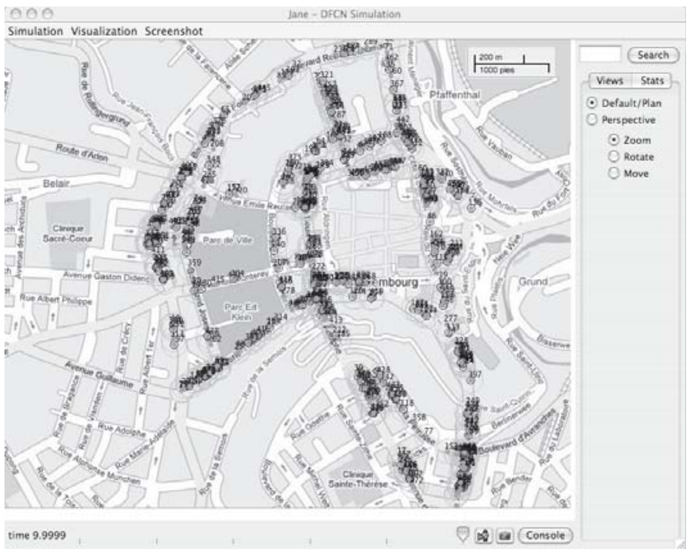


Fig. 4. Luxembourgian mobility model.

The devices move on a straight line with a constant speed from one position to another. The mobility model uses a directed graph given as XML file for device movement. Vertices are crossroads and contain routing probabilities, all possible destinies have the same probability of being taken; the next route is chosen at random. Edges can have an arbitrary width so that devices move on a lane and not only on a strict line (allowing overtaking). To make a real simulation we also have sense in the lanes, so the most congested areas in Luxembourg city center are reflected in our model. There are much more dense areas than others.

In this environment the diffusion process is more difficult than in the case of using random waypoint, since movement is more restricted, but for sure, the simulation is much more realistic. Our main goal is to get results as close as possible to a test on real devices moving along roads in a city center.

For simulating these scenarios we are using JANE simulator [13], [14]. The main feature of JANE is its three steps development; it facilitates the evaluation and minimizes the effort needed for software development in MANETs, so that the evaluated code in the simulated environment can be directly executed in real scenarios without modifications.

V. Simulation Results

To compare these protocols we are simulating them with the mobility model explained in section IV. We want to compare several aspects of their behavior in a VANET. These issues are:

- the broadcast coverage: that means, the number of devices that finally receive the message;
- the message complexity: the number of broadcast and unicast needed to spread the message in the network;
- the broadcast time: the time needed for the diffusion of the message;
- the bandwidth used: the use of the network;

We are comparing the protocols in two scenarios: (1) the first one is a dense network with 1000 devices, and (2) the second one is not so dense, 500 devices (hereinafter sparse network in our context).

As we explained before, the speed of the devices varies between 30 and 75 km/h. As we are dealing with car PCs, the coverage range of the devices is around 100 m.

In such scenarios we needed a training process for tuning and adjusting DFCN parameters. We tried to maximize the number of the devices reached when we tuned the parameters. As a result, we are using *minBenefit*=0.4 and *RAD*∈[0 6] seconds, for both scenarios, and the threshold regarding the density is the only one changing between the two cases: we set *densityThreshold*=15 for the dense network and *densityThreshold*=12 for the sparse one.

We also include in the comparison the well known Simple Flooding (SF) broadcasting protocol [15], [16]. It is the most intuitive idea for disseminating a message in a network. It does not try to reduce the number of re-emissions, so it does not need any knowledge about the neighborhood. The strategy of this algorithm is quite simple, when a device receives a message, it will send it only once. This algorithm was included in our comparison in order to show the validity of the other compared protocols, since Simple Flooding reaches in really short time all devices in a partition, but it is not efficient at all regarding the bandwidth. Additionally, Simple Flooding is not working properly in partitioned networks, since it is not able to spread the message outside the partition where the source node is.

In our experiments, we disseminated a message every 30 seconds during a period of 10 minutes (that means we made 20 broadcasting processes starting from the same device, but from different positions since it was moving). This was simulated in 30 different topologies to make sure our results were realiable.

Table I presents the average of the results obtained regarding the number of devices receiving the message, the number of forwarded messages needed in each dissemination and the ratio between them for the 30 different topologies.

TABLE I
Results obtained for both dense and sparse networks.

		Devices	Broadcasts sent	Ratio
Dense	**BODYF**	$\mathbf{608.9 \pm 30.31}$	312.13 ± 9.78	0.51
	DFCN	395.23 ± 166.92	$\mathbf{194.185 \pm 86.29}$	**0.49**
	SF	292.54 ± 115.86	292.59 ± 115.86	1
Sparse	**BODYF**	$\mathbf{134.81 \pm 29.11}$	82.84 ± 15.90	0.62
	DFCN	123.71 ± 55.58	$\mathbf{68.29 \pm 32.25}$	**0.55**
	SF	72.44 ± 31.60	72.49 ± 31.60	1

As Table I shows, the area covered by BODYF (in terms of devices receiving the message) is considerably higher than the one achieved by DFCN or SF, in dense and sparse networks. The number of forwarded messages in BODYF is always higher, but also the devices reached, so we calculated the ratio between forwarded messages and covered area to make a fair comparison of the algorithms. The ratio is calculated as the number of broadcast per device reached. Simple Flooding is the one with less coverage and higher ratio. We can see that the number of messages sent by DFCN per device reached is slightly lower than in the case of BODYF.

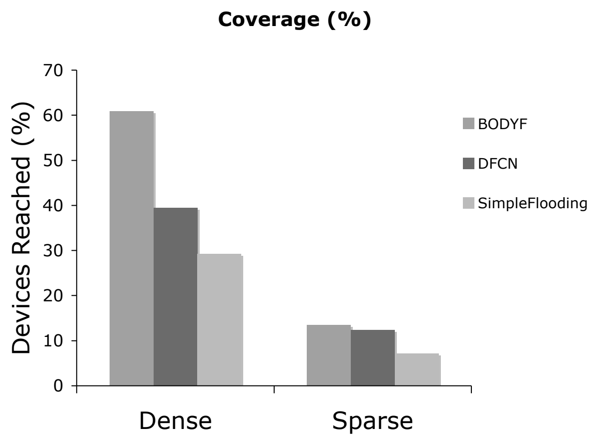


Fig. 5. Coverage percentage in both dense and sparse networks.

In Figure 5, we show the percentage of devices receiving the message, what gives an idea of the total area covered. As it can be seen, BODYF clearly outperforms DFCN and Simple Flooding in the number of devices reached by the broadcasting process.

This is an important feature for VANETs, since there are many applications demanding large coverage of devices at a reasonable cost, like emergency or traffic jam notifications. One of the main reasons for minimizing the number of forwarded messages in MANETs is the battery consumption in processing and re-sending a message. In our work, we are dealing with VANETs, where the optimization of the battery use is not necessary so that all devices have power supply. Additionally, the difference between DFCN and BODYF in terms of this ratio is quite low, therefore we can say that BODYF makes a reasonable use of the network resources.

We should also take into account the network overload in our comparison. DFCN inserts all the neighbors of the device which is forwarding the message. This list is composed by addresses, each address is an IPv6 address, so it is 16 bytes. In our simulations, we obtained that the average of the number of neighbors of each forwarding node was 28.70 and 15.92 in dense and sparse networks respectively. Therefore, we were able to calculate the average of the overload related to this list in all the broadcast process, for both dense and also sparse networks:

1) Dense network
   - The extra bandwith needed for sending the list of neighbors in the message is **89.18 Kbytes** (approx. 713.46 Kbits).
2) Sparse network
   - The extra bandwith needed for sending the list of neighbors in the message is **17.39 Kbytes** (approx. 139.15 Kbits).

We can compare the total bandwidth the broadcast process uses. For BODYF and Simple Flooding, as they do not add any load to the message, we just need to calculate the number of bytes of the broadcast messages sent (the header size is 40 bytes). DFCN includes the list of the neighbors the sender node has, so, apart from the number of messages sent (with headers), we need to sum the load this list adds. The results are shown in Table II. As we can see, even needing more forwarded messages to disseminate the information, BODYF use less bandwidth than DFCN regarding the data of the message. SF is the one with less bandwidth usage, but it is because of the low rate of coverage. The difference between Simple Flooding and BODYF in terms of overload is very low, even when the coverage achieved of the latter is much higher. Between the two protocols with higher coverage, DFCN sends less forwarding messages than BODYF, but due to the list of neighbors that DFCN includes in the message, the total bandwidth used by BODYF is lower.

TABLE II

BANDWIDTH USED BY THE BROADCAST PROCESS.

	BODYF	DFCN	Simple Flooding
Dense	$12.49KBytes$	$96.95KBytes$	$\mathbf{11.70KBytes}$
Sparse	$3.31KBytes$	$20.12KBytes$	$\mathbf{2.90KBytes}$

In BODYF, we do not need to add any extra load to the broadcast message. Even though it is out of the scope of this paper, since we suppose the tree-based topology is already needed by the network, and not only for our broadcasting; we here also calculate how expensive is, in terms of messages sent, the creation and maintenance of the tree, and the circulation of the token. The messages exchanged for building and maintaining the tree-based topology are similar to ACKs, while we use a 40 bytes header for each exchanged message in our estimations. We have measured the number of broadcast and unicast needed for creating and maintaining the network and also for circulating the token. We made this simulation in both networks, and the results obtained were:

1) Dense network
   - bandwidth used for the creation and maintenance of the tree, and also for the circulation of the token: **48.72KBps** (approx. 389.82 Kbits/second).
2) Sparse network
   - bandwidth used for the creation and maintenance of the tree, and also for the circulation of the token: **23.93 KBps** (approx. 191.45 Kbits/second).

The amount of data used by the tree-based topology for a high dense network of 1000 devices is 389.82 Kbits/second. We can make the average per device also, and it would be around 389.82 bits/second per device in dense networks or 382.9 bits/second for every device in sparse ones, what is not a high overload, since the tree-based topology provides many advantages.

Finally, we also compared the algorithms in terms of broadcast time. For that, we did 30 simulations with different topologies where only one broadcast was sent. In each topology we let the network evolve for 60 seconds, and after that spread the message, each time from a different point of the network (different devices). For each topology we start the broadcast from 10 different devices. We consider the broadcast is finished if no more messages are fowarded after 7 seconds. We chose 7 seconds since the delay set for each device (in DFCN) oscilates between 0 and 6 seconds. We did that for the three protocols with both network densities. The results are shown in Table III, where we specify the average time, the number of devices reached in each broadcast process and the ratio between these two values. Notice that, in this table we are testing 30 topologies and starting the dissemination process from 10 different devices for each topology, so the results in Table III are the average values after 300 simulations. Regarding the time needed for spreading a message, we must take into account that BODYF takes longer, but the difference in devices reached is important. We calculated the ratio between the number of devices reached and the time the dissemination took (devices/s), and in both networks, SF had a higher rate, since it is extremely fast. DFCN has the worst ratio in both densities. Simple Flooding is much faster than BODYF but the coverage reached is much lower. BODYF nearly achieved double size of devices than SF in dense networks.

TABLE III

TIME NEEDED FOR DISSEMINATING A MESSAGE .

		Time (s)	Devices	Ratio
Dense	BODYF	90.43 ± 18.85	$\mathbf{479.90 \pm 107.78}$	5.30
	DFCN	27.97 ± 6.45	93.83 ± 31.02	3.35
	SF	$\mathbf{8 \pm 0}$	245.83 ± 125.98	$\mathbf{30.73}$
Sparse	BODYF	23.56 ± 5.68	$\mathbf{37.01 \pm 13.06}$	1.57
	DFCN	17.19 ± 2.83	24.13 ± 6.56	1.40
	SF	$\mathbf{8 \pm 0}$	34.31 ± 9.39	$\mathbf{4.28}$

One of the main problems of DFCN is the difficulty to tune all its parameters. In our case, once we tuned the parameters, if we changed something in the simulation, the number of devices reached was highly reduced. This behavior is shown in the average of devices in both tests, since there is a really big difference between them (coverage and time tests) being the scenario, topologies, speed, number of devices, and thresholds the same in both experiments. So we can conclude DFCN is highly dependent on the topology, while this is clearly not the case of BODYF or SF.

VI. Conclusions

In this paper we present a broadcasting protocol over a tree-based topology, BODYF, and we compare its performance versus DFCN and Simple Flooding. DFCN is a neighbor based topology protocol which is fast, designed to minimize the resources required, and generally accepted by the community [1], [10], [17]. Simple Flooding is one of the bases of broadcasting protocols [15], [16].

In this work, we have compared these protocols in a realistic scenario (a vehicular ad hoc network) with two different number of devices. Our results show that, the coverage achieved by BODYF is much higher than DFCN or Simple Flooding in both scenarios, sparse and dense. Although the number of messages used for achieving this coverage is higher, the difference between DFCN and BODYF in terms of the ratio was very small, hence, we concluded that BODYF makes reasonable use of the network resources. The low level of coverage achieved by Simple Flooding, is due to the network partitions. This protocol is not able to disseminate the message outside the partition where the source node is, while DFCN and BODYF are. Regarding the broadcast time, we checked that Simple Flooding is the fastest protocol, but the coverage is lower than BODYF. BODYF also outperformed DFCN, achieving higher rate of devices per second.

Finally, we can conclude that if the main goal of the application is to maximize the covered area we should use BODYF, but if the network resources are very limited or having a high coverage is not really necessary (e.g., using a hybrid network, we just want to spread a message up to the closest hotspot), DFCN could be slightly more suitable. However, in the VANET scenario considered here, network resources are not as limited as in the case of MANETs, in which devices have, for instance, a limited battery life.

As future work, we plan to compare BODYF to other source-tree based and cluster-based protocols to really know how good its behavior is versus other kind of approaches. We would also like to run other algorithms on top of our tree-based topology in such highly mobile networks, as it could be to propagate a large document, even downloading parts of this information from different devices if it is needed.

References

[1] E. Alba, B. Dorronsoro, F. Luna, A. J. Nebro, P. Bouvry, and L. Hogie, "A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan manets," *Computer Communications*, vol. 30, no. 4, pp. 685–697, 2007.

[2] "Carlink project website. http://carlink.lcc.uma.es."

[3] Y. Yi, M. Gerla, and T. J. Kwon, "Efficient flooding in ad hoc networks: a comparative performance study," in *IEEE International Conference on Communications (ICC'03)*, 2003, pp. 1059–1063.

[4] A. Jüttner and A. Magi, "Tree based broadcast in ad hoc networks," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 753–762, 2005.

[5] A. Casteis and S. Chamuette, "Dynamicity aware graph relabeling system - a local computation model to describe manet algorithms," in *Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and System*, 2005, pp. 198–204.

[6] E. Sopena and Y. Metivier, "Graph relabeling systems: a general overview," *Computers and Artificial Intelligence*, vol. 16, no. 2, pp. 167–185, 1997.

[7] A. Casteis, "Model driven capabilities of the da-grs model," in *Intl. Conf. on Autonomic and Autonomous Systems (ICAS'06). San Francisco. USA*, 2006, p. 24.

[8] S. J. Yang, H. K. Oh, and S. H. Park, "Efficient multicast routing protocol for mobile hosts in IPv6 based networks," *Electronic Letters*, vol. 38, no. 16, pp. 936–938, 2002.

[9] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, "A Bandwidth-Efficient Broadcasting Protocol for Mobile Multi-hop Ad hoc Networks," in *Demo proceeding of the 5th International Conference on Networking (ICN'06)*. IEEE, October 2006, p. 71.

[10] L. Hogie, F. Guinand, and P. Bouvry, "A heuristic for efficient broadcasting in the metropolitan ad hoc network," in *8th Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems*, 2004, pp. 727–733.

[11] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*, ser. OR/CS Interfaces. Springer-Verlag, 2008.

[12] A. Andronache and S. Rothkugel, "Hytrace-backbone-assisted path discovery in hybrid networks," in *The Seventh International Conference on Networking ICN*, to appear.

[13] D. Gorgen, H. Frey, and C. Hiedels, "JANE–the Java Ad-hoc Network Environment," in *Proceedings of the 40th Annual Simulation Symposium*, 2007, pp. 163–176.

[14] "The Java Ad-hoc Network Environment (JANE). http://syssoft.uni-trier.de/jane/index.php/JANE_Basics."

[15] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2002, pp. 194–205.

[16] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 1999, p. 151162.

[17] L. Hogie, "Mobile ad hoc networks, modelling, simulation and broadcast-based applications," Ph.D. dissertation, University of Luxembourg, Luxembourg, 2007.

OPTIMIZING DISTRIBUTED COLLABORATIVE FILTERING IN MOBILE NETWORKS

Patrick Gratz and Adrian Andronache
University of Luxembourg
Faculty of Science, Technology and Communication (FSTC)
L-1359 Luxembourg
E-mail:{patrick.gratz, adrian.andronache }@uni.lu

KEYWORDS

Mobile networks, clustering, distributed, collaborative filtering.

ABSTRACT

Collaborative filtering mechanisms filter information by using collaboration among multiple data sources, typically involving large data sets. In this work we optimize the communication and computational load of a distributed collaborative filtering protocol designed to augment the information exchange in mobile networks.

INTRODUCTION

Recommender systems using collaborative filtering (CF) are a popular technique for information overload reduction and desired data discovery on the Internet. To achieve this, the user is provided by the collaborative filtering system with recommendations or predictions on data items based on the opinions of other *like-minded* users (Sarwar et al. 2001). The opinions on the data items can be obtained explicitly from users by rating or implicitly by counting clicks, viewing time, and alike.
Considering the increase in popularity of mobile devices like smart phones, PDAs and Tablet-PCs the problem of information overload also emerges in mobile networks. This devices are becoming faster in processing, gain more memory capacities, are able to execute more and more powerful applications and at the same time being equipped with various wireless and/or cellular communication capabilities.
In a previous work we introduced an ad hoc collaborative filtering mechanism designed to augment the information exchange in mobile hybrid networks (Gratz et al. 2008). The introduced application was designed for residents and tourists of big cities where many people—potential mobile device users—meet at train or metro stations, at universities and schools, in shopping centers, restaurants, and pubs. In this scenario the shopping centers of the city provide podcasts about offers of the day that contain multimedia files showing the products. The restaurants are podcasting the menu of the day also containing multimedia about the food, drinks and location. The podcasts of the theatres are containing information about tonight's shows and highlights and the podcasts of the discotheques are informing about events and party mottos presenting the DJ's and the music that will be played. The mobile device users in the city can subscribe to the podcasts of the favorite restaurants, theatres, shopping centers etc., thus being up to date about current offers and events. They can subscribe and download the podcasts on computers with broadband Internet connection and download them onto their mobile device, which is the current common procedure. In contrast to this, our application aims to augment the podcast providing mechanism by enabling the mobile devices to exchange episodes of subscribed podcasts, to recommend similar, well-rated podcasts and to subscribe to new ones in an ad-hoc mobile environment. Additionally, devices with a cellular network interface are able to pull new podcast episodes from the Internet servers and share them locally within ad-hoc networks.
Imagine a traveling user meeting other people in metros, trains, shopping centers, pubs or restaurants. Some of them might have recently downloaded new podcast episodes either via cellular uplinks or by copying them from desktop computers onto their mobile device. The client on the traveler's device will group up with the subscribers of similar podcasts and exchange new or missing episodes. Thus, the traveler will be able to receive podcasts on the way without having to use an Internet connection.
The application can be used for instance by a tourist which arrives in a big city and does not have knowledge about the local podcast providers. He will meet other tourists and residents with mobile devices at the airport, train or metro station, shopping centers, pubs and restaurants. The client on the tourist's mobile device will search the ad-hoc network for podcasts and provide the information to its user. Thus, the tourist is able to choose among the local podcasts about restaurants that offer the favorite food, locations and events he may like or shopping offers that can be interesting for him. By subscribing to the podcasts of interest found in the mobile ad-hoc network, the tourist will receive up to date information about offers and events in the city during the stay.
The collaborative filtering mechanism introduced in (Gratz et al. 2008) enables the user to get recommendations about the local offers based on the taste of other like-minded users in nearby mobile environments. The ability to get recommendations from the local ad-hoc network has the advantage that the

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

system can take into account new information without to require a connection to a central repository on the Internet. Thus, the mechanism provides updated recommendations even if the user has no backbone connection by a cellular network or access point. Furthermore, all ratings made by a user on the way, e.g. for the menu of the day in a restaurant will instantly have an impact on the calculated recommendations for like-minded users in the local vicinity.

In (Gratz et al. 2008) two algorithms to determine sets of similar neighbors were introduced. The first one called Hierarchical Cluster-based Neighborhood Resolution (HCNR) uses a weighted cluster topology generated by the Weighted Application Aware Clustering Algorithm (WACA) presented in (Andronache et al. 2006). The second algorithm—Weighted Neighborhood Resolution Algorithm (WNR)—is based on a simple peer-to-peer communication pattern. In order to compare the introduced algorithms for proper operation they have been implemented and tested on top of the JANE simulator by performing several experiments. As the results of these experiments show, HCNR provides a slightly better precision and scales distinctly better concerning the number of sent unicasts if we increase the network connectivity. However, WNR has the advantage that the used bandwidth is very small compared to HCNR.

In this paper we aim to optimize the communication and computational load of the distributed, cluster-based algorithm HCNR by using another, better suited clustering protocol.

The remainder of the paper is structured like follows: the next section presents related work. Section III briefly introduces the Hierarchical Cluster-based Neighborhood Resolution (HCNR) algorithm. In Section IV the clustering algorithm is described which is employed to optimize HCNR. Section V describes the simulation setups and Section VI presents the results. The paper concludes with future work in Section VII.

RELATED WORK

Recommender systems using collaborative filtering are one of the most successful recommendation techniques (Resnick et al. 1994; Shardanand and Maes 1995). In this section we introduce some existing research work about collaborative filtering in mobile environments.

In (Cöster and Svensson 2005) an incremental collaborative filtering algorithm for applications, where users are occasionally connected to a central server is introduced. The general idea is to store a subset of selected user profiles, together with a ranked list of predictions. When the user is in offline mode, a service on the local device can still recommend items based on the predictions made the last time the user was connected. Each time the user supplies new ratings, the list of predictions will be recomputed, even if the user is not connected to the server. In the case that a user encounters another user, the authors suggest that they exchange their profiles and recalculate their prediction lists. The past influence of the other user should be removed from all predictions and the new influences should be added. At last this case is not evaluated or considered any further in the paper and is a part of future work.

A further portable recommender system along with five peer-to-peer architectures for finding neighbors is presented in (Miller et al. 2004). The authors introduce a new collaborative filtering algorithm called PocketLens that can run on connected servers, on usually connected workstations or occasionally connected portable devices.

The presented algorithm is a variant of the item-item algorithm introduced in (Cöster and Svensson 2005) with modifications for a peer-to-peer environment. To reach the goal of portability a local similarity model is created for the user. Thereby, the algorithm only needs access to the ratings of the owner and one other user at a time. In this manner, the model is created incrementally in a distributed fashion.

In (Tveid 2001) an approach for making a scalable recommendation system for mobile commerce using P2P is considered. The main idea of the proposed approach is to transform the problem of finding recommendations using collaborative filtering, into a search problem in scalable P2P systems like Freenet or Gnutella. Thereby, a query (vector with votes on products) is broadcasted from the querying node to all neighbor peers. When a peer receives a query it calculates the proximity with other cached queries. If the proximity is higher than a threshold, the cached voting vector is sent back otherwise the query is broadcasted further. For sparse voting vectors the authors propose a binary interpolative compression algorithm. Furthermore, to improve the performance and quality of recommendations they propose an approach for clustering similar peers.

An approach to collaborative filtering in a mobile tourist information system for visitors of a festival based on spatio-temporal proximity in social contexts is proposed in (Spindler et al. 2006). This new approach is based on the idea that users who go to the same place at the same time tend to have similar tastes. In order to keep track about the visited places each user is equipped with a portable computer coupled with a GPS unit. Furthermore, a central server provides a database with information about all the events, restaurants, venues and bars at the festival (Belotti et al. 2005).

The proposed approach uses a user-based CF technique and calculates similar users via a spatio-temporal proximity measure, i.e. two users are considered as similar if they consume the same items simultaneously. The following exchange of rating information between such similar users is done via an ad-hoc peer-to-peer interaction. However, the defined similarity measure has one drawback. Users consuming the same periodic event at different times still share interests, but are not considered as similar. In a future work, the authors intend to investigate how their CF approach can be extended in order to exchange ratings between users in spatial but not temporal proximity. Furthermore, they want to evaluate the introduced CF system at the Edinburgh Fringe festival.

(Jacobsson et al. 2006) introduce an approach for a mobile recommender system where media can find people rather than the other way around. Whereas, media files are autonomous, rule-following agents capable of building their own identities from interactions with other agents and users. The general idea is that the interaction of large ensembles of those interacting agents, distributed over mobile devices in social networks can emerge a collaborative filtering-like behavior.

HIERARCHICAL CLUSTER-BASED NEIGHBORHOOD RESOLUTION (HCNR)

The collaborative filtering recommender process can be roughly divided into three phases: determine similar neighborhood, update the recommender model, and calculate a prediction.

In order to deliver good recommendations a typical collaborative filtering system depends on a critical mass of users with commonly rated items. However, in our application scenario it is very likely that a tourist who visits a certain city for the first time has no commonly rated podcasts with users in his nearby environment. Nevertheless, this fact does not except that the tourist has no similar taste with other users in the local neighborhood. The tourist can have rated different podcasts that are similar concerning the content of those in the nearby neighborhood. For this purpose we calculated the similarity between two users based on an approach proposed by (Pazzani 1999) called collaboration via content. The idea behind this approach is to exploit a content-based profile for each user in order to calculate the similarity between two users via their content-based profiles instead of their commonly rated items. In the context of our application scenario the content-based profile is represented by a list of weighted keywords. For this purpose we presume that each podcast feed contains a set of keywords describing its content. Given the corresponding rating values $r(p)$ for all podcasts P_i rated by a specified user u_i together with the appropriate set of keywords K_i describing these podcasts, a weight for each keyword $k_j \in K_i$ can be calculated as follows:

$$w_j = \frac{\sum_p r(p)}{occur(k_j)}, \qquad p \in P_i \wedge p \ni k_j$$

The function $occur(k_j)$ returns the number of podcasts containing keyword k_j. Thus, for each user u_i a vector of weighted keywords $wk_i = \langle (k_{i1}, w_{i1}), \ldots, (k_{in}, w_{in}) \rangle$ can be calculated, that represent his preferences. Given these content based profiles we can define the similarity between two users u_i, u_j via the cosine between the corresponding weighted keyword vectors:

$$sim(u_i, u_j) = \frac{\sum_{k \in K_{ij}} w_{ik} w_{jk}}{\sqrt{\sum_{k \in K_{ij}} w_{ik}^2}\sqrt{\sum_{k \in K_{ij}} w_{jk}^2}}, \; K_{ij} = K_i \cup K_j$$

The HCNR algorithm uses a weighted cluster topology generated by Weighted Application Aware Clustering Algorithm (WACA) (Andronache et al. 2006). WACA creates clusters in a hierarchical fashion. Each device elects exactly one device as its clusterhead, i.e. the neighbor with the highest weight. This clusterhead also investigates its one-hop neighborhood, similarly electing the device with the highest weight as its clusterhead. This process terminates in case of a device electing itself as its own clusterhead, due to the fact of having the highest weight among all its neighbors. We call all intermediary devices along such clusterhead chains sub-heads. Each device on top of a chain is called a full clusterhead, or, in short, clusterhead (Figure 1).

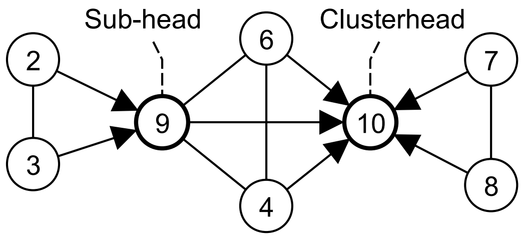


Figure 1: Topology built by WACA

Based on the WACA topology, the algorithm works as follows. At first, in order to determine a set of similar neighbors, each slave sends its own profile to the currently elected clusterhead. To keep the message complexity low, each device maintains a list of cluster-heads that have already received the current profile. As soon as the own profile changes this list will be cleared. After receiving the profiles from its slaves, the clusterhead and all sub-heads calculate a similarity matrix via the received profiles. Subsequent to this calculation, the sub-head sends the calculated similarity matrix and the list of received profiles to the cluster-head. The cluster-head stores this profile list together with the corresponding similarity values and calculates the similarity values for all missing pairs in order to complete the similarity matrix. Figure 2 shows how the algorithm is using the cluster topology to exchange the information.

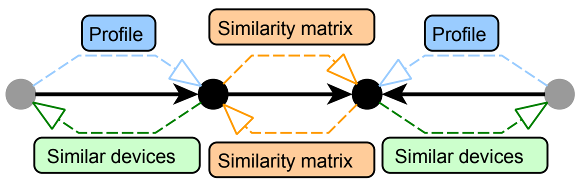


Figure 2: HCNR similarity matrix calculation and similar neighbor discovery

By doing so, the matrix on the clusterhead stores the similarities between all users connected to the current cluster. Finally, in order to determine the similar neighborhood for all slaves that are not directly connected with the clusterhead a copy of this similarity matrix is replicated to each sub-head in the cluster. The cluster-heads and the sub-heads provide a list of similar neighbors to each slave in the current cluster.

HCNR OPTIMIZATION

A drawback of WACA is the fact that the algorithm provides no mechanism to avoid the re-organization of two crossing clusters. One can imagine a group of mobile device users traveling by bus. The applications on the mobile devices use WACA to build clusters. On the way, the clusters may meet other clusters build for instance by devices in another bus waiting at the same traffic light. In this scenario WACA re-organizes the crossing clusters, which is needless since the devices build the initial clusters after the busses pass each other.

The HCNR algorithm works on top of a clustered mobile network, thus the robustness of the topology has a crucial impact on the communication and computational complexity of the collaborative filtering protocol.

In (Andronache et al. 2008) the Node and Link Weighted Clustering Algorithm—NLWCA—was introduced. The algorithm is designed to protect stable clusters from re-organization in order to avoid needless network communication.

Unlike WACA, which uses only the weight of the nodes to elect a local clusterheads, NLWCA also assigns weights to the links between the own node and the network neighbor nodes. This weight is used to keep track of the connection stability to the one-hop network neighbors. The algorithm increases the weight of the links to neighbors that are for a longer time in communication range. When a link weight reaches a given stability threshold the link is considered stable and the device is called stable neighbor device. The clusterhead is elected only from the set of stable neighbors which avoids the re-organization of the topology when two clusters are crossing for a short period of time (Figure 3).

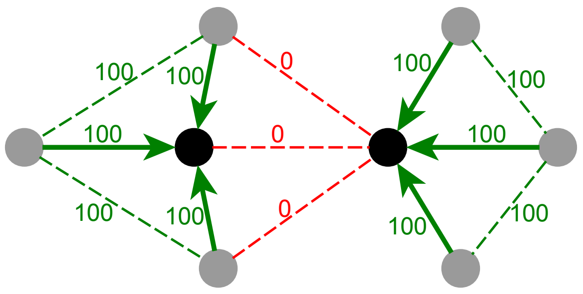


Figure 3. The low weight of the links avoids superfluous re-organization of the topology when for instance two clusters are cross in mobile networks.

The simulation results in Figure 4 and 5 show that NLWCA outperforms WACA in terms of number of re-elections independent of the used speed and node number. More simulation results and detailed results can be found in (Andronache et al. 2008).

Motivated by the good results in terms of topology stability obtained by the NLWCA, the algorithm was employed to optimize the communication of the HCNR protocol.

SIMULATIONS SETUP

In order to observe if the network and computational load of the HCNR protocol is improved on top of the cluster topology build by NLWCA, we performed several simulation runs using the JANE simulator (Görgen et al. 2007).

For our experiments, we used the MovieLens Data Set (available at: http://www.grouplens.org) that consists of 100,000 ratings for 1682 movies by 943 users, where each user has at least rated 20 movies.

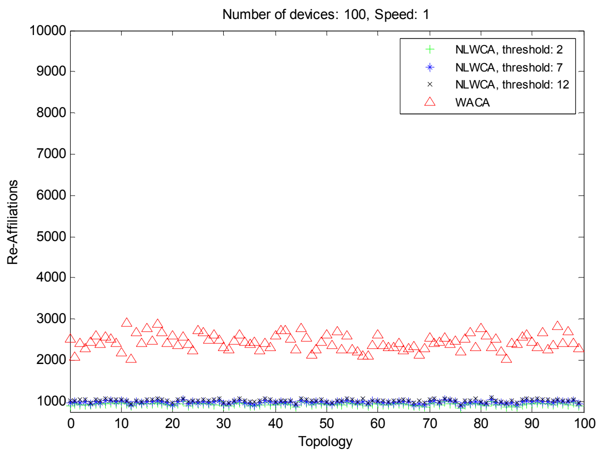


Figure 4: Results for 100 mobile device moving with 1m/s during 5 minutes on an area of 300x300m with sending range of 40m. NLWCA outperforms WACA in terms of re-elections number independent from the used stability threshold.

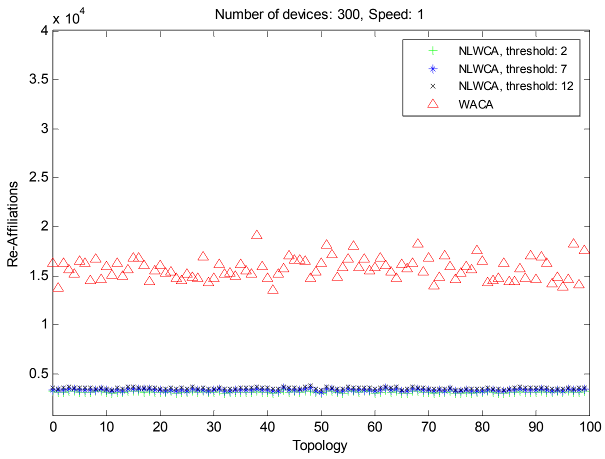


Figure 5: Results for 300 mobile devices using the same settings as above. Also in the dense networks, NLWCA outperforms WACA in terms of re-affiliation of nodes to new clusterheads.

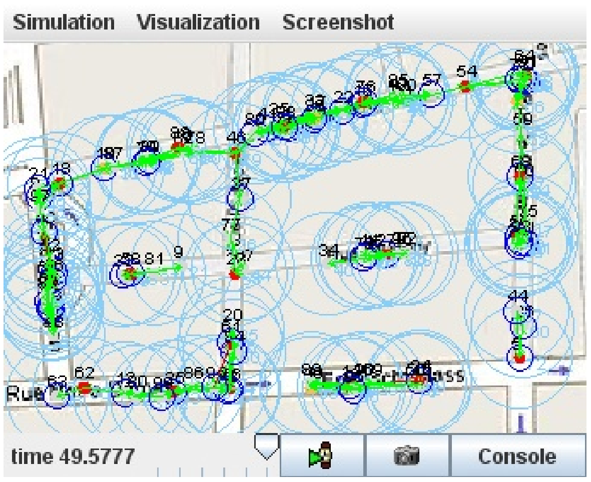


Figure 6: JANE simulating HCNR on top of NLWCA running on 100 devices. The mobile devices move on the streets of the Luxembourg City map. The devices move with a speed of 0.5 – 1.5 m/s (1,8 – 5,4 km/h)

We generated 5 different training sets containing 15 votes that are considered as the observed votes for each user and 5 different test sets containing the remaining votes. After each simulation run we compare the predicted votes with the corresponding votes in the test set and calculated the Mean Absolute Error (MAE), which has been used to measure prediction performance in several cases (Shardandand and Maes 1995; Breese et al. 1998; Herlocker et al. 1999). If a predicted item did not have an adequate entry in the test set it was eliminated from the evaluation. Note that we used the MAE only to compare how accurately our algorithms predict a randomly selected item rather than evaluating the user experience of generated recommendations.

For each experiment we used the Restricted Random Way Point mobility model (Blazevic et al. 2002), whereby the devices move along defined streets on the map of Luxembourg city for 5 minutes and 30 seconds (Figure 6). For each device the speed was randomly varied between [0.5;1.5] units/s in a first run and [11;16] units/s in a second run. While, every time a device reaches a crossroad, it randomly selects a street to turn in at next.

At startup, the devices are positioned at random selected crossroads and initialized with 15 selected votes in order to calculate an initial user profile. In order to avoid a data exchange at this point, where the devices are already strongly clustered at the crossroads, we delay the startup of our HCNR algorithm via a timeout of 30 seconds. After this timeout the devices begin to exchange their profiles in order to determine the k-most similar neighbors. For all experiments we simulated with 5 different training sets and 5 different topologies per training set. Furthermore, we selected for NLWCA a stability threshold of 2 for speed [0.5;1.5] and 12 for speed [11;16]. All results are shown with 95% confidence intervals.

RESULTS

Figure 7 shows the overall computed similarities after 5 minutes of simulation. As the figure shows, using NLWCA distinctly reduces the number of computed similarities in dense network settings. In particular, if the devices move with a speed between 11 and 16 units/s WACA produces nearly 3 times more similarity calculations as NLWCA. Due to the fact, that NLWCA produces provable lesser re-elections than WACA this result was expected. Since, each time a new cluster-head is elected this cluster-head potentially has to calculate similarities again.

In Figure 8, we illustrate the number of sent unicasts during the simulation. Again the use of NLWCA reduces overhead. Thus, for each setting HCNR using NLWCA needs constantly lesser unicasts than using WACA. However, more considerable is the diversity of the used bandwidth. As shown in Figure 9, in dense network settings WACA used nearly up to 2 times more bandwidth at a device speed between 0.5 – 1.5 units/s and about 7 times more bandwidth if the device move with a speed between 11 – 16 units/s.

However, all these communication and computational optimization comes at a cost.

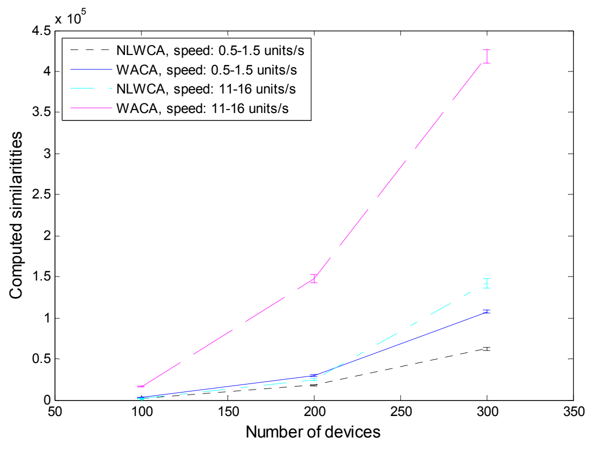


Figure 7: Overall computed similarities

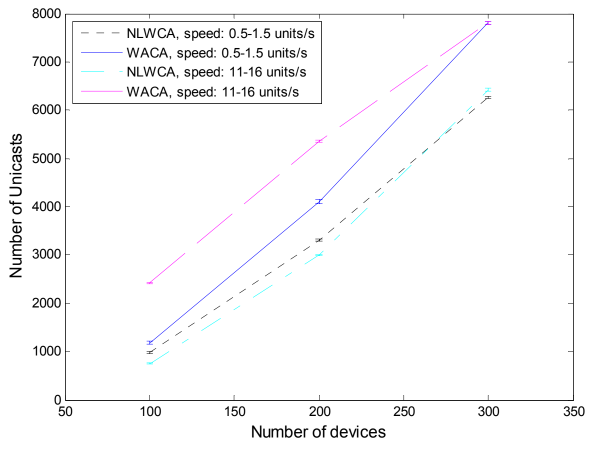


Figure 8: Number of sent messages

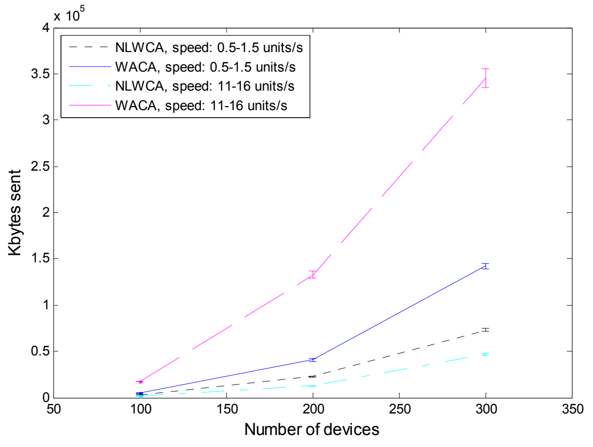


Figure 9: Bandwidth usage

As Figure 10 shows, the MAE of the calculated predictions, with 300 mobile devices is about 1.8% respectively 6.4 % better when using WACA concerning the two different speed intervals. Again, this result was expected due to the fact, that NLWCA allows only communication between devices building stable clusters based on a defined stability threshold.

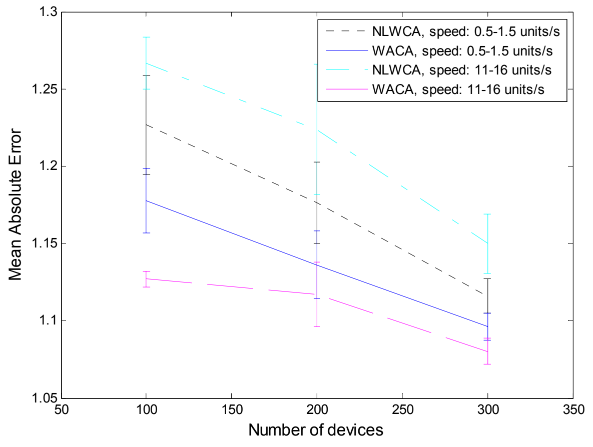


Figure 10: Mean absolute error of calculated predictions

Thus, crossing devices, which are only in the communication range for a short period of time, will be not part of the cluster and hence, do not participate on a profile exchange. Since, the precision of calculated predictions also depends on the number of discovered similar neighbors, it was expected that the precision of the calculated predictions will be inferior when using NLWCA. Nevertheless, using NLWCA instead of WACA significantly reduces the overhead concerning communication and computational load at the cost of a relatively small loss in precision.

CONCLUSION AND FUTURE WORK

In this work we optimized HCNR in order to reduce the communication and computational overhead of the collaborative filtering based recommender system, which was designed to overcome the potential problem of information overload in mobile ad hoc networks.

To achieve this goal we employed a network link stability aware clustering protocol, which provides HCNR with a better suited topology than the previous used mechanism WACA.

Several simulations were done using the JANE simulator and the results show that the new mechanism significant improves the communication and computational load on the network nodes. However, the prediction precision is slightly lowered since the devices communicate only with the set of neighbors that are considered to be stable connected.

In order to improve the prediction precision, in future work we will employ an inter-cluster communication protocol, thus enabling an inter-cluster similarity matrix exchange.

REFERENCES

Andronache, A., M.R. Brust and S. Rothkugel. 2006. "Multimedia Content Distribution in Hybrid Wireless Networks using Weighted Clustering". In *Proceedings of the 2nd ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP'06)*, Torromolinos, Spain, 1-19.

Andronache, A., M.R. Brust and S. Rothkugel. 2008. "NLWCA—Node and Link Weighted Clustering Algorithm for Backbone-assisted Mobile Ad Hoc Networks". *The Seventh International Conference on Networking (ICN)*, Cancun, Mexico, 460-467.

Belotti, R., C. Decurtins, M.C. Norrie, B. Signer, and L. Vukelja. 2005. "Experimental Platform for Mobile Information Systems". In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom 2005)*, Cologne, Germany, 258-269.

Blazevic, L., S. Giordano, J. Y. Le Boudec. 2002. "Self Organized Terminode Routing". In *Cluster Computing Vol.5, No. 2,* Springer Netherlands, 205-218.

Breese, J.S., D. Heckerman and C. Kadie. 1998. "Empirical analysis of predictive algorithms for collaborative filtering". In *Proceedings of the 14th Conference on Uncertainty in Artifical Intelligence (UAI-98)*, 43-52.

Cöster, R. and M. Svensson. 2005. "Incremental Collaborative Filtering for Mobile Devices". In *Proceedings of the 2005 ACM symposium on Applied Computing* (*SAC'05),* Santa Fe, New Mexico, USA, 1102-1106.

Görgen, D., H. Frey and C. Hiedels. 2007. "JANE-The Java Ad Hoc Network Development Environment". In *40th Annual Simulation Symposium (ANSS'07),* 163-176.

Gratz, P., A. Andronache and S. Rothkugel. 2008. "Ad Hoc Collaborative Filtering for Mobile Networks". *The Second IEEE International Workshop on Ad Hoc and Ubiquitous Computing (AHUC),* Taichung, Taiwan.

Herlocker, J. L., J.A. Konstan, A. Borchers, and J. Riedl. 1999. "An algorithmic framework for performing collaborative filtering". In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99),* 230-237.

Jacobsson, M., M. Rost and L.E. Holmquist. 2006. "When Media Gets Wise: Collaborative Filtering with Mobile Media Agents". In *Proceedings of the 11th International Conference on Intelligent User Interfaces* (*IUI'06),* Sydney, Australia, 291-293.

Miller, B.N., J. A. Konstan, and J. Riedl. 2004. "PocketLens: Toward a Personal Recommender System". In *ACM*

Transactions on Information Systems, *Vol. 22, No. 3*, 437-476.

Pazzani, M.J. 1999. "A Framework for Collaborative, Content-Based and Demographic Filtering". In *Artificial Intelligence Review*, 393-408.

Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J.Riedl. 1994. "Grouplens: An open architecture for collaborative filtering of netnews". I*n Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, Chapel Hill, North Carolina: ACM, 175-186.

Sarwar, B., G. Karypis, J. Konstan and J. Riedl. 2001. "Item-based Collaborative Filtering Recommendation Algorithms". In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, Hong Kong, Hong Kong, 285-295.

Shardanand, U. and P. Maes. 1995. "Social information filtering: Algorithms for automating "word of mouth"". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, vol. 1, 210-217.

Spindler, de A., M.C. Norie, M. Grossniklaus and B. Signer. 2006. "Spatio-Temporal Proximity as a Basis for Collaborative Filtering in Mobile Environments". In *Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS 2006),* Luxembourg, Grand Duchy of Luxembourg.

Tveid, A. 2001. "Peer-to-peer based Recommendations for Mobile Commerce". In *Proceedings of the 1st International Workshop on Mobile Commerce* (*WMC 01,* Rome, Italy, 26-29.

Building a Practical Event-Scheduler for a Multi-Processor Architecture

S. Rooney, D. Bauer, L. Garcés-Erice
IBM Zurich Research Laboratory
CH-8803 Rüschlikon - Switzerland
{sro, dnb, lga}@zurich.ibm.com

***Abstract*— We describe a scheduler that processes a high number of typed events per second while enabling certain event types to be allocated more resources than others in a work-conserving fashion. The scheduler is the core of a high volume messaging system, it uses a lock-free approach allowing it to scale with increasing number of processors. The scheduler threads coordinate using a lock-free concurrent priority queue. As far as we know, no formal analysis of this lock-free data structure has been given so far in the literature. We analyze the expected behavior of the scheduler and report on its actual performance on a multiprocessor machine.**

***Index Terms*— Scheduling, Event System, Lock-Free, Middleware**

I. Introduction

Queued event systems detach the production of an event from its handling. Structuring applications through the use of such abstractions allows a looser form of coupling than via function calls. When the queues themselves are lock-free then there is no synchronization at all between event producers and handlers. In a multiprocessor environment this helps in proving the correctness of the program and allows better parallelization. As the operational semantics of placing an event in a remote queue are the same as that of placing it in a local one, they do not suffer the same impedance as remote procedure calls where the distributed system attempts to give the illusion that a remote network invocation has the same semantic as pushing a new frame onto the stack [1]. For these reasons we expect a queued event approach to become the unifying abstraction for communication within distributed middleware, whether the communication is intra or inter machine.

Events are typed by the function used to handle them. As distinct event types may be of different importance to the application it is advantageous if the relative importance of the events can be expressed to the dispatching mechanism such that it can be taken into account when allocating resources to them. The entity that dispatches events in the typed event queues is a task scheduler in which the tasks comprise the event queue and the associated function to execute. The scheduler chooses the next task to schedule based on the importance assigned to the corresponding event type.

We shall use a distributed messaging system built in Java as the context in which we describe the rest of our work. In a messaging system, messages on a named topic are carried from one or more publishers to one or more subscribers. Messages can either be out-going from local applications or in-coming from the network. In the first case, the associated task transmits them over the network; in the latter, it executes the associated application-specific callback.

The selection of the scheduling mechanism is determined by the needs of this messaging system. The arrival distribution of events and their dispatch time are not in general known *a priori*. As a messaging system may be used by different applications in different contexts, the means by which importance is attached to topics should be as general as possible. The scheduler must be work-conserving and parallelizable, allowing it to make best use of the available resources and to scale with increasing numbers of processors. We don't assume any particular support from the underlying OS and assume that thread scheduling is non-preemptive.

In the next section we motivate our choice of a time-based credit scheduler for the messaging system and compare it with other alternatives. We then describe the implementation of the system showing how coordination between scheduling threads can be achieved using an appropriate lock-free data structure. We give the invariant for this data structure and explain why this invariant is strong enough for use within the messaging system. We describe what fairness means for a work-conserving scheduler on a multiprocessor system. Finally, we investigate various aspects of the overall system in particular in relation to throughput, delay, scalability with processors and performance isolation.

II. Description of the Scheduler

At a first glance the scheduling of message queues has strong similarities with algorithms used for giving different service rates to flows within network routers. For example, the Weighted Fair Share (WFS) queuing discipline [2] allows different service rates to be allocated to different queues according to some requested share. WFS was developed in the context of packet-switched networks to protect one network flow from another. It belongs to a set of virtual-time-based algorithms that provide fairness and/or delay bounds. Message scheduling is distinct from packet scheduling in that the time a message takes to be serviced cannot be easily estimated; in a packet scheduler it is a simple function of the packet length. This means that the guarantees given by the message scheduler are by nature less precise.

Our proposal is to use time-based credit-scheduling within the message scheduler, in which resources are partitioned

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

amongst tasks by the allocation of credits. The total credits allocated to a task is a fraction (defined by the *share* given to the task) of the time period *T*. The scheduler always chooses to schedule the task with the highest number of remaining credits that has a non-empty queue, i.e. the scheduler is work-conserving. Each time a task is scheduled, the scheduler measures how long it runs before it completes. This amount is deducted from the total number of credits for that task. We now describe the guarantees that this time-based credit scheduler gives with respect to throughput.

Assume that $0 \leq v_i \leq 1$ is the share allocated to task *i* and that s_i is the number of messages arriving at the task queue per second. The question we wish to address is how many messages r_i will actually be serviced for task *i*. It is important to realize that due to the work-conserving nature of the scheduler, no one task can be viewed completely in isolation, i.e. the amount of messages serviced for any given task share depends on the messages arriving for other tasks at other queues. The number of messages serviced for a task can be viewed as containing two components: that which is guaranteed by the share and any spare capacity that a task can take advantage of. We denote the total maximum achievable rate by R^{max}. R^{max} is a function of the system settings, but for the moment we consider it as a constant.

For example, suppose we have three tasks T_0, T_1 and T_2 with shares $v_0 = 0.5$, $v_1 = 0.3$ and $v_2 = 0.2$, suppose further that we know that $R^{max} = 10{,}000$. We would like to know what values are received for the three tasks if $s_0 = 4{,}000$, $s_1 = 6{,}000$ and $s_2 = 5{,}000$. It is clear that r_0 is 4,000 because task T_0 is sending less than its share. That leaves 6,000 to be shared among T_1 and T_2 . Both T_1 and T_2 will get their guaranteed share (3,000 and 2,000 respectively) plus some fraction of the spare capacity (1,000) that T_0 has reserved but is not using. The unreserved capacity is shared proportionally to the shares v_1 and v_2, so T_1 gets 60% and T_2 gets 40%, meaning that $r_1 = 3{,}600$ and $r_2 = 2{,}400$.

We now give the general formula for calculating r_i for an arbitrary resource allocation between a set of tasks and an arbitrary attempted sending rate on those tasks on a single processor. We consider the effect of multiprocessors in Section III-E. We define u_i to be the fraction of resources that a task is *attempting* to consume and we define x_i to be the ratio of v_i to u_i.

$$u_i = \frac{s_i}{\sum_j s_j}$$

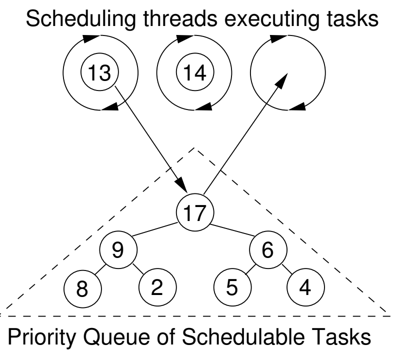
$$x_i = \frac{v_i}{u_i}$$

An x_i value of 1 means that a task is attempting to use exactly what it has reserved. A value greater than 1 means that it has unused capacity and a value less than 1 means that it is trying to use more than its has reserved, i.e. it is trying to take advantage of any unused capacity. For a given setting of v_j, s_j and R^{max} the expected receive rate r_i is:

$$r_i = Min(s_i, v_i . \frac{R^{max} - \sum_{x_j > x_i} r_j}{1 - \sum_{x_j > x_i} v_j})$$

This states that the share that a task will get is its *adjusted* share of what is left over after tasks with higher x_i have been allocated. The amount of additional unused capacity available to a task is that which is left over by tasks which are less speculative than itself. This is equivalent to the Weighted Max-Min fairness allocation developed for the Available Bit Rate (ABR) service within ATM [3] where u_i corresponding to the demand or rate and v_i to the normalized weight.

III. Implementing the scheduler on a multi-core architecture


Scheduling threads executing tasks
13
14
17
9
6
8
2
5
4
Priority Queue of Schedulable Tasks

(a) Scheduler overview

Thread `SchedulingThread`

```
1:  while $true$ do
2:      $t \leftarrow SchedulableTasks.get()$
3:      if $t.credit = 0$ then
4:          $resetAllTaskCredits()$
            /* the Task with the most credits has 0. Period $T$ is over. */
5:          continue $while$
6:      end if
7:      $e \leftarrow t.eventQueue.get()$
8:      $T_o \leftarrow getTime()$
9:      $t.processEvent(e)$
10:     $t.credit \leftarrow t.credit - (getTime() - T_o)$
11:     if $--t.queueSize > 0$ then
12:         $SchedulableTasks.put(t)$
            /* Task $t$ has more events, so it is put back for scheduling */
13:     end if
14: end while
```

(b) Scheduling thread

Fig. 1. Scheduler using multiple scheduling threads.

A task t contains a queue $t.eventQueue$ into which messages or timer events are placed. In addition, it contains control information such as the allocated share and the amount of outstanding credits $t.credit$. A task is ready to run when it has credit and its queue contains at least one event. Tasks are passive data structures; the scheduler employs a number of scheduling threads to execute tasks. Figure 1(b) shows the basic operation of a scheduling thread. The core of the scheduler is a priority queue containing the set of schedulable tasks. Figure 1(a) shows the schema of the interaction in which tasks are ordered by their number of threads in the queue (shown as a binary heap) and scheduling threads add and remove elements from this queue concurrently.

We refer to this priority queue as *SchedulableTasks*. The task with the most credit has the highest priority and is therefore at the head of the priority queue, from where it is removed by a scheduling thread for execution (see Figure 1(b)). Once invoked, the task processes exactly one event. The scheduling thread measures the execution time and uses this to decrease the credit of the task, thus changing its priority. If the highest priority task is out of credit, then no task has credit left and the scheduling thread resets the credits of all tasks, i.e. the scheduler is work-conserving. This situation occurs at the latest when the scheduling period *T* expires; but may occur earlier if some of the tasks had less work than their allocated share.

Using multiple scheduling threads allows the scheduler to run on multiple CPU cores. For efficient operation, it is important that a scheduling thread is not blocked by another. Two contention points exist, the event queue used by each task (see Section III-A) and the priority queue used at the core of the scheduler (see Section III-B).

A. Coordination between writer threads and scheduling threads

The event queue of tasks is implemented as a Michael–Scott [4] lock-free FIFO queue. This queue allows scheduling threads (the readers) and input threads or application threads (the writers) to concurrently access the same queue without the need for locking. Whereas in [4] the size of the queue is not limited, we trivially extend the algorithm such that a writer blocks if the queue reaches its capacity and then is woken up by the next reader. All the information about the credits a task has is maintained within the task itself, and because it is only updated by the scheduler thread that took the task out of *SchedulableTasks*, this information does not need to be thread-safe.

Writer threads and scheduling threads coordinate in deciding whether a task is schedulable or not. A writer thread that adds a new event to a task that is currently not scheduled will add it to *SchedulableTasks*. Likewise, a scheduling thread that finds the event queue of a task empty will remove the task from the set.

Figure 2 shows how this coordination takes place. In Figure 2(a) the writer thread recognizes that a task is not in the schedulable task set and adds it; simultaneously, a scheduler thread accesses the same data structure and removes the tasks with the most credits. In Figure 2(b) the task is scheduled, the writer thread adds events to the event queue while the scheduler thread removes the first one. In Figure 2(c) the scheduler thread puts the task back into the schedulable task set after updating its credits, while the writer thread continues to write. In Figure 2(d) the scheduler thread has emptied the queue and does not return the task to the set; the next time the writer writes it will notice this and perform the action described in Figure 2(a) again.

The scheduler thread removes the task from *Schedulable-Tasks* and returns it if there is still work to do (i.e., an event is in the task's queue). If the scheduler thread detects that there it is no more work to do for a task, it is not returned to the queue. It is the responsibility of the writer thread to put the task back into *SchedulableTasks* when the task has again events in its queue. This coordination is achieved without locking by using an atomic counter $t.queueSize$ for the number of events in the task's event queue. The writer increments $t.queueSize$ after having written an event, whereas the scheduler thread decrements it after reading an event. The scheduler thread will only return the task to the set if the value is non-zero. A writer recognizes that a scheduler thread did not return a task if the value before it succeeded in incrementing the atomic counter was zero.

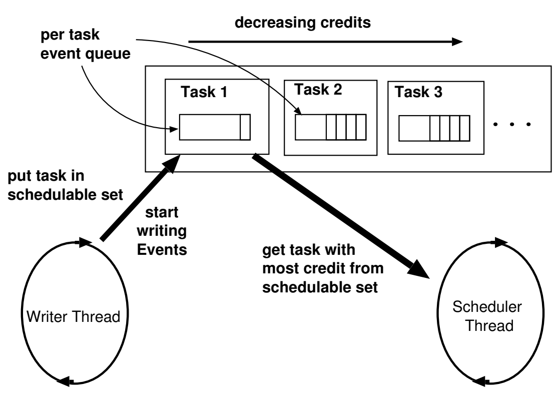
B. Lock-free access to the concurrent priority queue

The priority queue that implements *SchedulableTasks* is accessed concurrently by writer threads and scheduler threads. Because of their importance in scheduling on multiprocessor operating systems, much work has been done on algorithms for concurrent priority queues. This work covers both blocking [5], [6], [7] and non-blocking approaches [7], [8]. While in the event-system different tasks can have an arbitrary number of time credits, meaning that in theory we need to use an algorithm that supports arbitrary priorities, in practice there is little to be achieved by distinguishing between tasks that have a number of credits which are within some range of each other. This observation allows us to quantize the credits such that there is a fixed number of priority levels.

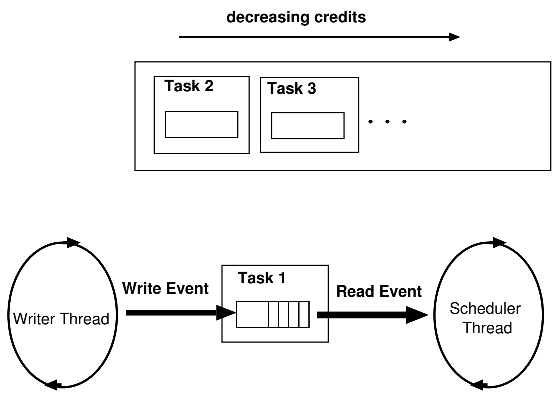
A blocking quantizing priority queue algorithm has been described in [7]. The range of priorities is divided such that every valid priority falls into one of N buckets. Elements in the same bucket are retrieved according to their arrival order. Shavit *et al.* call this a *SimpleLinear* priority queue and report through the use of simulation that its performance compared with a range of other approaches is best for small numbers of processors (fewer than 16). They propose the use of a funneling mechanism, whereby processes accessing the same bucket can recognize and avoid contention by having only one of them perform the required operations.

Our implementation follows the same idea as *SimpleLinear*, but unlike *SimpleLinear*, it is non-blocking. Each bucket is associated with a lock-free FIFO queue. These buckets are placed in an array such that the highest priority (most credits) bucket is at the extreme left and the priority reduces as we move to the right. The scheduler thread starts at the highest priority bucket and moves to the right until it finds a bucket with a non-empty queue. It then attempts to take the highest priority task in this queue. If the scheduler thread succeeds, it dispatches this task, if not it moves to the right again. If it reaches the end of the structure without finding work it returns a null value. Figure 3 shows the lock-free priority queue. The precision of the share can be traded off against the efficiency of the system by increasing or decreasing the number of buckets.

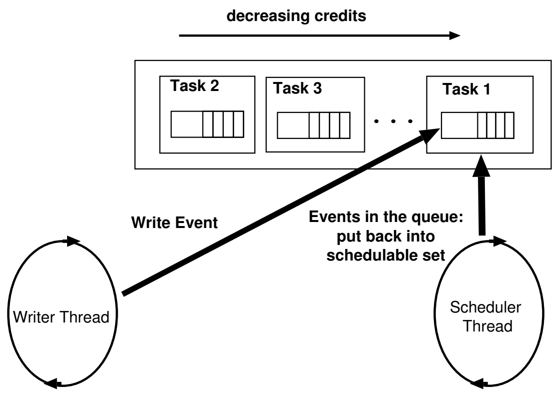
The total number of operations required to identify into which bucket to add a task is constant, but the number to find the first non-empty bucket increases linearly with the number of buckets, N. The thread determines whether a bucket is non-empty by checking an atomic counter. For a given number of buckets the worse case removal time is constant, i.e. when the element is found in the least priority bucket.



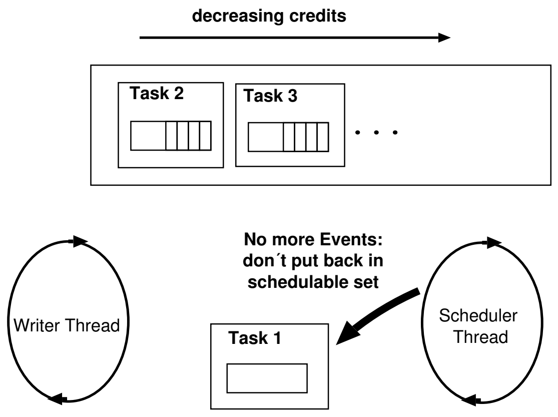
(a) Events arrive for an unscheduled task.



(b) Writer thread writes to the task's event queue and scheduler reads from it.



(c) Remaining events in the queue: Scheduler thread puts the task back.



(d) No more events in the task's queue.

Fig. 2. Coordination between writer and scheduler threads.

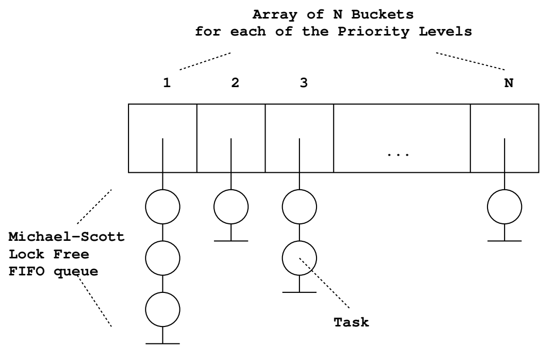


Fig. 3. Simple lock-free priority queue.

If no task overruns its alloted time slice then we can guarantee that no task goes unscheduled for longer than 10 ms by setting the value T over which the share is respected to 10 ms. Now assume we wish to be able to distinguish tasks at a time granularity of 100 μs then we need 100 buckets. This in turn requires on average 50 integer comparisons each time we schedule a task. To reduce this overhead we group buckets into bucket groups of size M. An additional atomic counter is kept for the entire bucket group as well as for the individual buckets. The average number of comparisons is then $Y = M/2 + N/(2M)$. The optimal value of M for fixed N is now the value at which the derivative of Y with respect to M is zero, which is given by $M = \sqrt{N}$. The expected number of operations is $\sqrt{N}$ and the worst case $2\sqrt{N}$, i.e. for 100 buckets we expect to have 10 additional integer comparisons for removing a task and in the worst case we have 20. The same technique can be performed at multiple different levels, i.e. by creating groups of bucket groups etc. In practice trying to distinguish between messages at very fine granularities makes little sense as the variance in other factors beyond the messaging system's control dominate, e.g. network delay. We find that for time granularities that make sense in a messaging system running over a LAN (in the 100-1000 μs range) one level of bucket groups is adequate.

When no schedulable task has work to do, the scheduler resets the credits allocated to all tasks in the *SchedulableTasks* set. The scheduler recognizes that this has occurred when the task returned from the *SchedulableTasks* set has zero or less credit. The scheduler removes all the tasks from the set, resets their credit as described in Section III, and returns them to the set. Multiple scheduler threads can identify the need for credit resetting and perform this operation in parallel. This is a consequence of the fact that a task can be read by exactly one scheduler thread from the set and that all credit information about the task is contained within it.

It may be the case that one or more tasks with credit are currently being serviced by other scheduler threads when a given scheduler thread identifies that no current task in the *SchedulableTasks* has work. A thread that was servicing a task while the credits were being reset by another thread needs to recognize that this has occurred and update this task's credit before putting it back into *SchedulableTasks*. This is achieved using an atomic counter that is incremented after each reset. Each thread keeps the value of this counter before getting a task and checks whether it has changed before it puts the task back.

C. Linearizability and the concurrent priority queue

Herlihy's *linearizability* condition [9] can be stated informally as follows. A data structure is linearizable if and only if:

- all possible histories over that data structure have a legal sequential equivalent;
- the order of operations that do not execute concurrently are respected within the sequentially equivalent history.

If a concurrent data structure is linearizable we can then reason about it using its sequential equivalent. Paper [7] claims that the algorithm described in the preceding section is linearizable, however this is not the case. We show that it is not by the means of a counter example. As for any lock-free concurrent priority queue, because elements are being simultaneously added and removed, the get operation may not retrieve the task with the highest priority in the queue at the moment the operation completes. In our implementation, it is possible that another thread adds a task to a bucket that the reading thread has already passed over. The following describes a valid sequence of operations in our implementation using the notation of [9], where x is a task with higher priority than task y:

$$\begin{gathered} \mathit{Get}()A;\ \mathit{Put}(x)B;\ \mathit{Ok}()B;\ \mathit{Put}(y)B; \\ \mathit{Ok}()B;\ \mathit{Ok}(y)A;\ \mathit{Get}()C;\ \mathit{Ok}(x)C; \end{gathered}$$

Each operation consists of a start of invocation and its completion. For example, *Put*(x) *A* stands for the start of the *put* invocation of task x into the priority queue by thread *A*, and *Ok*$()$ *A* corresponds to its completion. The history is not linearizable when the priority of x is higher than the priority of y as no *legal* sequential history can respect the fact that the addition of x completed before the addition of y but was removed after.

The practical consequence of this is that thread *A* retrieves a lower priority task than thread *C*, even though the get operation of thread *A* precedes the get of thread *C* and task x was put in before task y. Figure 4 illustrates this inversion of priorities.

The advantage of linearizability is that linearizable data structures can be composed such that the resulting combined history is also linearizable. Showing that a data structure is linearizable allows it to be used in many different contexts. We are interested in the use of the concurrent priority queue specifically in the context of the credit scheduler. Therefore we must demonstrate that the priority queue behaves appropriately

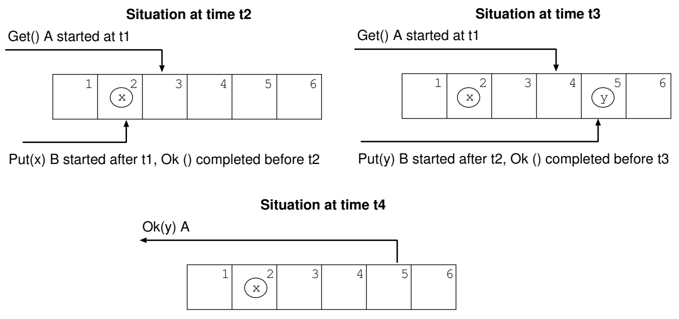


Fig. 4. The interleaving of get and put operations ($t4 > t3 > t2 > t1$).

in that context. We now give the invariant of the concurrent priority queue and show that it is adequate for our purposes.

D. The concurrent priority queue invariant

Let a put operation be defined as follows:

$$PUT ::= [e : Element, start : Time, end : Time]$$

where *start* and *end* are the times that the operation started and completed at and e is the element added with priority *e.prio*. Let a get operation be defined as:

$$GET ::= [p : PUT, start : Time, end : Time, e : Element]$$

where *start* and *end* are the times that the operation started and completed at and p is the put operation whose element the get operation retrieves. It must be the case that an element is retrieved only after it has been added, i.e.:

$$\forall g : GET \ ,\ g.end > g.p.start$$

We define a history of the observed get operations ordered by their time of completion, i.e.:

$$H \ : \ SEQ\ of\ GET \ ,\ \forall i < j \quad H[i].end \leq H[j].end$$

Then the following must hold $\forall i, j \quad i < j$:

$$H[i].e.prio < H[j].e.prio \ \ => \ \ H[i].start < H[j].p.end$$

This simply states that if an element e was removed from the queue with a higher priority than one removed before it, then the operation that placed said element e in the queue must have completed after the preceding removal had already started. This is sequentially consistent according to Lamport's definition in [10], i.e. an equivalent sequential history can always be produced, but because that sequential history would involve inverting the order of certain put and get operations, it is not linearizable.

Although the data structure is not linearizable and hence does not have the same behavior as a sequential priority queue, this does not exclude its use within the scheduler. In the specific context of the scheduler, a given scheduling process always gets a task from the queue and then puts it back. A given process history is therefore an interleaved series of get's and put's that are always sequential with respect to each other in the complete history. This means that, while it is possible that the data structure occasionally does not behave like a sequential priority queue, the duration of this discrepancy is bounded: the process that placed the task T_a that suffers from the priority inversion will immediately attempt to get the current highest-priority task. If no other task has been added, T_a will be chosen; if a higher priority T_b has been added then T_b will be chosen and the process that added T_b will in turn attempt to get the current highest-priority task and so on. It follows that even allowing for occasional priority inversions the overall share that a task receives over a given time period will be respected.

E. Fair shares on a multi-core machine

The scheduler guarantees that *when* a scheduler threads executes it will choose the *available* task that best fits the schedule. Suppose we have a one CPU machine in which the scheduler thread is the only thread that runs, then all tasks will always be available and the share allocated to the task will simply be a fraction of the CPU.

In a real system there are many other threads running. Within the messaging system alone, we have timer threads, threads monitoring the I/O, and threads supporting the control part of the messaging protocol. The JVM itself typically runs several daemon threads, e.g. for garbage collecting, and then there is everything else that runs on the machine, e.g. the application. In short, in reality the scheduler threads are not always running, and the share allocated to a task on a single processor machine is a share of the fraction of the time the scheduler runs.

The situation is more complicated on a multi-core machine because when a scheduler thread is serving a task, that task is unavailable to other scheduler threads. This is simply a consequence of the fact that a task is allocated to at most one scheduler thread at any given moment.

Assume that the probability that a scheduler thread is scheduled by the OS is independent and denoted by P and that there are N processors and N scheduler threads in the system. If $P = 1$, then no task can ever get more than $1/N$ of the total time the scheduler's threads run. So for example if $N = 1$ the maximum share that a task can receive is 100%, if $N = 2$ the maximum is 50%, etc. More generally, the probability of there being k scheduler threads running simultaneously is given by the binomial distribution whose expected value is $N \cdot P$.

Hence we expect the number of processors on which scheduler threads run in parallel to simply be the product of the number of processors and their probability of getting scheduled. The maximum percentage of total scheduling time that any task can get is then given by

$$\frac{100\%}{\mathrm{Max}(1, \mathrm{N} \cdot \mathrm{P})}$$

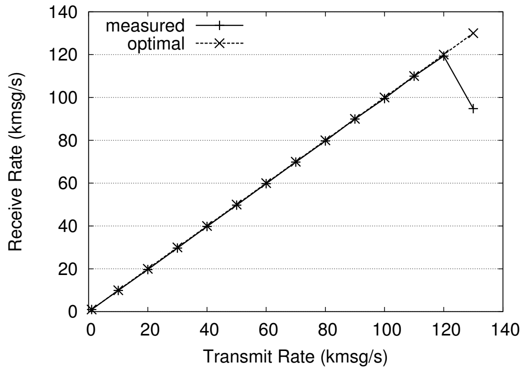
For fixed N this approaches 100% as P gets smaller, and for fixed P it approaches $1/N$ as N gets bigger.

Note that the scheduler is not *fair* in the sense used in WFQ [2], i.e. it is possible for a task to get more than it requested while another gets less than it requested. To make the scheduler both fair and work-conserving, it would be necessary to allow multiple scheduler threads to dispatch events from the same event queue simultaneously. This would not only complicate the scheduler, but, more importantly, would mean that FIFO delivery of messages within a given topic is no longer guaranteed by the scheduler.

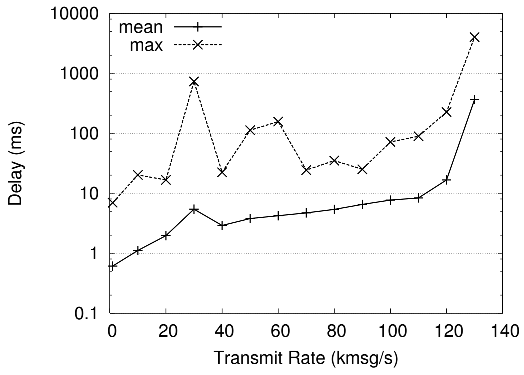
IV. Performance evaluation

In all of the tests the following configuration has been used unless otherwise stated. The time period over which the shares are valid is set to 10 ms. The number of buckets used in the priority queue is set to 100. We use a number of scheduler threads equal to the number of cores on the machines. The communication between machines is always Gigabit Ethernet, and we use TCP for the transport layer with the socket size set to 128 kbytes. The message size is 128 bytes. All machines are running a 2.6 Linux Kernel with Java 6. The machine on which the subscribers run has 2 × 2.3 GHz 4-core processors, i.e. 8 cores in total. All publishing machines have 2 × 3.0 GHz HyperThreading processors.

A. Bandwidth/delay for one publisher to one subscriber



(a) Effective throughput



(b) Delay as a function of the throughput

Fig. 5. Bandwidth/delay for one publisher to one subscriber.

We measure the maximum number of messages per second we can send between a single publisher and a single subscriber. We also measure the effect on the average delay of increasing the sending rate. To avoid having to synchronize clocks on distinct machines, the end-to-end delay is measured by marking some random sample of the sent messages at the publishing application and having the subscribing application echo these packets back to the publisher over UDP. This method over-estimates the end-to-end delay for low-sending rates (fewer than 1,000 msg/s) as the additional network latency is a significant fraction of the total delay, but is a good approximation for higher rates where the network latency plays a lesser role.

Figure 5 shows the evolution of the effective throughput and the end-to-end delay as a function of the publishing rate. The system sustains a rate of 120,000 msg/s. The average end-to-end delay rises from below 1 ms for 1,000 msg/s to slightly above 10 ms at 120,000 msg/s. Above this figure the system is no longer sustainable, and the average delay rises dramatically.

B. Scalability with increasing number of topics

We measure how the messaging system behaves with increasing number of topics. We run a set of subscribers within a single JVM. Each subscriber subscribes to a distinct topic. We run each publisher on a different machine. Each publisher attempts to send at the same fixed message rate. Figure 6(a) shows how the cumulative throughput at the subscribers evolves as the number of publishers increases for a per-publisher sending rate of 70,000 msg/s. We find an almost linear scaling with increasing number of topics up to the number of cores on the machine, demonstrating that topics can be serviced in parallel. The average delay for the sets of topics behaves similarly to that reported in Section IV-A, with an average delay of less than 50 ms for a cumulative throughput of 500,000 msg/s.

An interesting effect is observed when we increase the fixed sending rate to 80,000 msg/s (see Figure 6(b)) and 90,000 msg/s (see Figure 6(c)). The cumulative throughput initially increases linearly and then collapses, meaning for example that the total throughput for 8 publishers at 90,000 msg/s is significantly less than at 70,000 msg/s. We currently assume that this is an artifact of the Linux kernel when subjected to high interrupt rates, but have not yet been able to verify this.

The promising performance in Figure 6(a) shows that our design is able to take advantage of a multi-core architecture. The reader may better grasp the importance of these results by comparing them with Figure 7: there we perform exactly the same experiment as before, but we replace the lock-free *SchedulableTasks* in the scheduler with a simple queue accessed using a lock. The lock is necessary to avoid having the queue corrupted by threads manipulating the data structure concurrently. Figure 7 shows that the traditional approach does not scale. Using more than one scheduler thread improves performance slightly, but for increasing number of threads this effect is offset by the increased contention on the lock (8 scheduler threads perform worse than 2, even if the machine has 8 cores).

C. Task shares

Section II described the mathematical model of the scheduler for a single scheduling thread. Section III-E described how

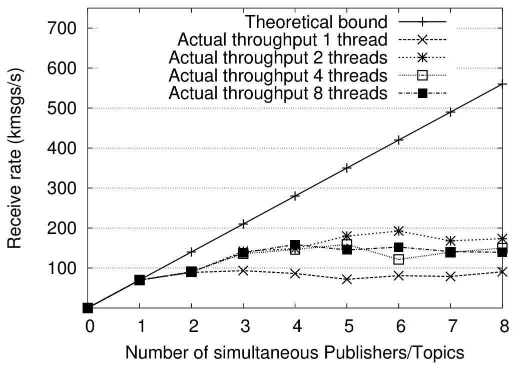


Fig. 7. Using a locked queue in the scheduler: scalability at 70,000 msg/s per topic with increasing number of topics.

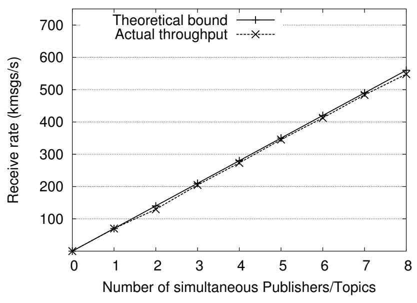
the actual share allocated to a task is influenced by the number of processors over which the messaging system runs and by the probability of scheduler threads being able to execute on those processors.

We test the actual measured received rates when ten topics are allocated distinct shares on a two-processor machine running either one or two scheduler threads. The system is pushed to saturation by attempting to send an aggregate sending rate twice that of the maximum measured achieved rate for the configuration. The share allocated to a topic and the fraction of the aggregated total it attempts to send at, are independent random variables from two distinct Poisson distributions. Thus, it can occur that a topic attempts to send at a very high rate with a very low share. The test is run until new results do not significantly change the average. The test was repeated 500 times, each time with a different randomly chosen configuration. We then measure the actual aggregate throughput R^{max} achieved at a given setting and used it to calculate the predicted received rates of the individual topics at those setting according to the model in Section II. Finally, we calculate the error as the distance between the vector of predictions and the vector of measured values. The relative error is defined as that error divided by the magnitude of the vector of measured values.

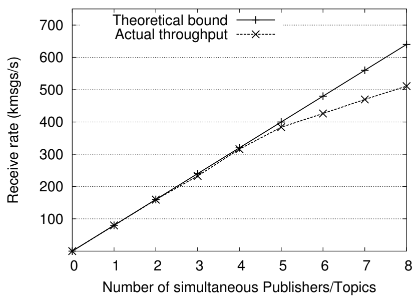
The CCDF (Complementary Cumulative Distribution Function) of the relative errors is given in Figure 8. For one scheduler thread, 85% of all runs have an error *smaller* than 10%; for two scheduler threads, the same is achieved for 80% of the configurations. The results in Figure 8(a) show the performance of the scheduler at saturation for randomly chosen settings. When the system is not at saturation the fidelity of the scheduler with respect to the model is almost exact (see Figure 8(b) for 100,000 msg/s total sending rate).

V. Conclusion

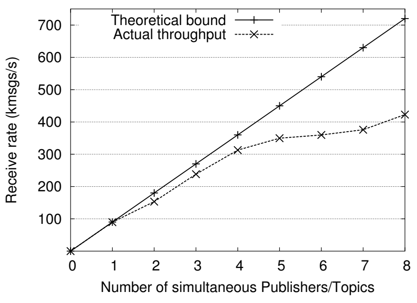
We have shown how a messaging system supporting different qualities of service for different topics can be built using a event dispatching model. We have motivated our choice of time-based credit scheduling within the messaging system and given a statement of the guarantees that it provides. We have described how the scheduler is parallelizable allowing it to scale on a multi-processor system through the use of



(a) Scalability at 70,000 msg/s per topic

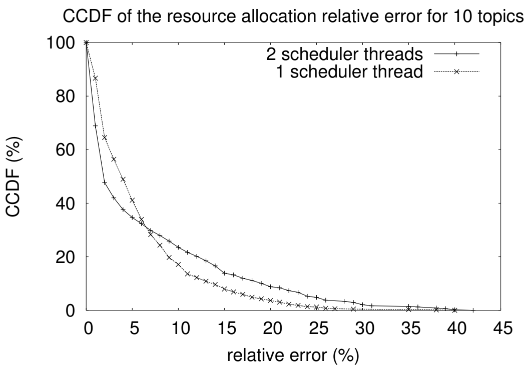


(b) Scalability at 80,000 msg/s per topic

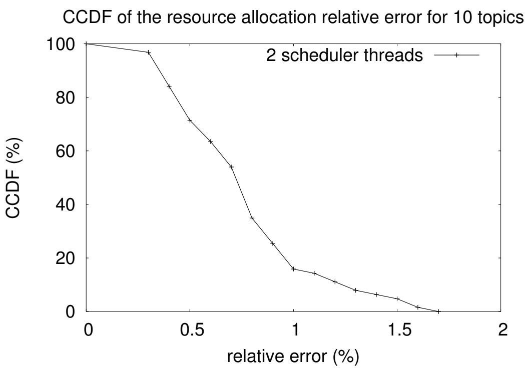


(c) Scalability at 90,000 msg/s per topic

Fig. 6. Throughput scalability with increasing number of topics.



(a) Share allocation accuracy on a saturated system.



(b) Share allocation accuracy on a non-saturated system.

Fig. 8. Difference between requested and measured shares.

a lock-free approach and given a formal invariant for the concurrent priority queue used. We have motivated our use of this data structure proving that it is not linearizable but that in the context used, its invariant is strong enough to allow the required scheduling behavior. We have reported on the performance of the messaging system showing that on a real multi-core architecture the throughput scales with number of cores, we have also described the measured delay and the consequences for topic isolation on running the scheduler on a multiprocessor machine.

References

[1] A. Tenenbaum and R. van Renesse, “A critique of the remote procedure call paradigm,” in *In proceedings of Euteco*, 1988.

[2] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” in *SIGCOMM ’89: Symposium Proceedings on Communications Architectures & Protocols*. New York, NY, USA: ACM Press, 1989, pp. 1–12.

[3] N. Yin, “Max-Min Fairness vs MCR Guarentee on Bandwidth Allocation for ABR,” in *IEEE Proc ATM’96, workshop San Franciso, CA*, 1996.

[4] M. M. Michael and M. L. Scott, “Simple, fast, and practical non-blocking and blocking concurrent queue algorithms,” *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, pp. 267–275, 1996.

[5] Q. Huang, “An evaluation of concurrent priority queue algorithms,” Massachusetts Institute of Technology, MIT Cambridge, MA, USA, Tech. Rep., 1991.

[6] G. C. Hunt, M. M. Michael, S. Parthasarathy, and M. L. Scott, “An efficient algorithm for concurrent priority queue heaps,” University of Rochester, Rochester, NY, Tech. Rep. TR560, Dec. 1994. [Online]. Available: citeseer.ist.psu.edu/hunt96efficient.html

[7] N. Shavit and A. Zemach, “Scalable concurrent priority queue algorithms,” in *PODC ’99: Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: ACM Press, 1999, pp. 113–122.

[8] H. Sundell and P. Tsigas, “Fast and lock-free concurrent priority queues for multi-thread systems,” in *Proceedings of the 17th International Parallel and Distributed Processing Symposium*. IEEE press, 2003. [Online]. Available: citeseer.ist.psu.edu/sundell03fast.html

[9] M. P. Herlihy and J. M. Wing, “Linearizability: A correctness condition for concurrent objects,” *ACM Transactions on Programming Languages and Systems*, vol. 12, no. 3, pp. 463–492, Jul. 1990.

[10] L. Lamport., “How to make a multiprocessor computer that correctly executes multiprocess programs,” *IEEE Transactions on Computers*, vol. 28, no. 9, pp. 690–691, Sep. 1979.

DETECTING PROTOCOL ERRORS USING PARTICLE SWARM OPTIMIZATION WITH JAVA PATHFINDER

Marco Ferreira
Escola Superior de Tecnologia
e Gestão de Leiria
Instituto Politécnico de Leiria
Email: mpmf@estg.ipleiria.pt

Francisco Chicano and Enrique Alba
Dpto. Lenguajes y Ciencias
de la Computación
University of Málaga
Email: {chicano,eat}@lcc.uma.es

Juan A. Gomez-Pulido
Dep. of Technologies of Computers
and Communications
University of Extremadura
Email: jangomez@unex.es

KEYWORDS

Validation, testing, protocols, model checking, Java PathFinder, Particle Swarm Optimization

ABSTRACT

Network protocols are critical software that must be verified in order to ensure that they fulfil the requirements. This verification can be performed using model checking, which is a fully automatic technique for checking concurrent software properties in which the states of a concurrent system are explored in an explicit or implicit way. However, the state explosion problem limits the size of the models that are possible to check. Particle Swarm Optimization (PSO) is a metaheuristic technique that has obtained good results in optimization problems in which exhaustive techniques fail due to the size of the search space. Unlike exact techniques, metaheuristic techniques can not be used to verify that a program satisfies a given property, but they can find errors on the software using a lower amount of resources than exact techniques. In this paper, we propose the application of PSO to solve the problem of finding safety errors in network protocols. We implemented our ideas in the Java Pathfinder (JPF) model checker to validate them and present our results. To the best of our knowledge, this is the first time that PSO is used to find errors in concurrent systems. The results show that PSO is able to find errors in protocols in which some traditional exhaustive techniques fail due to memory constraints. In addition, the lengths of the error trails obtained by PSO are shorter (better quality) than the ones obtained by the exhaustive algorithms.

1 Introduction

One of the most important phases in protocol design is the testing phase. Unlike other less critical software (like a videogame or a graphical design program), network protocols might be verified in order to prove that they fulfil the requirements. An error discovered after the deployment of a network protocol in a grid computing environment can entail the loss of an important amount of money due to repair and maintenance costs.

Model checking (Clarke et al., 2000) is a well-known and fully automatic formal method for verifying that a given hardware or software system fulfils a property. This verification is performed by analyzing all the possible system states (in an explicit or implicit way) in order to prove (or refute) that the system satisfies the property. Examples of properties are the absence of deadlocks, the absence of violated assertions, and the fulfilment of an invariant. It is possible also to specify more complex properties using temporal logics like Linear Temporal Logic (LTL) (Clarke and Emerson, 1982) or Computation Tree Logic (CTL) (Clarke et al., 1986). One of the best known explicit model checkers is SPIN (Holzmann, 2004), which takes a software model codified in Promela and a property specified in LTL as inputs. Promela is not a programming language used for real programs, it is just a language for modelling concurrent systems in general, and protocols in particular. This drawback is currently solved by the use of translations tools or the model checker Java PathFinder (Groce and Visser, 2004), which in its last versions directly works on bytecodes of multithreaded Java programs.

The amount of states of a given concurrent system is very high even in the case of small systems, and it increases exponentially with the size of the model. This fact is known as *the state explosion problem* (Valmari, 1998) and limits the size of the model that an explicit state model checker can verify. This limit is reached when it is not able to explore more states due to the absence of free memory. Several techniques exist to alleviate this problem. They reduce the amount of memory required for the search by following different approaches. On one hand, there are techniques which reduce the number of states to explore, such as partial order reduction (Clarke et al., 1999) or symmetry reduction (Lafuente, 2003). On the other hand, we find techniques that reduce the memory required for storing one state, such as state compression, minimal automaton representation of reachable states, and bitstate hashing (Holzmann, 2004). Symbolic model checking (Burch et al., 1994) is another very popular alternative to explicit state model checking that can reduce the amount of memory required for the verification. In this case, a set of states is represented by a finite propositional formula. However, exhaustive search techniques are always handicapped in real concurrent programs because most of these programs are too complex even for the most advanced techniques. There-

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

fore, techniques of bounded (low) complexity as metaheuristics will be needed for medium/large size programs working in real world scenarios.

In this work we propose the use of Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) for finding errors in concurrent systems. This is the first time (to the best of our knowledge) that PSO has been applied to this problem. We have included our PSO algorithm inside the Java Pathfinder (JPF) model checker. This way we can find errors in protocols written in Java, which allows us to check real implementations instead of models.

The paper is organized as follows. The next section presents the required foundations on model checking and previous work on which ours is based. Section 3 presents a formal description of the problem while in Section 4 our algorithmic proposal is detailed. Finally, the results are presented and commented in Section 5 and the conclusions and future work depicted in Section 6.

2 Background

Explicit state model checkers work by searching for a counterexample of the property in the model. They explore the synchronous product (also called Büchi automaton) of the transition system of the model and the negation of the property and they search for a cycle of states containing an accepting state reachable from the initial state. If such a cycle is found, then there exists at least one execution of the system not fulfilling the property (see (Holzmann, 2004) for more details). If such kind of cycle does not exist then the system fulfils the property and the verification ends with success. This search is usually performed with the Nested Depth First Search (NDFS) algorithm (Holzmann et al., 1996).

Safety properties can be checked by searching for a single accepting state in the Büchi automaton. That is, when safety properties are checked, it is not required to find an additional cycle containing the accepting state. This means that safety property verification can be transformed into the search for one objective node (one accepting state) in a graph (Büchi automaton) and general graph exploration algorithms like Depth First Search (DFS) and Breadth First Search (BFS) can be applied to the problem. Furthermore, in (Edelkamp et al., 2004) the authors utilize heuristic information for guiding the search. They assign a heuristic value to each state that depends on the safety property to verify. After that, they utilize classical algorithms for graph exploration such as A*, Weighted A* (WA*), Iterative Deepening A* (IDA*), and Best First Search (BF). The results show that, by using heuristic search, the length of the counterexamples can be shortened (they can find optimal error trails using A* and BFS) and the amount of memory required to obtain an error trail is reduced, allowing the exploration of larger models.

The utilization of heuristic information for guiding the search for errors in model checking is known as *heuristic* or *directed model checking*. The heuristics are designed to lead the exploration first to the region of the state space in which an error is likely to be found. This way, the time and memory required to find an error in faulty concurrent systems is reduced in average. However, no benefit from heuristics is obtained when the goal is to verify that a given program fulfils a given property. In this case, all the state space must be exhaustively explored.

When the search for short error trails with a low amount of computational resources (memory and time) is a priority (for example, in the first stages of the implementation of a program), non-exhaustive algorithms using heuristic information can be used. Non-exhaustive algorithms can find short error trails in programs using less computational resources than optimal exhaustive algorithms (as we will see in this paper), but they cannot be used for verifying a property: when no error is found using a non-exhaustive algorithm we still cannot ensure that no error exists. Due to this fact we can establish some similarities between heuristic model checking using non-exhaustive algorithms and software testing (Michael et al., 2001). In both cases, a large region of the state space of the program is explored in order to discover errors; but the absence of errors does not imply the correctness of the program. This relationship between model checking and software testing has been used in the past for generating test cases using model checkers (Ammann et al., 1998).

A well-known class of non-exhaustive algorithms for solving complex problems is the class of metaheuristic algorithms (Blum and Roli, 2003). They are search algorithms used in optimization problems that can find good quality solutions in a reasonable time. The search for accepting states in the Büchi automaton can be translated into an optimization problem and, thus, metaheuristic algorithms can be applied to the search for safety errors. In fact, Genetic Algorithms (GAs) (Godefroid and Khurshid, 2004) and Ant Colony Optimization (ACO) (Alba and Chicano, 2007) have been applied in the past.

3 Problem Formalization

The problem of searching for safety property violations can be translated into the search of a path in a graph (the Büchi automaton) starting in the initial state and ending in an objective node (accepting state). We are also interested in minimizing the length of the error trail obtained. We formalize here the problem as follows.

Let $G = (S, T)$ be a directed graph where S is the set of nodes and $T \subseteq S \times S$ is the set of arcs. Let $q \in S$ be the *initial node* of the graph and $F \subseteq S$ a set of distinguished nodes that we call *final nodes*. We denote with $T(s)$ the successors of node s. A finite path over the graph is a sequence of nodes $\pi = s_1 s_2 \ldots s_n$ where $s_i \in S$ for $i = 1, 2, \ldots, n$. We denote with π_i the ith node of the sequence and we use $|\pi|$ to refer to the length of the path, that is, the number of nodes of π. We say that a path π is a *starting path* if the first node of the path is the initial node of the graph, that is, $\pi_1 = q$. We will use π_* to refer to the last node of the sequence π, that is, $\pi_* = \pi_{|\pi|}$.

Given a directed graph G the problem at hands consists in finding a starting path π ending in a final node with minimum $|\pi|$. That is, minimize $|\pi|$ subject to $\pi_1 = q \wedge \pi_* \in F$.

The graph G used in the problem is derived from the synchronous product B of the model and the negation of the LTL formula of the property. The set of nodes S in G is the set of states in B, the set of arcs T in G is the set of transitions in B, the initial node q in G is the initial state in B, and the set of final nodes F in G is the set of accepting states in B. In the following, we will also use the words *state*, *transition*, and *accepting state* to refer to the elements in S, T, and F, respectively.

4 Algorithmic Proposal

In this section we describe our proposal for searching for safety errors in concurrent systems in general, and network protocols in particular. We will start by describing the PSO algorithm, then we will show how the particles represent execution paths of the concurrent systems, and finally we will describe the fitness function used to evaluate the particles (execution paths).

4.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based metaheuristic search algorithm developed by Kennedy and Eberhart in 1995 (Kennedy and Eberhart, 1995). It works in the same way as genetic algorithms and other evolutionary algorithms in the sense that they all update a set of solutions (called *swarm* in the context of PSO) applying some operators and using the fitness information to guide the set of solutions to better regions of the search space. PSO differs from these algorithms by simulating the social behaviour of a swarm and by not employing a survival of the fittest model.

In PSO, a particle is a point in the search space, that is, it represents a possible solution. Each particle, besides its position, has a velocity and knowledge about its own experience and about its neighbours' experience. Although several topologies (or social networks) can be found in the literature, in the original PSO every particle has experience information of the all swarm, that is, every particle in the swarm is considered a neighbour of every other particle. This topology is sometimes referred as *gBest* and is the one used in our experiences. The basic algorithm of the PSO we used can be found in Algorithm 1.

Algorithm 1 Pseudocode of a PSO

```
1: $P$ = generateInitialPopulation();
2: while not stoppingCondition() do
3:     evaluate($P$);
4:     calculateNewVelocityVectors($P$);
5:     move($P$);
6: end while
7: return the best found solution
```

The algorithm starts by creating an initial population. Then it evaluates the fitness of each particle and updates, if necessary, its own experience (*pBest*) and the swarm experience (*gBest*). The fitness value of each individual is calculated by a fitness function that is dependent on the problem. A particle's *pBest* variable is updated when the fitness function returns a better value (larger if we are maximizing, smaller otherwise) than the current one for the particle. The *gBest* variable is also updated if that fitness value is also better than the previous *gBest* fitness value. Using *pBest* and *gBest*, the velocity and position of each particle is updated according to the following expressions:

$$\begin{aligned} v_i &= w \cdot v_i + c_1 \cdot rand_1 \cdot (pBest_i - x_i) \\ &\qquad + c_2 \cdot rand_2 \cdot (gBest_i - x_i) \ , \quad (1) \\ x_i &= x_i + v_i \ , \quad (2) \end{aligned}$$

where v_i represents the particle velocity, w represents the inertia factor, c_1 and c_2 are learning factors, $rand_1$ and $rand_2$ are two random values in the range $[0, 1]$, and x_i represents the particle position. The algorithm continues to move the particles until a stopping criterion, usually a maximum number of iterations, is met.

In our implementation of the algorithm the inertia factor changes during the search, as suggested in (Shi and Eberhart, 1998). At the beginning w is high in order to perform an explorative search. The inertia factor is decreased during the search in order to switch to a more exploitative search. The idea behind this mechanism is to search first a promising region in the search space and then to exploit that promising region. To avoid falling into local optima, we have added a perturbation operator to the basic PSO algorithm. When the best fitness value (the one of *gBest*) does not improve for a number of iterations, defined as $itupert$ (ITerations Until PERTurbation), the particles are randomly moved to another position with the hope that, using that new starting point, they can find a better solution. When that happens, the inertia factor is also reset to its maximum value so that PSO tends to search globally, instead of locally.

We have also implemented a cache feature to our PSO. Since we are looking for paths in large graphs, we want to avoid having to evaluate twice the same path (or partial path). We avoid that evaluation by storing in memory all the visited states and their associated fitness values. When the PSO needs to evaluate a new path, it uses this memory for all the states already visited, avoiding the costly job of expanding and evaluating the same state again. This is not needed to find an error, being just a performance improvement, and as soon as more memory is needed, our PSO frees this cache memory.

4.2 Representation of the Execution Paths

We are trying to find paths that lead to an error state in a protocol. A path can be described as the sequence of transitions that occurred from an initial state to a final state. Figure 1 shows an example of a state graph evidencing the path described by transitions 0, 1, 1. The number of transitions available varies from state to state.

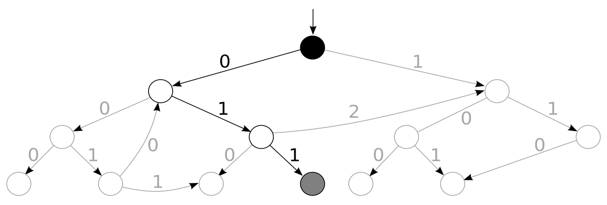


Figure 1: Example graph showing the states of a program and the variable number of transitions in each state.

The number of transitions needed to reach a goal state (the path size) is usually unknown beforehand, that is, we do not know what is the length of the shortest path from the initial state to a goal state. If the particles were composed of a fixed length sequence of transitions, the algorithm could always fail in finding a goal state. This leaves two problems to be solved by the solution representation: which transition to select in each state of the path and how long the path is.

One way to select a transition would be to simply use an integer value representing the number of transition as shown in Figure 1. Two problems arise from this approach: what would be the range of that value and how to deal with discrete values in the PSO. While discrete values could be handled specially adapting the algorithm, the range of the integer value would be more difficult to determine. Instead, we solved this problem by using a floating-point value in the range $[0, 1)$. We can discover which transition to follow from a given state simply by multiplying that value by the number of transitions available, truncating the result to the nearest smaller integer.

To allow for different solution lengths in the population, we used another floating-point value, this time in the range $[1, maxLen]$ where $maxLen$ is a parameter for the algorithm. This value indicates how many transitions the path will have. Every particle is thus composed of a vector of $maxLen + 1$ floating-point values, or dimensions, where the first value indicates the path size and the remaining values indicate which transition to select at that point of the path.

4.3 Fitness Function

JPF can search for two different kinds of safety property violations: deadlocks and assertion violations. In our experiments we searched for deadlocks because the implementations of the protocols we use in the experimental section are intentionally seeded with errors that lead to a deadlock situation. Also, shorter paths are preferred to longer paths and so, our fitness function $f(x)$ is defined as follows:

$$f(x) = DL * deadlockfound + numblocked + \frac{1}{1 + plen} \ , \quad (3)$$

where the constant DL represents the value given if a deadlock is found and $numblocked$ represents the percentage (in the $[0, 100]$ range) of blocked threads at the end of the path. The variables $deadlockfound$ indicates if a deadlock has been found (value 1) or not (value 0) and $plen$ represents the number of transitions in the path. The PSO will try to maximize $f(x)$. We used 10000 for DL which will give a large preference to a path leading to a deadlock.

5 Experiments

We have implemented our algorithmic proposal in JPF and report, in this section, the results of the experiments comparing the performance of PSO against three exhaustive algorithms. To evaluate the efficacy of PSO to find errors in protocols implemented in Java we selected two fairly complex and well known protocols: the Group Address Registration Protocol (GARP) and the General Inter-ORB Protocol (GIOP). We based our implementation of these protocols on previously studied Promela implementations found in (Nakatani, 1997) and (Kamel and Leue, 1998). GARP was proved correct in (Nakatani, 1997), but several implementation restrictions introduced both assertion violations and deadlocks in it. GIOP also has a known deadlock (Kamel and Leue, 1998). The implementation of GIOP we use is composed of two users (clients) and one server. Since these two protocol implementations have faults, and due to their fairly complex nature, they provide a good benchmark to compare the efficacy of PSO against the traditional exact search methods.

5.1 Algorithms and Parameters

For the experiments we use three exhaustive techniques in addition to the PSO: Depth First Search, Breadth First Search, and Random Depth First Search (RandomDFS). The latter is a variation of DFS in which the transition choice order is randomized when a state is expanded. Using this functionality allows DFS to behave slightly differently than standard. Usually, DFS follows the first transition in each state. When that transition is fully explored, DFS advances to the next transition and repeats the process. With the option to randomize the transition choice order turned on, JPF first randomizes the transitions order before presenting them to DFS, obtaining a stochastic exhaustive algorithm. While DFS still thinks it is using the first transition, in reality it may be using any other transition. This option may allow DFS to find errors in situations where it normally would not find any due to memory constraints.

There are several parameters that must be defined for PSO to work correctly. Being so, we used the common value of 2 for both c_1 and c_2. We have also used a linear decreasing inertia factor (w) from 1.2 to 0.6. The linear decreasing inertia choice has been previously studied with (Shi and Eberhart, 1998) suggesting it improves the PSO performance. We have also used 300 as $maxLen$, thus limiting the paths to a maximum of 300 transitions. To avoid falling into local optima, we defined $itupert$ as 5 iterations without improvement until a perturbation is made. As stopping criterion, we have used 30 iterations, that is, the PSO will move each particle 30 times. We used 20 particles in the PSO swarm. Table 1 summarizes these parameters.

Table 1: Parameters of PSO

Parameter	Value
w	Linearly decreasing from 1.2 to 0.6
c_1	2
c_2	2
$maxLen$	300
$itupert$	5
Stopping Criterion	30 iterations
Number of Particles	20

For each of the protocols we have executed PSO and RandomDFS 50 times to get a high statistical confidence since they are stochastic algorithms. We report the average (avg), the standard deviation (std), the minimum (min), and the maximum (max) of the different measurements collected. BFS and DFS were only executed once, since they are deterministic algorithms. These experiments were executed in a Pentium Core 2 Duo at 2.14 GHz using Windows XP Service Pack 2 and the Sun JRE 1.6.0_02 limited to 512 MB of RAM.

5.2 Results

In Table 2 we show the results on the GARP protocol. We present the hit rate (number of executions that found an error against the total number of executions), the time taken until an error was found (first error time) measured in seconds, the total time taken by the execution of the search, the depth of the first error found (first error depth) measured as the error trail length, the smallest error trail found, and the total memory used in the execution of the algorithms measured in megabytes.

In this first protocol, DFS and BFS fail to find an error, running out of memory before the error is found. Using RandomDFS, randomizing the search order, the error is found on only 9 of the 50 runs. PSO is able to find an error in every run. Furthermore, although we only show the path length to the first error found and the length of the shortest path leading to an error, PSO finds multiple errors, both deadlocks and assertion violations, in each run.

Analyzing the total time spent by each of successful runs, we can observe that RandomDFS has a faster execution time. That is explained by the fact that PSO continues to try to improve the error trail (making it shorter) while RandomDFS stops as soon as an error is found. If we look at the first error time, which is the elapsed time until any error is found, then we can see that the difference between RandomDFS and PSO is small (in fact, a statistical test not shown reveals that the difference is not statistically significant).

Table 2: Results of the algorithms with GARP

Measure	Statistics	BFS	DFS	RandomDFS	PSO
Hit rate	prop	0/1	0/1	9/50	50/50
First error time (s)	avg	–	–	5.89	6.40
	std	–	–	8.52	6.67
	min	–	–	1	0
	max	–	–	29	31
Total time (s)	avg	–	–	5.89	91.64
	std	–	–	8.52	19.81
	min	–	–	1	48
	max	–	–	29	137
First error depth	avg	–	–	4621.44	178.62
	std	–	–	4848.39	38.92
	min	–	–	555	121
	max	–	–	16922	288
Smallest error depth	avg	–	–	4621.44	133.86
	std	–	–	4848.39	11.53
	min	–	–	555	120
	max	–	–	16922	173
Memory used (MB)	avg	–	–	97.67	412.12
	std	–	–	130.31	1.85
	min	–	–	31	408
	max	–	–	457	416

With respect to the first error depth, PSO shows a clear advantage by having an error trail with approximately 179 transitions while RandomDFS obtains an error trail of approximately 4621 transitions (using the same time). PSO continues to improve that error trail size, reducing the size of that first error trail by approximately 45 transitions.

Regarding the memory consumed by both algorithms, we can see that RandomDFS has huge variations, ranging from a small amount of memory (31 MB) up to running out of memory (when no error is found). PSO uses about 412 MB of memory, with little variation. That is explained by the fact that PSO does not require to save visited states to perform the search. However, if memory is available, our PSO will save those states in a cache to improve the speed of operation. That means that PSO will use all the available memory just to improve performance. If there is no more memory available, performance is degraded, but the search continues. This contrasts with the behaviour of DFS and BFS that cannot find an error in the GARP protocol using 512 MB.

Now we turn to the GIOP protocol. Table 3 shows the results. In this protocol, BFS is again unable to find an error due to memory constraints, but DFS finds one error successfully. RandomDFS and PSO are also able to find errors in all the runs.

As expected, PSO requires again more processing time than the exact algorithms. DFS and RandomDFS find an error very quickly while PSO requires about 19 seconds to find an error. This may be due to the size limit of 300 transitions we impose to PSO. The first error found by

Table 3: Results of the algorithms with GIOP

Measure	Statistics	BFS	DFS	RandomDFS	PSO
Hit rate	prop	0/1	1/1	50/50	50/50
First error time (s)	avg	–	4	1.26	19.10
	std	–	0	0.44	15.76
	min	–	4	1	1
	max	–	4	2	71
Total time (s)	avg	–	4	1.26	149.52
	std	–	0	0.44	29.25
	min	–	4	1	74
	max	–	4	2	181
First error depth	avg	–	2120	1293.16	291.32
	std	–	0	282.51	4.25
	min	–	2120	907	280
	max	–	2120	2301	300
Smallest error depth	avg	–	2120	1293.16	280.52
	std	–	0	282.51	3.00
	min	–	2120	907	272
	max	–	2120	2301	287
Memory used (MB)	avg	–	38	32.68	414.84
	std	–	0	2.51	1.88
	min	–	38	31	408
	max	–	38	42	418

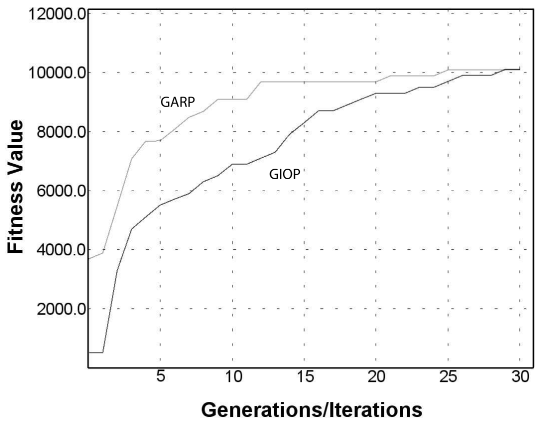


Figure 2: Convergence of PSO on GARP and GIOP

PSO is very near this limit and very near the smallest error found. In this protocol, PSO is not able to improve the error trail as much as with GARP. DFS finds an error with length of 2120 transitions, while RandomDFS requires, on average, about 1293 transitions to reach an error. The error trail given by PSO is, again, better for helping the protocol developers to debug it, with only 280 transitions. Regarding the memory required, we can see that DFS and RandomDFS require much less memory than PSO that, as explained before, uses all the available memory as cache.

Before concluding this section we are going to present the evolution of the swarm of PSO during the search. In Figure 2 we plot the fitness value of the best particle in each iteration. The horizontal axis represents the iterations, with the first one in the left and the last one in the right. The vertical axis represents the fitness value. For each iteration we take the *gBest* fitness values at that iteration in each of the 50 runs and plot their average. We want to illustrate with this analysis how the convergence of PSO takes place in both protocols.

We can observe in this graph that PSO converged faster on GARP and an error has been always found after 25 iterations (in all the executions). In the GIOP protocol the curve is not as steep as in the GARP problem, which is in accordance with the results we observed in Table 3, where the time to the first error is larger than in GARP (Table 2). In any of the protocols we can observe that PSO is guided towards the error, either maintaining the previous result or improving it in each iteration. In the last iterations, we can see that the fitness value has surpassed the 10000 value. According to (3) and the constant values we used for DL, whenever a deadlock is found the fitness value is larger than 10000, since for a deadlock to occur all the existing threads must be blocked, increasing the fitness value. The fact that the average of the 50 runs at the last iterations is larger than 10000 also shows that every run had found an error.

In conclusion, we can say that the results presented in this section show that while BFS should give the optimum (shortest) error trail, it cannot be used with complex protocols, since it requires too much memory. DFS also has memory constraints with complex protocols, especially if the error cannot be found in the first transitions, as observed for the GARP. While randomizing the transition order may help DFS, the error trails returned by that search method are still very large and therefore difficult to use for debugging. These experiments show that PSO promises to find good, if not optimum, error trails with a small or no time penalty comparing to DFS. Also, PSO can be used with complex programs without running out of memory, as it does not need to store the previously visited states. Storing them can improve performance, which means we can use the available memory for the algorithm's benefit.

6 Conclusions and Future Work

We have presented here a novel application of the PSO algorithm to find safety errors in network protocols using a model checking based approach. We have implemented and used it with JPF, a model-checker that allows the checking of concurrent programs written in Java, a known language to most developers. To the best of our knowledge, this is the first time that PSO has been applied to find errors in concurrent systems, using either JPF or any other model checker.

As future work, we believe it would be interesting to study the influence on the results of some techniques for reducing the amount of memory required in the PSO algorithm, such as partial order reduction or symmetry reduction. It would be interesting to investigate if the results of PSO can be improved using some more recent changes to the basic PSO algorithm, like Fuzzy Adaptive PSO (Shi et al., 2001) or different topologies of the PSO particle neighbourhood against the gBest topology we used in our implementation. It would be also interesting to compare PSO against other metaheuristic searches in the protocol validation problem, like Genetic Algorithms. Finally, some other heuristics could be used in the fitness function of the PSO to check if the error trail or the processing time could be improved.

7 Acknowledgments

This work has been partially funded by the Spanish Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project), and also by European CELTIC through the Spanish Ministry of Industry funds from FIT-330225-2007-1 (the CARLINK project).

REFERENCES

Alba, E. and Chicano, F. (2007). Finding safety errors with ACO. In *Genetic and Evolutionary Computation Conference*, pages 1066–1073, London, UK. ACM Press.

Ammann, P., Black, P., and Majurski, W. (1998). Using model checking to generate tests from specifications. In *Proceedings of the 2nd IEEE International Conference on Formal Engineering Methods*, pages 46–54, Brisbane, Australia. IEEE Computer Society Press.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Burch, J. R., Clarke, E. M., Long, D. E., McMillan, K. L., and Dill, D. L. (1994). Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4).

Clarke, E., Grumberg, O., Minea, M., and Peled, D. (1999). State space reduction using partial order techniques. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(3):279–287.

Clarke, E. M. and Emerson, E. A. (1982). Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, London, UK. Springer-Verlag.

Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263.

Clarke, E. M., Grumberg, O., and Peled, D. A. (2000). *Model Checking*. The MIT Press.

Edelkamp, S., Leue, S., and Lluch-Lafuente, A. (2004). Directed explicit-state model checking in the validation of communication protocols. *International Journal of Software Tools for Technology Transfer*, 5:247–267.

Godefroid, P. and Khurshid, S. (2004). Exploring very large state spaces using genetic algorithms. *International Journal on Software Tools for Technology Transfer*, 6(2):117–127.

Groce, A. and Visser, W. (2004). Heuristics for model checking Java programs. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(4):260–276.

Holzmann, G. J. (2004). *The SPIN Model Checker*. Addison-Wesley.

Holzmann, G. J., Peled, D., and Yannakakis, M. (1996). On nested depth first search. In *Proceedings of the Second SPIN Workshop*, pages 23–32. American Mathematical Society.

Kamel, M. and Leue, S. (1998). Validation of the general inter-orb protocol (giop) using the spin model-checker. Technical report, Department of Electrical and Computer Engineering University of Waterloo.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4.

Lafuente, A. L. (2003). Symmetry Reduction and Heuristic Search for Error Detection in Model Checking. In *Workshop on Model Checking and Artificial Intelligence*.

Michael, C. C., McGraw, G., and Schatz, M. A. (2001). Generating software test data by evolution. *IEEE Transactions on Software Engineering*, 27(12):1085–1110.

Nakatani, T. (1997). Verification of a group address registration protocol using promela and spin.

Shi, Y. and Eberhart, R. (1998). Parameter selection in particle swarm optimization. *Evolutionary Programming*, 7:611–616.

Shi, Y., Eberhart, R., Team, E., and Kokomo, I. (2001). Fuzzy adaptive particle swarm optimization. In *Congress on Evolutionary Computation*, volume 1.

Valmari, A. (1998). *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, chapter The state explosion problem, pages 429–528. Springer.

AUTHOR BIOGRAPHIES

MARCO FERREIRA is professor in the Escola Superior de Tecnologia e Gestão de Leiria, Portugal. Nowadays he is a PhD student in the University of Extremadura. His main research interests is the application of metaheuristic algorithms to model checking.

FRANCISCO CHICANO is assistant professor in the University of Málaga, Spain. He received the PhD degree in Computer Science from the same university in 2007. His main research lines include the application of metaheuristic algorithms to optimization problems and, in particular, to Software Engineering problems. His Web-page can be found at `neo.lcc.uma.es/staff/francis`.

ENRIQUE ALBA is tenure in the University of Málaga, Spain. He received the PhD degree in Computer Science from the same university in 1999. His main research lines focus on metaheuristic algorithms in general. His Web-page can be found at `www.lcc.uma.es/~eat`.

JUAN A. GOMEZ-PULIDO is professor in the University of Extremadura, Spain. He received the PhD degree in Computer Science from the Complutense University of Madrid in 1993. His main research interests are applications of reconfigurable hardware to accelerate evolutionary algorithms used in optimization problems. His Web-page can be found at `arco.unex.es/jangomez`.

AN ALGORITHM AND SOME NUMERICAL EXPERIMENTS FOR THE SCHEDULING OF TASKS WITH FAULT-TOLERANCY CONSTRAINTS ON HETEROGENEOUS SYSTEMS

Moustafa NAKECHBANDI
Jean-Yves COLIN
LITIS, Le Havre University,
5, rue Philippe Lebon, BP 540, 76058, Le Havre cedex, France.
{moustafa.nakechbandi, jean-yves.colin}@univ-lehavre.fr

KEYWORDS
DAG, Scheduling with communication, Fault tolerant, Heterogeneous systems.

ABSTRACT

In this paper, we propose an efficient scheduling algorithm for problems in which tasks with precedence constraints and communication delays have to be scheduled on an heterogeneous distributed system with an one fault hypothesis. Based on an extension of the Critical-Path Method CPM/PERT, our algorithm combines an optimal schedule with some additional tasks duplication, to provide fault-tolerance. Backup copies are not established for tasks that have already more than one original copy. The result is a schedule in polynomial time that is optimal when there is no failure, and is a good resilient schedule in the case of one server failure. We finally compare the optimal solutions with the resilient solutions found by this algorithm on several semi-random DAGs.

I. INTRODUCTION

Heterogeneous distributed systems have been increasingly used for scientific and commercial applications. Recent examples of such applications include Automated Document Factories (ADF) in banking environments where several hundred thousands documents are produced each day on networks of several multiprocessors servers. Or high performance Data Mining (DM) systems (Palmerini 2004) that need to process very large data collections using very time-consuming algorithms. Or Grid Computing systems (Ruffner et al. 2003, Venugopal et al. 2004) such as Computational Grids which focus primarily on very computationally-intensive operations, or Data Grids which control the sharing and management of large amounts of distributed data.

However, efficiently using these heterogeneous systems is a hard problem, because the general problem of optimally scheduling tasks is NP-complete, even when there are no communication delays (Kwok and Ahmad 1999, Garey and Johnson 1979). When the application tasks can be represented by Directed Acyclic Graphs (DAGs), many dynamic scheduling algorithms have been devised. For some examples, see (Maheswaran and Siegel 1998, Iverson and Özgüner 1998, Chen, and Maheswaran 2002). Also, several static algorithms for scheduling DAGs in meta-computing systems are described in (Colin and Chrétienne 1991, Topcuoglu et al. 1999, Alhusaini, et al. 1999, Kwok and Ahmad 1999). Most of them suppose that tasks compete for limited processor resources, and thus these algorithms are mostly heuristics. Problems with fault tolerant aspects are less studied. Reliable execution of a set of tasks is usually achieved by task duplication and backup copies (Qin and Jiang 2006, Randell 1975, Chen and Avizienis 1978, Girault, et al. 2004).

A very classical and useful tool to study static scheduling problems with DAG is the Critical Path Method (also known as CPM, or PERT method, or CPM/PERT) (Maheswaran and Siegel 1998). Using a relaxation of the constraint on the number of available processors, this method gives results such as a lower bound on the execution time (or makespan) of the application and lower bounds on the execution dates of all tasks of the DAG. Because of the relaxation, tasks can be executed as soon as possible. Improvements and limits of this method to distributed systems with communications delays may be found in (Colin and Chrétienne 1991, Colin et al. 1999, Nakechbandi et al. 2002), for example. In (Colin et al. 2005), we studied the problem of scheduling the tasks of a DAG on the servers of an heterogeneous system. There, the relaxation used in CPM/PERT was replaced by the dual relaxation that each server has no constraint on the number of tasks it can simultaneously process. That is, each server can simultaneously process a non limited number of tasks without loss of performances. Our goal was to compute a lower bound on the execution time of a realistic solution, and compute lower bounds on the execution dates of all tasks of the DAG. In (Nakechbandi et al. 2007), we further supposed that one server (and at most one) could suffer from a crash fault. The algorithm presented there improved on the one presented in (Colin et al. 2005) by adding backup copies to the optimal solution build.

The solution we propose now is simpler than the one presented in (Nakechbandi et al. 2007). Additionally, we present some numerical experiments and simulation results. This rest of this paper is divided into four main parts. In the first one, we present the problem, and in the second one, we present a solution to the problem. In the third part, we make some numerical experiments using randomly generated tasks graphs, comparing the optimal solutions with the resilient solutions found by this algorithm. Finally, in the fourth part, we discuss the advantages and disadvantages of the proposed solution.

II. THE CENTRAL PROBLEM

2.1 The Distributed Servers System

We call Distributed Servers System (DSS) a virtual set of geographically distributed, multi-users, heterogeneous or not, servers. Therefore, a DSS has the following properties: first, the processing time of a task on a DSS may vary from a server to another. The processing time of

Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

each task on each server is supposed known. Second, although it may be possible that some servers of a DSS are potentially able to execute all the tasks of an application, it may also be possible in some applications that some tasks may not be executed by all servers. In a DSS problem, we suppose that the needs of each task of an application are known, and that at least one server of the DSS may process it.

The classical CPM/PERT relaxation of the number of processors, is replaced in the DSS problem with the dual relaxation that each server has no constraint on the number of tasks it can simultaneously process. Thus we suppose that the concurrent executions of some tasks of the application on a server have a negligible effect on the processing time of any other task of the application on the same server.

The transmission delay of a result between two tasks depends on the tasks and on their respective sites. The communication delay between two tasks executed on the same server is supposed equal to 0.

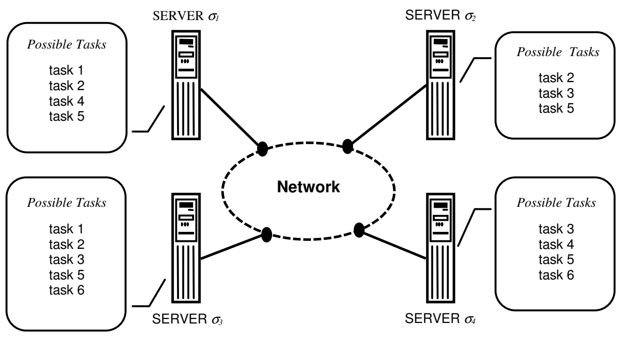


Fig. 1: Example of Distributed Servers System with the list of the executable services for each server.

2.2 Directed Acyclic Graph

An application is decomposed into a set of indivisible tasks that have to be processed. A task may need data or results from other tasks to fulfil its function and then send its results to other tasks. The transfers of data between the tasks introduce dependencies between them. The resulting dependencies form a Directed Acyclic Graph. Because the servers are not necessarily identical, the processing time of a given task can vary from one server to the next. Furthermore, the duration of the transfer of a result on the network cannot be ignored. This communication delay is function of the size of the data to be transferred and of the transmission speed that the network can provide between the involved servers. Note that if two dependent tasks are processed themselves on the same server, this communication delay is considered to be 0.

The central scheduling problem *P* on a Distributed Server System, is represented therefore by the following parameters:

- a set of servers, noted $\Sigma = \{\sigma_1, ..., \sigma_s\}$, interconnected by a network,
- a set of the tasks of the application, noted $I = \{1,..., n\}$, to be executed on Σ. The execution of task i, $i \in I$, on server σ_r, $\sigma_r \in \Sigma$, is noted i/σ_r. The subset of the servers able to process task i is noted Σ_i, and may be different from Σ,
- the processing times of each task i on a server σ_r is a positive value noted π_{i/σ_r}. The set of processing times of a given task i on all servers of Σ is noted $\Pi_i(\Sigma)$. $\pi_{i/\sigma_r} = \infty$ means that the task i cannot be executed by the server σ_r.
- a set of the transmissions between the tasks of the application, noted U. The transmission of a result of an task i, $i \in$ I, toward a task j, $j \in$ I, is noted (i, j). It is supposed in the following that the tasks are numbered so that if $(i, j) \in$ U, then $i < j$,
- the communication delays of the transmission of the result (i, j) for a task i processed by server σ_r toward a task j processed by server σ_p is a positive value noted $c_{i/\sigma_r, j/\sigma_p}$. The set of all possible communication delays of the transmission of the result of task i, toward task j is noted $\Delta_{i,j}(\Sigma)$. Note that a zero in $\Delta_{i,j}(\Sigma)$ mean that i and j are on the same server, i.e. $c_{i/\sigma_r, j/\sigma_p} = 0 \Rightarrow \sigma_r = \sigma_p$. And $c_{i/\sigma_r, j/\sigma_p} = \infty$ means that either task i cannot be executed by server σ_r, or task j cannot be executed by server σ_p, or both.

Let $\Pi(\Sigma) = \bigcup_{i \in I} \Pi_i(\Sigma)$ be the set of all processing times of the tasks of *P* on Σ.

Let $\Delta(\Sigma) = \bigcup_{(i,j) \in U} \Delta_{i,j}(\Sigma)$ be the set of all communication delays of transmissions (i, j) on Σ.

The central scheduling problem *P* on a distributed servers system DSS can be modelled by a multi-valued DAG $G = \{I, U, \Pi(\Sigma), \Delta(\Sigma)\}$. In this case we note $P=\{G, \Sigma\}$.

Example 1 : Figure 2 presents an example of DAG.

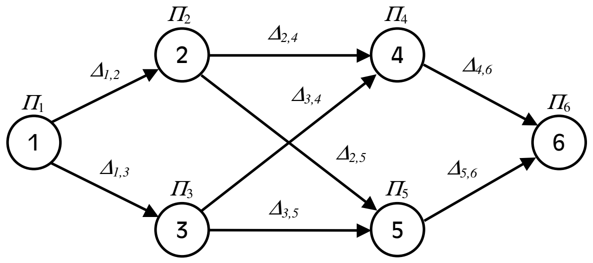


Fig. 2. Example of DAG : the Π_i vector on a node is the vector of the processing time of task i on the various servers, and $\Delta_{i,j}$ on an arc is the communication delays matrix between the two tasks depending on the servers that process them.

On this example, if we have 4 servers $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ and if $\Pi_1 = (\mathbf{3}, \infty, 2, \infty)$, then $\pi_{1/\sigma_1} = 3$. And $\pi_{1/\sigma_2} = \infty$, meaning that server σ_2 cannot execute task 2 etc.

On the same example, communications from task 1 to task 2 are given by matrix $\Delta_{1,2}$ in Fig.3.

	σ_1	$\boldsymbol{\sigma_2}$	σ_3	σ_4
σ_1	0	3	2	∞
σ_2	∞	∞	∞	∞
$\boldsymbol{\sigma_3}$	2	**3**	0	∞
σ_4	∞	∞	∞	∞

Fig. 3. Example of communication delays matrix $\Delta_{1,2}$ between task 1 and task 2.

In the matrix of Fig. 3, one can see that if task 1 is processed on server σ_3 and task 2 is processed on server σ_2, then $c_{1/\sigma 3,\ 2/\sigma 2} = 3$.

2.3. Definition of a feasible solution

We note PRED(i), the set of the predecessors of task i in G: $\mathrm{PRED}(i) = \{ k / k \in I \text{ et } (k,i) \in U \}$

And we note SUCC(i), the set of the successors of task i in G: $\mathrm{SUCC}(i) = \{ j / j \in I \text{ et } (i,j) \in U \}$

A feasible solution S for the problem P is a subset of executions $\{ i/\sigma_r, i \in I \}$ with the following properties:

- each task i of the application is executed at least once on at least one server σ_r of Σ_i,
- to each task i of the application executed by a server σ_r of Σ_i, is associated one positive execution date t_{i/σ_r},
- for each execution of a task i on a server σ_r, such that PRED(i) $\neq \varnothing$, there is at least an execution of a task k, $k \in$ PRED(i), on a server σ_p, $\sigma_p \in \Sigma_K$, that can transmit its result to server σ_r before the execution date t_{i/σ_r}.

The last condition, also known as the Generalized Precedence Constraint (GPC) (Colin et al. 1999), can be expressed more formally as:

$$\forall i/\sigma_r \in S \begin{cases} t_{i/\sigma_r} \geq 0 & \text{if } \mathrm{PRED}(i) = \varnothing \\ \forall k \in \mathrm{PRED}(i), \exists \sigma_p \in \Sigma_k / t_{i/\sigma_r} \geq t_{k/\sigma_p} + \pi_{k/\sigma_p} + c_{k/\sigma_p, i/\sigma_r} & \text{else} \end{cases}$$

It means that if a communication must be done between two scheduled tasks, there is at least one execution of the first task on a server with enough delay between the end of this task and the beginning of the second one for the communication to take place. A feasible solution S for the problem P is therefore a set of executions i/σ_r of all i tasks, $i \in I$, scheduled at their dates t_{i/σ_r}, and verifying the Generalised Precedence Constraints GPC. Note that, in a feasible solution, several servers may simultaneously or not execute the same task. This may be useful to generate less communications. All the executed tasks in this feasible solution, however, must respect the Generalized Dependence Constraints.

2.4. Optimality Condition

Let T be the total processing time of an application (also known as the makespan of the application) in a feasible solution S, with T defined as:

$$T = \max_{i/\sigma_r \in S} (t_{i/\sigma_r} + \pi_{i/\sigma_r})$$

A feasible solution S^* of the problem P modelled by a DAG $G = \{I, U, \Pi(\Sigma), \Delta(\Sigma)\}$ is optimal if its total processing time T^* is minimal. That is, it does not exist any feasible solution S with a total processing time T such that $T < T^*$.

III. THE DSS_1FAULT ALGORITHM

The algorithm proposed here, named DSS_1FAULT, has two phases: the first one is for the scheduling of original copies where we use the DSS-OPT algorithm (Colin et al. 2005) and the second one is for adding and scheduling additional backups copies when necessary.

3.1. Scheduling the original copies

We schedule original copies of tasks in our algorithm with the DSS-OPT algorithm (Colin et al. 2005). The DSS-OPT algorithm is an extension of CPM/PERT algorithms type to the distributed servers problem. In its first phase, it computes the earliest feasible execution date of each task on every server, and in its second phase it builds a feasible solution (without server fault) starting from the end of the graph with the help of the earliest dates computed in the first phase.

Let P be a DSS scheduling problem, and let $G = \{I, U, \Pi(\Sigma), \Delta(\Sigma)\}$ be its DAG.

One can first note that there is an optimal trivial solution to this DSS scheduling problem. In this trivial solution, all possible tasks are executed on all possible servers, as soon as possible, and their results are then broadcasted to all others servers. This is an obvious waste of processing power and communication resources, however, and something as optimal, but less wasteful in terms of used resources, is usually needed.

The first phase of the DSS_OPT routine, DSS_LWB(), goes from the initial tasks to the final ones, computing along the way the earliest feasible execution dates b_{i/σ_r} and earliest end date f_{i/σ_r}, for all possible executions i/σ_r of each task i of problem P.

The second phase of the DSS_OPT routine determines, for every task i that does not have any successor in G, i.e. task i is a "leaf" or final task, the execution i/σ_r ending at the earliest possible date f_{i/σ_r}. If several executions of task i end at the same smallest date b_{i/σ_r}, one is chosen, arbitrarily or using other criteria of convenience, and kept in the solution. Then, for each kept execution i/σ_r that has at least one predecessor in the application, the subset L_i of the executions of its predecessors that satisfy GPC(i/σ_r) is established. This subset of executions of predecessors of i contains at least an execution of each of its predecessors in G. One execution k/σ_p of every predecessor task k of task i is chosen in the subset, arbitrarily or using other criteria of convenience, and kept in the solution. It is executed at its earliest possible date b_{k/σ_p}. The examination of the predecessors is pursued in a recursive manner until the studied tasks do not present any predecessors in G.

3.2. Adding backup copies

The ADD_BACKUP_COPIES routine starts from tasks without any predecessors, similarly to DSS_LWB(), and proceed from there to the end of the DAG. First, if there is currently only one copy of a given task, it determines what is the worst possible delay it may encounter if a failure occurs on another server, while satisfying its GPC. It also determines the fastest server (not considering the server executing the only current copy of this task in the current solution) able to execute this task, and adds a backup copy on this server to the solution, again considering the worst possible delay resulting from this failure, while satisfying the GPC of this copy. Else the task has already several copies in the optimal solution, and the routine determines for each original copy of this task, what is the worst possible delay it may encounter if a failure occurs on another server, while satisfying its GPC.

The complete DSS_1FAULT algorithm is the following:

```
Input: $G = \{I, U, \Pi(\Sigma), \Delta(\Sigma)\}$
Output: A feasible solution with backup copies
DSS_1FAULT ()
    DSS_OPT()                          // first phase
    ADD_BACKUP_COPIES()                // second phase
end DSS_1FAULT
DSS_OPT()
    DSS_LWB ()
    $T = \max_{\forall i / SUCC(i)=\varnothing} \min_{\forall \sigma_r \in \Sigma_i} (r_{i/\sigma_r})$
    for all tasks $i$ such that SUCC($i$) = $\varnothing$ do
        $L_i \leftarrow \{ i/\sigma_r \,/\, \sigma_r \in \Sigma_i$ and $r_{i/\sigma_r} \le T \}$
        $i/\sigma_r \leftarrow$ keepOnefrom( $L_i$ )
        schedule ($i/\sigma_r$)
    end for
end DSS_OPT
DSS_LWB()
    for each task i where PRED($i$) = $\varnothing$ do
        for each server $\sigma_r$ such that $\sigma_r \in \Sigma_i$ do
            $b_{i/\sigma_r} \leftarrow 0$
            $r_{i/\sigma_r} \leftarrow \pi_{i/\sigma_r}$
        end for
        mark ($i$)
    end for
    while there is a non marked task $i$ such that
  all its predecessors $k$ in $G$ are marked do
        for each server $\sigma_r$ such that $\sigma_r \in \Sigma_i$ do
   $b_{i/\sigma_r} \leftarrow \max_{\forall k \in PRED(i)} ( \min_{\forall \sigma_p \in \Sigma_k} (b_{k/\sigma_p} + \pi_{k/\sigma_p} + c_{k/\sigma_p, i/\sigma_r}))$
            $r_{i/\sigma_r} \leftarrow b_{i/\sigma_r} + \pi_{i/\sigma_r}$
        end for
        mark ($i$)
    end while
end DSS_LWB
schedule($i/\sigma_r$)
    execute the task $i$ at the date $b_{i/\sigma_r}$ on the server $\sigma_r$
    if PRED($i$) $\neq \varnothing$ then
        for each task $k$ such that $k \in$ PRED($i$) do
            $L_k^{i/\sigma_r} \leftarrow \{ k/\sigma_q \;/\; \sigma_p \in \Sigma_K$ and
                $b_{k/\sigma_p} + \pi_{k/\sigma_p} + c_{k/\sigma_p, i/\sigma_r} \le b_{i/\sigma_r} \}$
            $k/\sigma_q \leftarrow$ keepOneFrom( $L_k^{i/\sigma_r}$ )
            schedule ($k/\sigma_q$)
        end for
    end if
end schedule
keepOneFrom($L_i$)
   return an execution $i/\sigma_r$ of task i in the list of the
   executions $L_i$.
end keepOneFrom.
ADD_BACKUP_COPIES()
    for each task $i$ such that PRED(i) = $\varnothing$ do
        if $i$ has only one copy scheduled then
          //compute one backup on the fastest server left, if
          // failure is on server of this copy
          Let $\sigma_r \neq \sigma_i$ be the fastest server able to execute $i$
          Execute a new backup copy of $i$ on $\sigma_r$ at date 0
        end if
        mark ($i$)
    end for
    while there is a non marked task $i$ such that all its
      predecessors $k$ in $G$ are marked do
        if $i$ has only one copy scheduled then
            Let $\sigma_i$ be the server executing the copy of $i$
            // First compute the delayed execution date of
            // task $i$ on this server, if the failure is on an
            // another server
            find the delayed execution date of the copy of $i$
             on $\sigma_i$ taking only into account the delayed
             execution dates of the copies and backups of
             each predecessor of $i$ to verify the GPC
            // Second compute one backup copy on the
            // fastest server left, if failure is on server of
            // primary
            Let $\sigma_r \neq \sigma_i$ be the fastest server able to execute $i$
            Execute a backup copy of $i$ on $\sigma_r$ taking only
            into account the delayed execution dates of the
            copies and backups of each predecessor of $i$ to
            verify the GPC
        else // $i$ has at least two copies scheduled, on
            // separate servers, of course
            // compute the delayed execution date of the
            // copy of task $i$ on each server, if the failure is
            // on an another server
            for each server $\sigma_i$ executing a copy of $i$ do
              Find the delayed execution date of the copy of
              $i$ on $\sigma_i$ taking only into account the delayed
              execution dates of the copies and backups of
              each predecessor of $i$ to verify the GPC
            end do
        end if
        mark ($i$)
    end while
end ADD_BACKUP_COPIES
```

Example 2 : If we consider the graph of the example 1, and using 4 servers the DSS_OPT gives the following optimal scheduling (Fig. 4.) :

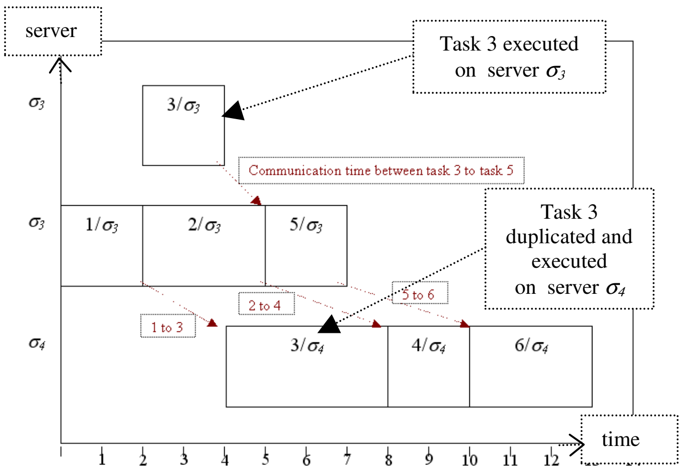


Fig. 4. Gantt chart given by DSS_OPT. The fact that task 3 is executed at the same time that task 2 on server σ_3 comes from the CPM/PERT relaxation.

By adding backup copies using ADD_BACKUP_COPIES we get the following fault-tolerance scheduling (Fig. 5.):

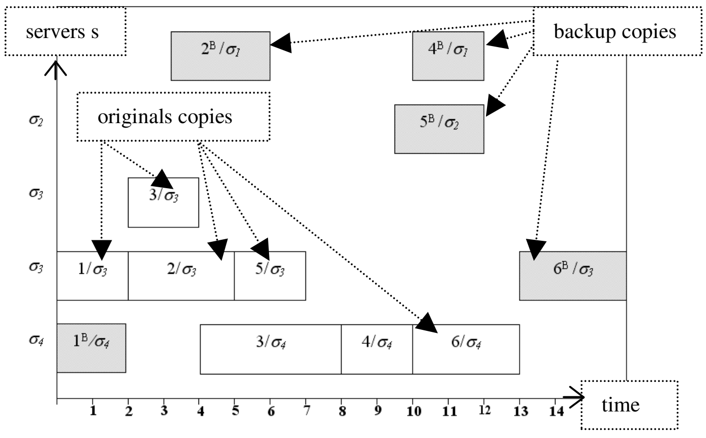


Fig. 5. Gantt chart given by DSS_1FAULT

Because the computed execution time of each task on each server is its earliest execution time on this server, and because only the copy with the earliest ending time, of each task without any successor, is used in the solution calculated by DSS_OPT() , and finally because all other copies are used only if they ensure that the final copies receives their data in time else they are not used, it follows that the feasible solution computed by DSS_OPT is optimal in execution time for the problem without server failure.

Lemma 1: The feasible solution calculated by the DSS_OPT algorithm is optimal if there is no server failure.

Because the copies in the DSS_1FAULT solution coming from the DSS_OPT solution will not be delayed if there is no server failure, and because additional backup will not be used in this case, then we have:

Theorem 1: The solution calculated by DSS_1FAULT is optimal if there is no server failure.

Also, in the final solution computed by DSS_1FAULT(), each task of the DAG has at least two copies (coming from the DSS_OPT() routine), or one copy (coming from the DSS_OPT() routine) and one backup copy (build by the ADD_BACKUP_COPY() routine) , always executed on different servers.

Furthermore, the execution date of each backup copy and the delayed execution date of each original copy coming from DSS_OPT is always evaluated by ADD_BACKUP_COPIES() taking into account the delayed execution dates of the copies and the execution dates of the backups copies of each predecessor, using the worst possible case of failure of a predecessor, we have:

Theorem 2: The solution calculated by DSS_1FAULT is feasible if there is at most one server failure.

The most computationally intensive part of DSS_OPT() is the first part DSS_LWB(). In this part, for each task *i*, for each server executing *i*, for each predecessor *j* of *i*, for each server executing *j*, a small computation is done. Thus the complexity of DSS_LWB() is O(n^2s^2), where *n* is the number of tasks in *P*, and *s* is the number of servers in DSS. Thus, the complexity of the DSS_OPT() algorithm is O(n^2s^2).

Similarly, in ADD_BACKUP_COPIES(), for each task *i*, for each copy of task *i* (at most one copy per server), for each predecessor *j* of *i*, for each copy of *j* (at most one per server), one small computation is done. Thus the complexity of ADD_BACKUP_COPIES() is bounded by O(n^2s^2), where *n* is the number of tasks in *P*, and *s* is the number of servers in DSS. Thus we have:

Theorem 3: The complexity of the DSS_1FAULT algorithm is O(n^2s^2).

IV. PERFORMANCE EVALUATION

To evaluate DSS_1FAULT, we have compared the fault tolerant solutions it generated on some classical problems and DAG to optimal solutions without fault tolerancy. These numerical experiments were done using simulations on three different kinds of graphs. The first one is a simple, semi-random, one level 'fork-join' DAG (see Fig. 6. a.), with limited parallelism. The second one is a regular simple two-dimensional grid DAG (see Fig. 6. b.), exhibited by some numerical applications, with lot of parallelism and very local communications. The last one is the "butterfly" DAG (see Fig6. c.) present in applications such as the FFT or shuffles algorithms, again with lot of parallelism, but a more complex communication pattern. The servers performances are independent random values for each task of the DAG, and so is each communication delay.

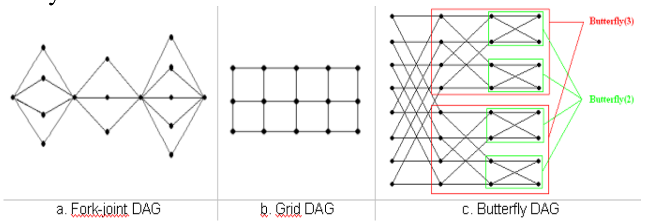


Fig. 6. Three different kind of graphs

4.1. Fork-Join DAG

As expected, this kind of DAG does show a very limited parallelism. On the Gantt chart example in Fig. 7, one can distinguish the original copies of the tasks on the left part of each server's simulated activity chart, and the added backup copies on the right part.

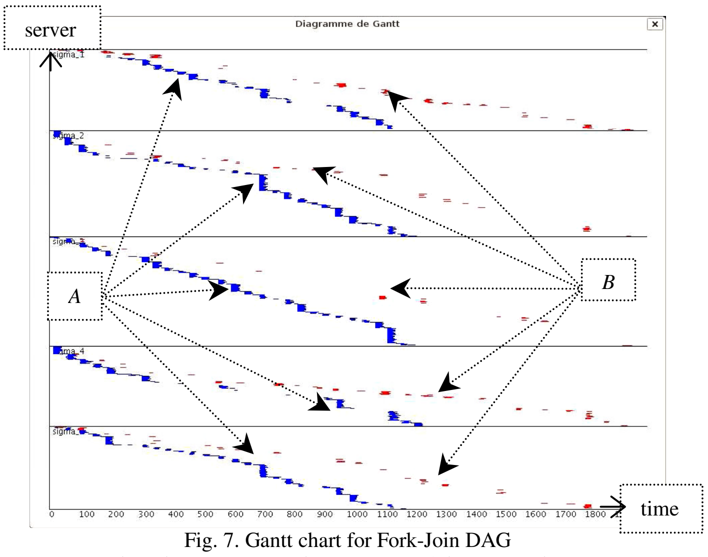


Fig. 7. Gantt chart for Fork-Join DAG

A. These lines represent the execution of the originals copies

B. These lines represent the execution of the backups copies.

4.2. 2-Dimensional grid DAG

This highly parallel DAG is much more efficiently executed on the servers. On the Gantt chart example in Fig. 8, the original copies of the tasks are grouped in the left part of each server's simulated activity chart, with the added backup copies spread more widely on the rights part. Although this is not clearly visible on the black and white figure, some added backup copies of the earliest tasks of the DG are present in the mist of the original copies.

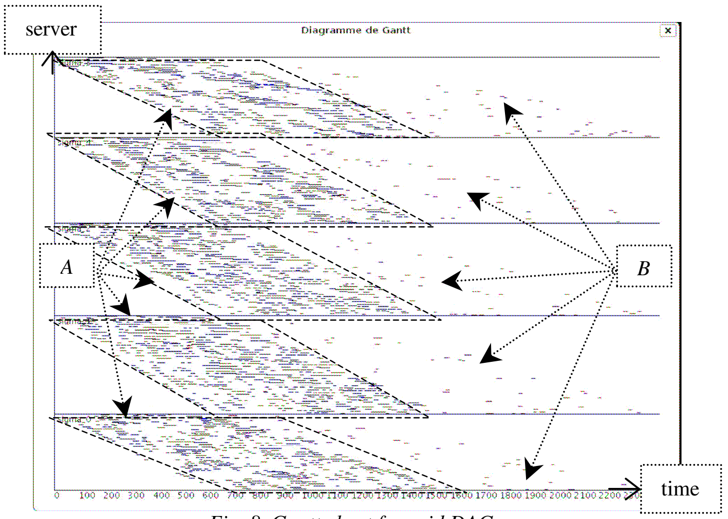


Fig. 8. Gantt chart for grid DAG
A. These lines represent the execution of the originals copies
B. These lines represent the execution of the backups copies.

4.3. Butterfly DAG

The Butterfly is a highly parallel DAG too. On the Gantt chart example of Fig. 9, the original copies are clearly distinguishable one the left, as bands. Because of the random nature of the server's performances, these bands tend to become fuzzier as time passes, however. The backup copies are scheduled later on the right part of the chart.

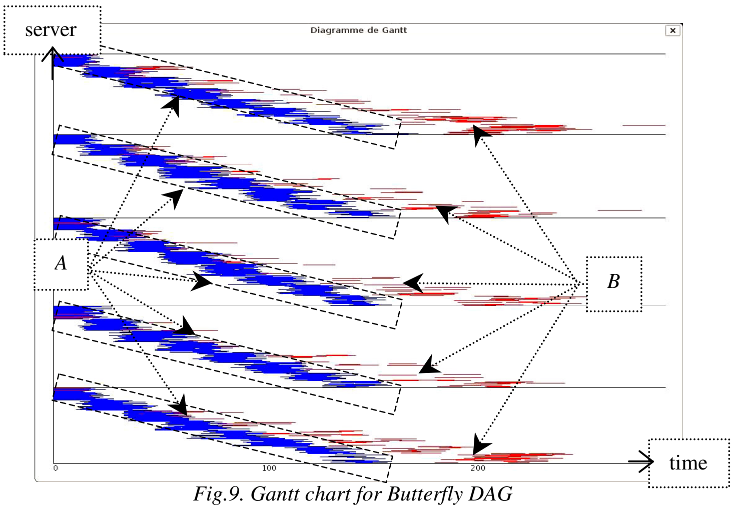


Fig.9. Gantt chart for Butterfly DAG
A. These lines represent the execution of the originals copies
B. These lines represent the execution of the backups copies.

4.4. Makespan with and without backup copies

In all three kinds of DAGs, it is found that the makespan average with backup copies is between 1.5 (usually) and 2 (at most) times the makespan without backup copies. For example, in the Butterfly DAGs, we obtained the following figure (Fig . 10). In this simulation the number of tasks varies from 10 to 1200 tasks and we have the average over 50 random DAGs.

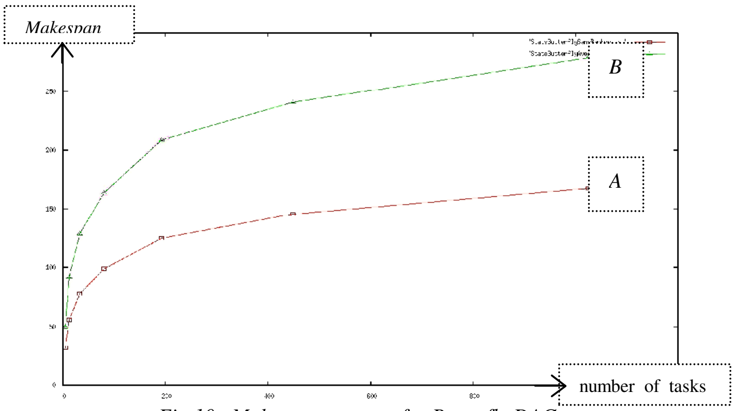


Fig.10. Makespan average for Butterfly DAGs
A. Makespan without backup, B. Makespan with backup,

V. ANALYSIS

This algorithm has two advantages:

- when no server fails, the DSS-1FAULT's solution is optimal as it uses the optimal solution computed by DSS-OPT.
- when there is a failure of one server, the DSS-1FAULT's solution is certain to finish correctly, because every tasks has two or more scheduled copies on different servers in the final solution. If more than one fault occur, the solution may still finish, but there is no guaranty there.

We do not establish backup copies for tasks which have already two or more original copies from the DSS-OPT algorithm scheduling to limit tasks duplication and processor. It also gives indications on the sensibility of an application to one server failure when compared to the solution without any server failure, because the makespan in the presence of one failure is a worst case analysis.

The model of failure, as it features at most 1 crash, may seem poor. However, if the probability of any failure is very low, and the probabilities of failure are independent, then the probability of two failures will be much smaller indeed. Furthermore, the algorithm may be extended to 2 or more failures, by using two or more backup copies per task. The efficiency of this kind of solution to the "k-failures" problem is not investigated, yet.

Finally, the solution solved by this new algorithm uses the classical CPM/PERT relaxation, namely that an unbounded number of tasks may be processed on each server in parallel without any effect on the tasks' processing time, in the same sense that the CPM/PERT method do not consider resources constraints in order to get earliest execution dates. This relaxation is not far from the reality, if each server is a multiprocessors architecture. Or if each server is a time-shared, multi-users system with a permanent heavy load coming from other applications, and the tasks of an application on each server represent a negligible additional load. In other cases, the same way these CPM/PERT results are used in some real-life systems as the priority values of tasks in some list-scheduling algorithms, the result found by our algorithm may be used as the first step of a list scheduling algorithm, in which the earliest execution dates of primary and backup copies are used as priority values to schedule these copies on the servers of a real-life system.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a polynomial scheduling algorithm in which tasks with precedence constraints and communication delays have to be scheduled on an heterogeneous distributed system environment with one fault hypothesis. To provide a fault-tolerant capability, we employed primary and backup copies. But no backup copies were established for tasks which have more than one primary copy.

The result have been a schedule in polynomial time that gives earliest execution dates to copies of tasks when there is no failure, and is a good resilient schedule in the case of one failure. Performance evaluation on some DAGs gave an increase in case of one server failure in makespan of 1.5 to 2 times the optimal makespan without server failure.

The execution dates of the original and backup copies may be used as priority values for list scheduling algorithm in cases of real-life, limited resources, and systems.

In our future work, we intend to study the same problem with sub-networks failures. Also, we intend to consider the problem of non permanent failures of servers. Finally, we want to consider the problem of the partial failure of one server, in which one server is not completely down but loses the ability to execute some tasks and keeps the ability to execute at least one other task.

REFERENCES

A. H. Alhusaini, V. K. Prasanna, C.S. Raghavendra. 1999. “A Unified Resource Scheduling Framework for Heterogeneous, Computing Environments”, *Proceedings of the 8th IEEE Heterogeneous Computing Workshop,* Puerto Rico, pp.156- 166.

R.E. Bellman. 1957. “Dynamic Programming”. *Princeton University Press, Princeton, New Jersey.*

H. Chen, M. Maheswaran 2002. “Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems”, *Proceedings of the 11th IEEE Heterogeneous Computing Workshop* ,pp. 88b-98b, Fort Lauderdale.

L. Chen, A. Avizienis. 1978. “N-version programming: a fault tolerant approach to reliability of software operation”, *Proceeding of the IEEE Fault-Tolerant Computing Symposium,* pp. 3-9.

J.-Y. Colin, P. Chrétienne 1991. "Scheduling with Small Communication Delays and Task Duplication", *Operations Research,* vol. 39, n o 4, 680684.

J.-Y. Colin , M. Nakechbandi, P. Colin, F. Guinand. 1999. “Scheduling Tasks with communication Delays on Multi-Levels Clusters”, *PDPTA'99 : Parallel and Distributed Techniques and Application,* Las Vegas, U.S.A..

J.-Y. Colin , M. Nakechbandi, P. Colin. 2005. "A multi-valued DAG model and an optimal PERT-like Algorithm for the Distribution of Applications on Heterogeneous, Computing Systems", *PDPTA'05,* Las Vegas, Nevada, USA, June, pp. 876-882.

M.J. Flynn. 1972. “Some computer organization and their effectiveness.”, *IEEE Transactions on Computer,* pp. 948-960, September.

M.R. Garey and D.S. Johnson. 1979. ”Computers and Intractability, a Guide to the Theory of NP-Completeness”, *W. H. Freeman Company,* San Francisco.

A. Girault, H. Kalla, and Y. Sorel. J 2004. “A scheduling heuristics for distributed real-time embedded systems tolerant to processor and communication media failures”. *International Journal of Production Research*, 42(14):2877-2898.

M. Iverson, F. Özgüner. 1998. “Dynamic, Competitive Scheduling of Multible DAGs in a Distributes Heterogeneous Environment”, *Proceedings of the 7th IEEE Heterogeneous Computing Workshop (HCW'98),* pp. 70–78, Orlando, Florida.

Yu-Kwong Kwok, and Ishfaq Ahmad. 1999. “Static scheduling algorithms for allocating directed task graphs to multiprocessors”, *ACM Computing Surveys* (CSUR), 31 (4): 406 – 471.

M. Maheswaran and H. J. Siegel. 1998. “A Dynamic matching and scheduling algorithm for heterogeneous computing systems”, *Proceedings of the 7th IEEE Heterogeneous Computing Workshop(HCW '98),* pp. 5769, Orlando, Florida.

M. Nakechbandi, J.-Y. Colin , C. Delaruelle. 2002. “Bounding the makespan of best pre-scheduling of task graphs with fixed communication delays and random execution times on a virtual distributed system”, *OPODIS02,* Reims; pp. 225-233.

M. Nakechbandi, J.-Y. Colin, J.B. Gashumba. 2007. "An efficient fault-tolerant scheduling algorithm for precedence constrained tasks in heterogeneous distributed systems"; *CIS2E06 International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering*, December, 2006. Published in : Innovations & advanced techniques in computer & information sciences & engineering, Springer, 06-2007, pp 301-307.

P. Palmerini. 2004. “On performance of data mining: from algorithms to management systems for data exploration*”, PhD. Thesis: TD-2004-2, Universit`a Ca’Foscari di Venezia.*

X. Qin and H. Jiang. 2006. “A Novel Fault-tolerant Scheduling Algorithm for Precedence Constrained Tasks in Real-Time Heterogeneous Systems” , *Parallel Computing*, vol. 32, no. 5-6, pp. 331-356.

B. Randell. 1975. “System structure for software fault-tolerance”, *IEEE Trans. Software Eng.* 1(2,) pp. 220-232.

Christoph Ruffner, Pedro José Marrón, Kurt Rothermel. 2003 “An Enhanced Application Model for Scheduling in Grid Environments”, *TR-2003-01, University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS).*

H. Topcuoglu, S. Hariri, and M.-Y. Wu. 1999. “Task scheduling algorithms for heterogeneous processors”. *In 8th Heterogeneous Computing Workshop (HCW’ 99),* pp. 3–14.

Srikumar Venugopal, Rajkumar Buyya and Lyle Winton. 2004. “A Grid Task Broker for Scheduling Distributed Data-Oriented Applications on Global Grids”, *Technical Report, GRIDS-TR-2004-1, Grid Computing and Distributed Systems Laboratory,* University of Melbourne, Australia.

AUTHOR BIOGRAPHIES

Moustafa NAKECHBANDI is Associate Professor at the University of Le Havre, France. He received a Ph.D (1984) in Computer Science from Besançon University. His research interests are in optimization problems relative to parallel computing and in fault-tolerant scheduling.

Jean-Yves COLIN is Assistant Professor at the University of Le Havre, France. He received a Ph.D (1989) in Computer Science from Paris 6 University. His research interests include scheduling in heterogeneous distributed systems, and optimization of parallel programs.

Structure and Latency Analyses for High Performance Computing System Based on Asynchronous Optical Packet Switching

Zhao Jun and Sun Xiaohan
Department of Electronic Engineering
Southeast University
Nanjing 210096, China
E-mail: zhaojun1115@seu.edu.cn
xhsun@seu.edu.cn

KEYWORDS
High performance computing system, Asynchronous optical packet switching, Distributed management structure, Stability, Latency

ABSTRACT

A novel high performance computing system based on optical packet switching and optical multicast technologies is presented. Distributed management architecture is used to alleviate the storing and computing pressures of every stage, which is easy to realize all-optical scalability. Asynchronous switching mode is accepted at every stage for high-speed and huge-capacity burst services transmission. The system scale and the stability features are analyzed, and a two stage system which interconnects 38,400 CPUs is adopted. Moreover, the average packets waiting latencies caused by the scheduling units and the recycling-fiber-delay-line based collision resolution units are simulated as 12.9ns and 0.63ns, respectively.

1. INTRODUCTION

High performance computing systems (HPCS) use high-bandwidth and low-latency links to interconnect huge amounts of distributed microprocessors for providing timely exchanging of high-speed and large-capacity services[1,2]. The BlueGene/L System, a joint development of IBM and the Department of Energy's (DOE) National Nuclear Security Administration (NNSA), has been significantly scaled up from 65,536 to 106,496 nodes and now has achieved a Linpack benchmark performance of 478.2 TFop/s. Accordingly, the development of HPCS is in the tendency of higher speed and more processors, so more pressures are placed on the performance of the interconnection network [3].

The traditional electrical link is becoming the bottleneck for high-speed and high-capability data transmission, on the other hand, the static optical interconnection technology, such as optical circuit switching, can not meet the need for burst services transmission [4]. Optical packet switching (OPS) technology which has the effective bandwidth utilization ability and fine exchanging granularity is becoming the most promising one in next generation optical network. HPCS based on OPS technology can improve the parallel exchanging ability for the system and is very attractively in HPCS design [5, 6]. However, the switching unit as well as the collision resolution module are still not mature, also, synchronous switching technology and centralized management structure are commonly used which are not easy to be constructed and not favorable for system scalability and will also induce more queue latencies [7].

This paper proposes a HPCS system based on asynchronous OPS and optical multicast (AOPS & M-HPCS) technologies. Distributed management structure is used to alleviate the storing and computing pressures of every stage and can easily to realize all-optical scalability. Multiple CPUs share one optical transceiver which can increase the system scale. The optical-switch (OS) based on semiconductor optical amplifiers (SOA) combining with optical splitter are used as switching units, meanwhile, the recycling fiber delay lines (Rec-FDL) are used as collision resolution units. The system scale, stability and delays induced by scheduling-unit and Rec-FDL are simulated.

2. ARCHITECTURE ANALYSES FOR AOPS & M-HPCS SYSTEM

Figure 1 shows two-stage AOPS & M-HPCS system structure where the master-node which has higher level controls n slave-nodes only, meanwhile, each slave-node manages n scheduling-units (SU) which controls m CPUs respectively and ordinal sends packets from CPUs to the optical transceiver (TX/RX). By using the 80 wavelength dense wavelength division multiplexing (DWDM) links with single channel capacity of 40 Gbit/s, the value of n can be confirmed as 80, and the two-stage system can interconnect $6{,}400\times m$ CPUs.

The Edge-note (EN) which constituted by electrical buffers and packets assembly units can assemble the data from CPUs into packet-payloads, which will be exchanged by the optical switching units (OSU) in optical domain. Information such as storage capacities and computing abilities needed by each service is carried in the packet-labels which will be extracted and processed electronically by the controlling-units (CU).Each packet is transmitted asynchronously for

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

decreasing the queue delay, and is exchanged firstly among CPUs inside the slave node where it is generated in, which can reduce the transmission latencies and arbitration complexities of every stage. If no destinations can be found in this slave-node, the packet will be exchanged to the master-node, and then will be allocated to other slave-nodes.

The SOA has lots of advantages such as nanosecond high-speed switching ability, low controlling voltage, easily been integrated, and so on. Consequently, it can improve the switching speed by using the SOA based optical-switches and a 1×81 optical splitter to construct the OSU. 80 of the splitter output are used to interconnect the SUs in the slave-node, while the additional one is dedicated to the master-node connection. A packet can be switched to one or more destinations by controlling the on-off states of all the SOAs which can realize the optical multicast.

It may cause wrong receiving if two or more packets arrive at the SOA-switch in its once tuning-time. Here we use a 2×2 optical switch (OS) combine with a Rec-FDL as the collision resolution units, which will send the lower priority packets into the Rec-FDL for delaying. The tunable wavelength converters (TWC) can avoid wavelength conflicts between the downstream signals and the slave-node signals and can also avoid collisions among upstream signals in the master-node.

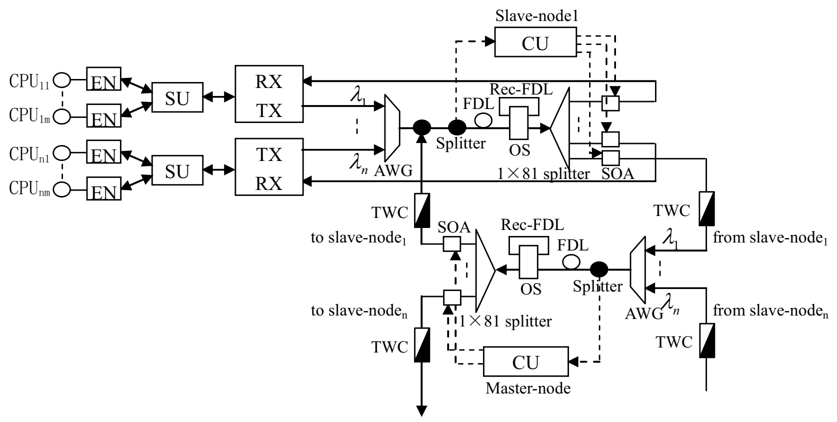


Figure 1: The Infrastructure for Two-stage AOPS＆M-HPCS (EN: Edge Node; SU: Scheduling Unit; RX/TX: Optical Transceiver; CU: Controlling Unit; Rec_FDL: Recycling Fiber Delay Line; OS: Optical Switch).

3. PERFORMANCE ANALYSES FOR AOPS＆M-HPCS SYSTEM

3.1. Latencies Caused by the Scheduling Units

The SU transmit the packets from all the attached ENs in a polling mode. The inquiry time *t′* for one EN is *L*/*V*, where the packet length *L* is 256Byte in this paper, and the optical transmitter rate *V* is 40Gbit/s. If the SU finds no packets in the EN, then *t′* equals to 0, and the probability distribution for *t′* is approximated expressed as follows:

$$P=\begin{cases}1-e^{-\lambda mL/V}, & t'=L/V\\ e^{-\lambda mL/V}, & t'=0\end{cases} \qquad (1)$$

Here, λ represents the new packets arriving rate. Therefore, the average value for *t′* can be obtained from (1), which is shown as $\bar{t}=(1-e^{-\lambda mL/V})(L/V)$, then the polling cycle for all the *m* ENs is $m\bar{t}$. If one EN generates *j* packets in this period, then the packets longest waiting latency is $T_1=jm\bar{t}$, and this value will change in the subsequent polling cycle, the fluctuation value Δt_1 has two possibilities, which are shown as follows: $\Delta t_1=-m\bar{t}$, there have no new packets arrive at this EN in time $-m\bar{t}$, while $\Delta t_1=im\bar{t}$ (i=1,2,…) represent that *i* new packets are generated. Therefore, the mean value for Δt_1 is:

$$E(\Delta t_1)=-m\bar{t}e^{-\lambda m\bar{t}}+\sum_{i=1}^{\infty}im\bar{t}\frac{(\lambda m\bar{t})^i}{i!}e^{-\lambda m\bar{t}} \qquad (2)$$

The system is stable only if the longest waiting latency is gradually decreasing. Accordingly, the expression of $E(\Delta t_1)<0$ must be contented for ensuring the system to be stable.

Assuming that the time required for T_1 to decrease to zero is represented as $T_1(j)$, which is schematically depicted by the calculation flow-chart as shown in figure 2, so the average latencies for the packets waiting in the SU is given by:

$$T_{SU}=\sum_{j=1}^{\infty}\frac{(\lambda m\bar{t})^j}{j!}e^{-\lambda m\bar{t}}T_1(j) \qquad (3)$$

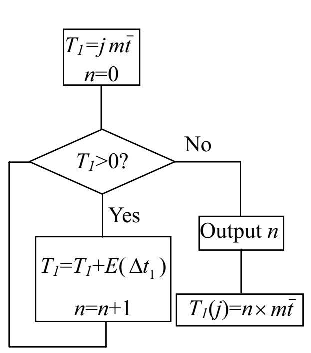


Figure 2: Calculation Flow-chart for $T_1(j)$

3.2. Latencies Caused by the Rec-FDL

A wrong receiving occurs whenever j+1 packets arrive simultaneously at a SOA-switch inside its once tuning-time t, then j lower priority packets will be switched into the Rec-FDL, with the arriving as well as departing time shown in figure 3. Here, ΔT represents the interval of the packets arriving time, T' represents the time-slot that may generate packets in the period of t since the packet p_j enter the Rec-FDL.

The longest waiting latency for the j packets is $T_2=jt$, which will be changed if other packets arrive at the Rec-FDL inside the next period of T'. There also have two values for the fluctuation-value (Δt_2), one is $\Delta t_2=-t$, represents that no packets arrive at the Rec-FDL, and the other is $\Delta t_2=it$ (i=1,2,…), which shows that i packets are sent into the Rec-FDL. Accordingly, the mean value for Δt_2 can be shown as follows:

$$E(\Delta t_2) = -te^{-80\lambda' T'} + \sum_{i=1}^{\infty} it \frac{(80\lambda' T')^i}{i!} e^{-\lambda' T'} \qquad (4)$$

Where, λ' represents the departing rate for the packets from the SU, and the value of 80 is the number of the SUs controlled by one slave-node.

If there have i CPUs generate packets in once polling cycle $m\bar{t}$, then $\lambda' = i/(m\bar{t})$, and the mean value can be shown in (5).

$$\lambda' = 80\sum_{i=0}^{m} \binom{m}{i} (1-e^{-\lambda m\bar{t}})^i (e^{-\lambda m\bar{t}})^{m-i} i/(m\bar{t}) \qquad (5)$$

The system is stable only if the T_2 can decrease to zero. Accordingly, the other stability condition for the system is $E(\Delta t_2)<0$.

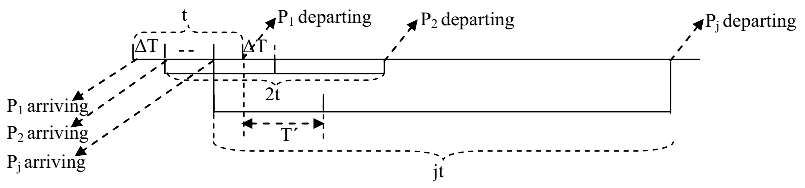


Figure 3: The Arriving and Departing Time for the j Packets inside the Rec-FDL.

The average waiting latencies ($T_{Rec-FDL}$) induced by the Rec-FDL can be expressed in equation (6), where $T_2(j)$ represents the time required for T_2 to decrease to zero, with the calculation flow-chart shown in figure 4.

$$T_{\mathrm{Rec-FDL}} = \sum_{j=1}^{\infty} \frac{(\lambda' t)^j}{j!} e^{-\lambda' t} T_2(j) \qquad (6)$$

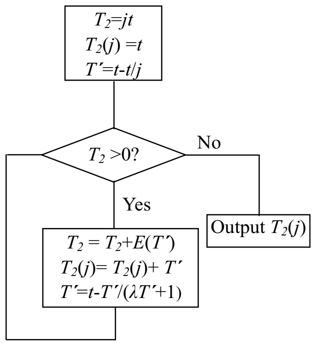


Figure 4: Calculation Flow-chart for $T_2(j)$

4. SIMULATION ANALYSES

The simulation parameters are shown as follows: t=2ns, T'=1ns. According to above analyses, $E(\Delta t_1)<0$ as well as $E(\Delta t_2)<0$ must be satisfied for ensuring the stabilities of the system. It can be seen from figure 5 that the number of the CPUs (m) controled by the SU varies inversely with λ, the maximum value of m is 19 as $\lambda=1\times10^6$packets/s, and is 29 when λ decreases to 0.6×10^6packets/s. Accordingly, both the system scale and the packets arriving rate must be considered in this AOPS & M-HPCS design. Moreover, these values can always ensure $E(\Delta t_2)<0$ according to figure 6. In this paper, the value of $\lambda=1\times10^6$packets/s and m=6 are adopted, therefore, the two-stage system can interconnect 38,400 CPUs.

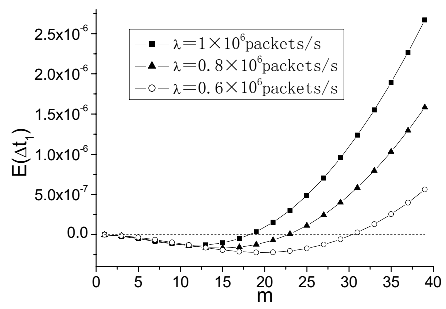


Figure 5: Relations of $E(\Delta t_1)$ versus m and λ

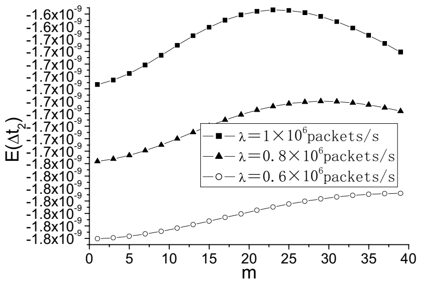


Figure 6: Relations of $E(\Delta t_2)$ versus m and λ

As the system scale increases with m, and the number of the packets, which enter the system simultaneously, are also increased with λ, therefore, the larger value of the $m\lambda$ will cause more collisions, which will induce higher blocking rate and more waiting latencies. From figure 7 and figure 8 we can see that when $m\lambda=6\times10^6$packets/s, the latencies caused by the SU collisions (T_{SU}) and the Rec-FDL collisions ($T_{Rec\text{-}FDL}$) equals to 12.9ns and 0.63ns, respectively.

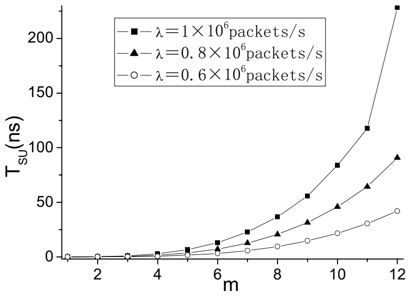


Figure 7: Relations of T_{SU} versus m and λ

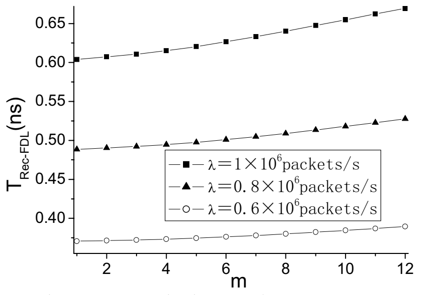


Figure 8: Relations of $T_{Rec\text{-}FDL}$ versus m and λ

Furthermore, the latencies caused by the Rec-FDL in the master-node can be analyzed with the same methods as described above, and the simulation results are influenced by the system parameters, such as the packet length, optical transceiver rate, as well as the tuning-time for the SOA-switch, and so on.

5. CONCLUSIONS

A novel HPCS based on asynchronous OPS and optical multicast technologies is presented. The dense wavelength division multiplexing transmission technologies together with the multistage distributed management topologies are used to construct a scalable interconnection network, which is suitable for timely and stochastic accesses for high-speed and massive burst services. The collision resolution unit based on the Rec-FDL are described in detail. The stabilities and the scale of the system are analyzed, and the latencies caused by the the Rec-FDL and the collisions in the scheduling units for a 38,400 CPUs interconnection system are simulated, which is 0.63ns and 12.9ns, respectively. Moreover, the experimental researches will be done later.

REFERENCES

[1] Hawkins C, Small B A, Wills D S, et al, "The data vortex, an all optical path multicomputer interconnection network [J]", *IEEE Transactions on Parallel and Distributed Systems*, 2007, 18(3), pp. 409-420.

[2] Hawkins C, Wills D S, "Impact of number of angles on the performance of the data vortex optical interconnection network [J]", *Journal of Lightwave technology*, 2006, 24(9), pp. 3288-3294.

[3] Drost R, Forrest C, Guenin B, et al, "Challenges in building a flat-bandwidth memory hierarchy for a large-scale computer with proximity communication [C]", *Proceedings of the 13th Symposium on High Performance Interconnects*, Palo Alto, CA, August 2005, Page(s):13-22.

[4] Barker K J, Benner A, Hoare R, "On the feasibility of optical circuit switching for high performance computing systems [C]", *IEEE, Conference on Supercomputing*. Seattle, USA, 2005, pp. 1-22.

[5] Masetti F, Chiaroni D, Dragnea R, et al, "High-speed high-capacity packet-switching fabric: a key system for required flexibility and capacity [J]", *Journal of Optical Networking*, 2003, 2(7), pp. 255-265.

[6] Hemenway R, Richard R. Grzybowski, "Optical-packet-switched interconnect for supercomputer applications [J]", *Journal of Optical Networking*, 2004, 3(12), pp. 900-913.

[7] Minkenberg C, Abel F, Muller P, et al, "Designing a Crossbar Scheduler for HPC Applications [J]", *IEEE Micro* , 2006, 26(3), pp. 58-71.

AUTHOR BIOGRAPHIES

JUN ZHAO, he is currently pursuing his Ph.D degree in the lab of optical communications, department of electronics engineering, Southeast University, Nanjing, China. His research interests include the high performance computing system design, optical packet switching, high speed optical signal processing, and transparent optical networks.

XIAOHAN SUN, a professor of electronics engineering, Southeast University, Nanjing, China. She was the visiting professor at Research Lab of Electronics, MIT, USA from 2002 to 2004. Her current research interests are photonic devices, high-speed photonics systems and next generation optical networks (NGON), including (a) optical pulse propagation in WDM systems influenced by nonlinear effects, PMD, crosstalk and so on, (b) management and control for NGON, (c) reliability and survivability for NGON, (d) Optical fiber sensor technology and optical imaging, and (e) design and measurement for semiconductor materials based PLCs.

HPCS 2008 POSTER and Work in Progress Session (Partial)

COMMUNICATION COST OF A MATRIX PRODUCT ON SUPER-HYPERCUBE ARCHITECTURE

Maryam Amiripour and Hamid Abachi
Department of Electrical and Computer Systems Engineering
Monash University
Australia
E-mail: maryam.amiripour@eng.monash.edu.au

KEYWORDS
Communication Cost, Hypercube, Super Hypercube, Dynastatic, Matrix Product

ABSTRACT

Processor allocation and the task scheduling technique in parallel processing systems play a significant role in improving the performance of a message-passing architecture. Adapting the right algorithms and further improvements in areas such as time complexity, execution time, speed up and synchronization mechanisms undoubtedly facilitates implementation of advanced applications on a parallel processing system. These applications include but are not limited to DNA computing, artificial immune systems and optical computing to name a few. This paper highlights the communication cost related to a Super-Hypercube topology for being a subclass of traditional Hypercube architecture.

Furthermore, a particular reference is made to the mathematical modeling of Hypercube and Super-Hypercube architectures. Finally, graphical presentations are carried out based on mathematical calculations to address the advantage of Super-Hypercube topology.

INTRODUCTION

Parallel processing systems are commonly applied in areas such as military, space, signal processing, image processing and pattern recognition that require high computational power. These parallel processing systems could be implemented to solve many engineering problems that suffer from lack of high reliability, performance, flexibility and compatibility, availability, portability, and low in cost and size.

Hypercube architectures perform well for a large range of problems. It is well suited for both general-purpose and special-purpose applications. They are mainly used in matrix operations, sorting, signal and image processing where extensive data processing is required (Walker 1998**.** Figure 1 illustrates the general form of this architecture.

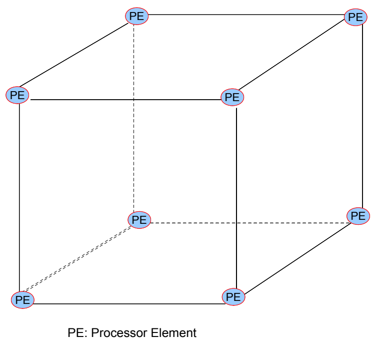


Figure 1: 3 D Hypercube Architecture

In Hypercube architectures when communication between two indirectly connected Processing Elements (PEs) is required, the message has to cross one or more hyper-planes and go through intermediate PEs before reaching its destination. The PEs involved are required to compute and handle message-passing, which reduces the overall computational power and performance. In addition, if one of the intermediate PEs is faulty or busy performing tasks, there will be a significant downtime in communication between the source and destination PEs.

In order to overcome Hypercube limitations such as routing and expandability, an enhanced version of the Hypercube architecture namely Super-Hypercube was developed (Abachi 1997). This architecture includes applying a Router (R) to the basic Hypercube. This router acts as a crossbar switch, which can provide a communication path between two indirect PEs. Its usage in conjunction with SGI (Silicon Graphics Inc.) products relieves the processor of the routing task and provides more efficient computing activities. Figure 2 shows the basic principle of this architecture.
.
As reported in (Grama et al. 2003), interconnection networks can be classified as either dynamic or static. The former interconnection is designed by using switches to connect PEs together. On the other hand, the latter deals with the networks consist of point-to-point communication links among PEs. In the Super-Hypercube topology, the adjacent PEs are directly connected together without use of the Router (R) and indirect PEs are connected together through a Router (R).

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

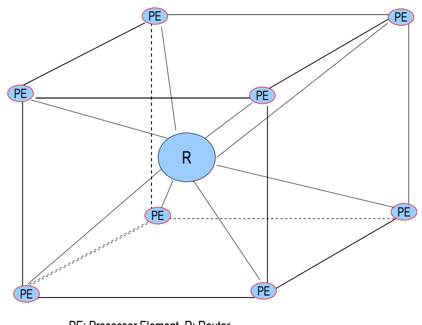


Figure 2: Super- Hypercube Architecture

The Super-Hypercube outlined in this paper uses combination of both categories (dynamic and static). In this paper it is coined as **Dyna**mic-**Static** (**Dynastatic**) interconnection.

The aim of this research is to identify the advantages of communication cost for Dynastatic Super-Hypercube architecture when is compared with the traditional Hypercube topology.

MATRIX MULTIPLICATION ON DISTRIBUTED MEMORY SYSTEMS

In many applications, matrix multiplication involves dealing with different sizes (squares vs. rectangular) and may include the communication cost. The size of the matrix can significantly impact on the performance of parallel matrix multiplication algorithm (Dongarra et al 2007).
This section outlines the general mathematical model for square matrix multiplication.

Basic Concepts, Definitions and Assumptions

Let A and B be matrices of size $m \times n$ and $n \times p$ respectively. The product of A and B is a matrix of size $m \times p$ which denoted by AB and is given by:
$(\mathrm{AB})_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} = a_{i1}b_{1j} + \cdots + a_{n1}b_{1n}$
for each pair i and j with $1 \leq i \leq m$ and $1 \leq j \leq p$.

For the purpose of this paper, we perform a matrix multiplication on a DMS which is more favorable than shared memory. In doing so, we consider the following definitions (Li 2007):

Definition 1: In order to construct our mathematical we consider that a DMS can support one-to-one communication in $T_{o-oc}(P)$ time unit. For this purpose, a fast and scalable parallel matrix algorithm is required.

Definition 2: We assume that a DMS consists of P PEs $\{p_0, p_1, \cdots, p_{p-1}\}$ with their own local memory$\{\mathrm{m}_0, m_1, \cdots, m_{p-1}\}$. In addition, we consider that PEs have the capability of communicating with each other through message-passing scheme. Moreover, the computation and communication are globally synchronized into steps. That is to say, a step is either a computation step or a communication step. In former, each PE has a capability of performing a local logic/ arithmetic operation or in worse scenario is idle and it utilises constant amount of time.
In latter, PEs could communicate with one another bio-directionally via an interconnection network. In this case, a communication step can be mathematically expressed as:
$((\Psi(0), w_0), (\Psi(1), w_1), \cdots, (\Psi(p-1), w_{p-1}))$ where $0 \leq i \leq p$. This results in PE p_j sending a value w_j to PE $\Psi(P_J)$ and Ψ is a mapping
$\Psi\colon \{0, 1, \cdots, p-1\} \rightarrow \{-1, 0, 1, \cdots, p-1\}$.

Definition 3: If PE p_j doesn't send any messages during the communication step, then $\Psi(j) = -1$ and w_j is undefined.
However, in a practical situation, there is at most one j such that $\Psi(j) = i$. This implies each PE can maximum receive one message in one-to-one communication step. This also reviles that based on definition 1, the DMS supports the above communication step in $T_{o-oc}(P)$ time frame.
From a practical application point of view, in the busiest communication step, every PE sends a message to another P processor and $(\Psi(0), (\Psi(1), \cdots, (\Psi(p-1))$is a permutation of $(0, 1, 2, \cdots, p-1)$.

Definition 4: Based on the above definitions and assumptions, if a computation step and the communication step in performing a parallel task on a DMS, are T_{cps} and T_{cms} respectively, then the time complexity of performing parallel tasks can be presented as: $O(T_{cps} + T_{cms}T_{o-oc}(P))$.

Furthermore, if the number of PEs in parallel processing system is less than the required sub-tasks, then the time complexity can be shown as:

$$O(\tfrac{T_{sq}(N)}{p} + T_{par-com}(N, P)) \qquad (1)$$

where N is the problem size, P is the number of PEs available, $T_{sq}(N)$ is the time complexity of the best sequential algorithm, and $T_{par-com}(N, P)$ is the overall communication overhead of a parallel computation.
From an algorithmic point of view, a DMS is characterized by the function $T_{o-oc}(P)$ which measures the communication capability of the interconnection network.

According to (Coppersmith and Winograd 1990), the fastest sequential algorithm for matrix multiplication

has the time complexity of $O(N^{\delta})$ where currently the best value for δ is 2.3755.
Based on these definitions, we try to find out the best time of running this sequential algorithm in parallel form on Hypercube and Super-Hypercube.

MATRIX MULTIPLICATION ON HYPERCUBE AND SUPER-HYPERCUBE

In multiplying two $N \times N$ matrices where the number of PEs is less than the number of sub-tasks , i.e. $1 \leq P \leq N^{\delta}$, we assume that l is an integer such that $l^{\delta} \leq P$ i.e. $l = \lfloor P^{\frac{1}{\delta}} \rfloor$ which has the matrices of sub-matrices $\frac{N}{l}$ (i.e., all the matrices $A = (A_{ij}), B = (B_{ij})$ and $C = (C_{ij})$ are partitioned to sub-matrices of size $\frac{N}{l} \times \frac{N}{l}$). Therefore, one can conclude that, in terms of computation time, if we multiply $\frac{N}{l} \times \frac{N}{l}$ matrix by $\frac{N}{l} \times \frac{N}{l}$ matrix sequentially on P PEs it will take $\frac{N^{\delta}}{P}$ units of time.

Calculation of the Communication Time for Matrix Multiplication on Hypercube

As we know the identification of each PE in h_{hp} dimensional Hypercube is based on their binary representation. The set of PEs which are distance d from one PE to another in Hypercube is showed by C_d and it includes $\binom{h_{hp}}{d}$PEs. Since Hypercube is a symmetrical architecture, so any algorithm which is written for any PE can be converted to an identical algorithm for PE $n \in \{1, 2, ...2^{h_{hp}-1}\}$by binary product of all PE i.d's referenced in any specific PE algorithm with n.

We assume that it takes $t_{trm}m_{len} + t_{st-t}$ time for a PE to send a message of length m to a neighbor, where t_{trm} represents the transfer rate of a message and t_{st-t} the time for start up and termination.

According to (F. Stout and Wagar. 1990), the fastest possible time for one PE in h_{hp} dimensional Hypercube to send a message to an arbitrary PE with distance d is:

$$T_{(comm)_{hp}} = \begin{cases} \frac{t_{trm}}{h_{hp}} m_{len} + 2\sqrt{\frac{(d+1)t_{st-t}t_{trm}}{h_{hp}}} m_{len}^{\frac{1}{2}} \\ +(d+1)t_{st-t} \\ \text{for } d \leq (h_{hp}-1) \\ \frac{t_trm}{h_{hp}} m_{len} + 2\sqrt{\frac{(h_{hp}-1)t_{st-t}t_{trm}}{h_{hp}}} m_{len}^{\frac{1}{2}} \\ +(h_{hp}-1)t_{st-t} \\ \text{for } d = h_{hp} \end{cases}$$

(2)

In this scenario, we assume that all PEs can communicate to one another simultaneously.
So, when multiplying two matrices of size $N \times N$on a h_{hp}-dimensional Hypercube by applying sub-matrices of size $\frac{N}{l} \times \frac{N}{l}$, each PE can broadcast the message of length $\frac{N^2}{P^{\frac{2}{\delta}}}$.

For calculating the communication time in the Hypercube, we consider the worse case scenario. This simply implies that if we intend to send a message of length $\frac{N^2}{P^{\frac{2}{\delta}}}$ from any PE to the farthest PE ($d = \log P_{hp}$), that would include the communication time for all the PEs within this range.

Therefore, the time takes to multiply two $N \times N$ matrices in forms of sub-matrices on a Hypercube is:

$$T_{(par-comp)_{hp}} = T_{(comp)_{hp}} + T_{(comm)_{hp}}$$
This results in:
$$T_{(par-comp)_{hp}} = \frac{N^{\alpha}}{P_{hp}} + \frac{t_{trm}}{\log p_{hp}}(\frac{N^2}{P_{hp}^{\frac{2}{\delta}}}) + 2\frac{\sqrt{(logP_{hp}-1)t_{st-t}t_{trm}}}{logP_{hp}}(\frac{N}{P_{hp}^{\frac{1}{\delta}}}) + (\log P_{hp} - 1)t_{st_t} \quad (3)$$

where P_{hp} denotes the number of PEs in Hypercube.

Calculation of the Communication Time for Matrix Multiplication on Super- Hypercube

Now we are in the position to expand the above methodology to cover the Super-Hypercube architecture. This means for the case of Super-Hypercube:

$$T_{(par-comp)_{shp}} = T_{(comp)_{shp}} + T_{(comm)_{shp}}$$

By including the Router (R) in the middle of Hypercube, we have provided a direct connection between any two PEs in Hypercube. Therefore, all PEs in Super- Hypercube are in equal distance to one another.

Moreover, the required time to multiply two $N \times N$ matrices in form of sub-matrices on a Super-Hypercube is:

$$T_{(par-comp)_{shp}} = \frac{N^{\delta}}{P_{shp}} + \frac{t_{trm}}{\log P_{shp}}(\frac{N^2}{P_{shp}^{\frac{2}{\delta}}}) + 2\frac{\sqrt{2t_{st-t}t_{trm}}}{\log P_{shp}}(\frac{N}{P_{shp}^{\frac{1}{\delta}}}) + 2t_{st-t}$$

(4)

where P_{shp} is the number of PEs in Super-Hypercube. In deriving equation (4) we assumed that $P_{shp} = P_{hp}$.

COMPARISON

Figure 4 shows the graphical presentation of communication cost with variable number of PEs and Figure 5 presents the graphical presentation of the communication cost with variable matrix size. In evaluating these results we have assumed that $t_{trm} = t_{st-t} = 1$.

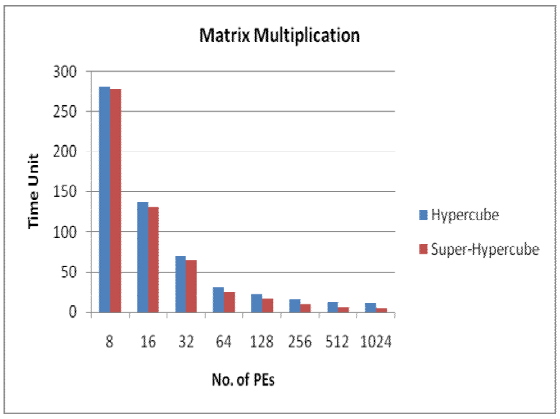


Figure 4. Graphical Presentation of communication cost for Matrix Multiplication with variable number of PEs.

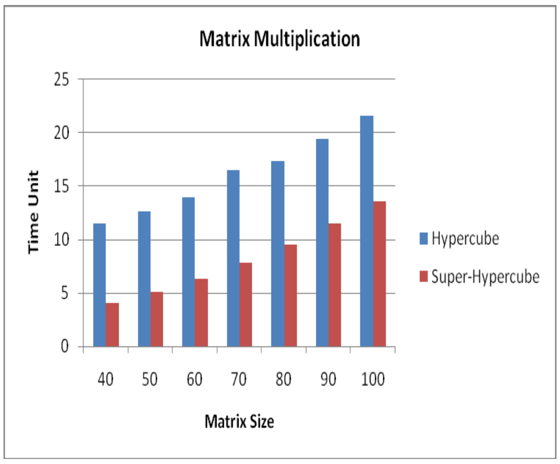


Figure 5. Graphical Presentation of communication cost for Martix Multiplication with variable matrix size.

By providing a direct path between any two indirect nodes through a Router (R), the communication time of a matrix product is sigificanytly shorter in Super-Hypercube compare with the Hypercube archticture. This is evident by analyzing Figure 4 and Figure 5 respectively. This accomplishment has played an important part towards improving the overall operation time and hence the performance of the message-passing architecture. This is clearly evident in Figure 4, when the number of PEs exceeds 256.

CONCLUSION

This paper has addressed the communication cost of a matrix product on message-passing architectures. The mathematical modeling for matrix multiplication on Hypercube and Super-Hypercube architectures was derived and numerical results for both architectures were presented. The existence of a Router (R) in a traditional Hypercube which results in having the Super-Hypercube has significantly improved the overall performance of the system. As a further work, we intend to propose an enhanced version of Super-Hypercube architecture. Then we will deliver the general formula for parallel time computation. This will be complementing the mathematical calculations and simulation carried out for this architecture. Furthermore, to validate this analysis, the most appropriate architecture will be chosen and compared with our findings to support our claims.

REFRENCES

Abachi, H and A.L, Walker. 1997. "Simulation Modeling of Fault-Tolerant Hypercube, Super-Hypercube and Torus Networks" *Proceeding of 12th International Conference on Computers and Their Applications (ISCA)*, Arizona, U.S.A, 50-53 (March).

Amiripour, M.; H. Abachi; and R. Lee. 2007. "Total System Cost and Average Routing Distance Analysis of Master-Slave Super-Super-Hypercube 4-Cube Message-Passing Architecture*" The International Journal of Computer and Information Science (IJCIS)*, Vol 10, No 2, 269-279 (June).

Coppersmith, D and S. Winograd. 1990. "Matrix Multiplication via Arithmetic Progressions." *J. Symbolic Computation*, Vol 9, 251-280.

Dongarra, J; J.F. Pineau; Y. Robert; Zh. Shi; and F. Vivien. 2007 " Revisiting Matrix Product on Mater-Worker Platform."IEEE Proceding on Parallel and Distributed Processing Symposium,(IPDPS 2007) ,1-8, (March)

F. Stout, Q. and B. Wagar. 1990. "Intensive Hypercube Communication: Prearranged Communication in Link-Bound Machines." *Journal of Parallel and Distributed Computing 10*, 167-181.

Grama, A.; A. Gupta; G. Karypis; and V. Kumar. 2003. "*Introduction to Parallel Computing*." Addison Wesley, U.S.A.

MARYAM AMIRIPOUR received her B.A. in Mathematics from Al-Zahra University in Iran in 1999. That was followed by a Post Graduate Diploma in Information and System Management form Queensland University in Australia in 2001.She is currently pursuing her PhD degree in Department of Electrical and Computer Systems Engineering at Monash University in Australia. Her area of research includes hardware design, modeling and simulation of advance parallel processing systems. The main parameters of her investigation include evaluation of performance, reliability, speed and cost analysis of massively parallel processing systems. She has a number of referred journal and conference papers in these areas. Her e-mail address is: maryam.amiripour@eng.monash.edu.au.

HAMID ABACHI received his Ph.D. degree in Computer Engineering from University College Cardiff in Wales, Britain, in 1981. He has twenty five years of teaching, research and administrative experiences in international universities around the world. He is currently an Associate Professor in the Department of Electrical and Computer Systems Engineering at Monash University in Australia. He has more than 95 referred international publications including Journal and conference papers. He has served as a member of international program committee to more than 72 international conferences around the world. On a number of occasions has acted as the conference chairman and on many occasions as the session chairman at international conferences. He has also participated in the plenary sessions at sessions at international conferences. He has been a co-recipient of the John Madsen Medal for his best Journal paper in the discipline of Electrical Engineering form the Institution of Engineers Australia (IEAust) in 2002, plus receiving a number of best paper awards in international conferences. In addition he has been invited as a keynote speaker at four international conferences. He is a Fellow of The Institution of Engineering and Technology (the IET, formerly IEE) in Britain, a Fellow of the Institution of Engineers in Australia (IEAust) and a Senior Member of IEEE in the USA. His research interests include design and simulation of parallel processing systems, modeling of advanced computer architectures, application of distributed multimedia computing in advanced Engineering Education. His e-mail address is: hamid.abachi@eng.monash.edu.au.

Application Of Novel Techniques In RIPEMD-160 Hash Function Aiming At High-Throughput

H. Michail,V. Thanasoulis, D. Schinianakis, G. Panagiotakopoulos and C. Goutis
Electrical and Computer Engineering Department
University of Patras
Gr-26500 Patra, GREECE
E-mail: michail@ece.upatras.gr, vthanasouli@upnet.gr, dimmugr@yahoo.gr, gpanagiotak@upnet.gr, goutis.ece.upatras.gr

KEYWORDS
Security, hash functions, RIPEMD-160, hardware implementation, high-throughput.

ABSTRACT

Hash functions, form a special family of cryptographic algorithms that address the requirements for security, confidentiality and validity for several applications in technology. Many applications like PKI, IPSec, DSA, MAC's need the requirements mentioned before. All the previous applications incorporate hash functions and address, as time passes, to more and more users-clients and thus the increase of their throughput is necessary. In this paper we propose an implementation that increases throughput and operating frequency significantly and at the same time keeps the area small enough for the hash function RIPEMD-160. The deployed technique involves the application of spatial and temporal pre-computation to the conventional operation block. The proposed implementation leads to an implementation that achieves 35% higher throughput.

INTRODUCTION

Nowadays many applications like the Public Key Infrastructure (PKI) (Entrust Technologies 1999), IPSec (National Institute of Standards and Technology 2005), and the 802.16 (Johnston and Walker 2004) standard for Local and Metropolitan Area Networks incorporate authenticating services. All these applications pre-suppose that an authenticating module, that includes a hash function, is nested in the implementation of the application. Moreover, digital signature algorithms like DSA (National Institute of Standards and Technology 1994) used for authenticating services like electronic mail, electronic funds transfer, electronic data interchange, software distribution, data storage etc are based on using critical cryptographic primitives like block ciphers and hash functions . Hashes are used also in identifying files on peer-to-peer filesharing networks, for example in an ed2k link. Furthermore, hashing cores are also essential for security in networks and mobile services, as in SSL. They are also the main modules that exist in the HMAC implementations that produce Message Authentication Codes (MAC's) (National Institute of Standards and Technology 1995).

Taking into consideration the rapid evolution of the communication standards that include message authenticity, integrity verification and non-repudiation of the sender's identity, it is obvious that hash functions are widely used in most popular cryptographic schemes. All the aforementioned applications ,which incorporate hash functions, are being used more and more as time goes by. So, it is necessary to increase their throughput, so as to enable the cryptographic system to satisfy immediately all requests from all users-clients. In some of these cryptographic schemes the throughput of the incorporated hash functions determines the throughput of the whole security scheme.
The latter mentioned facts were a strong motivation to propose a novel technique for increasing throughput of hash functions. In this work we propose an optimizes implementation for RIPEMD-160. The proposed implementation introduces a negligible area penalty, increasing the throughput and keeping the area small enough as required by most portable communication devices.

PROPOSED IMPLEMENTATION

In Fig. 1, the general architecture of RIPEMD-160 core with pipelined structure is illustrated. It has to be mentioned that no many research work has been conducted concerning the RIPEMD algorithm since the vast majority of both academia nad industry is focused on proposing optimizations for the SHA hash family which is the most widely adopted.
From (Dobbertin, et al 1996) it can be seen that RIPEMD-160 uses two parallel processes of five rounds, with sixteen operations for each round (5 x 16 operations for the process). This lead us to the logical assumption to use five pipeline stages for each process and a single operation block for each round among with the rest necessary parts. This way not only do we achieve to increase throughput drastically but also keep the hash core small enough.
However this what most researchers do and not much effort has been made in optimization of the operational block. In our paper we propose a methodology that intends to optimize the implementation both by applying pipeline stages but also by optimizing the internal operational block so as to achieve an even shorter critical path. This way it will be achieved to

Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

obtain an implementation with much higher throughput and with negligible and small area penalty which is the main objective of our work.

The critical path of the illustrated architecture is located between the pipeline stages. Thus, the optimization of the critical path is focused on the operation block. This way the increase of operating frequency can be achieved resulting to an implementation with a higher throughput. The throughput of a hash function implementation is given by the following equation:

$$Throughput = \frac{\#bits \cdot f_{operation}}{\#operations} \qquad (1)$$

where #bits is equal to the number of bits processed by the hash function, #operations corresponds to the required clock cycles between successive messages to generate each Message Digest and $f_{operation}$ indicates the maximum operating frequency of the circuit.

A message block, as provided by the padding unit, is at most 512 bits, therefore the two terms that can be manipulated in Eq.(1) is either #operations or the circuit's operating frequency, $f_{operation}$. Manipulation of the #operations is translated to the introduction of more than five pipeline stages. This is possible but it might result in area violation since extra circuitry must be inserted.

Thus, the targeted design approach should focus on increasing the operating frequency, $f_{operation}$, without introducing any significant area penalty.

Optimizing block's operating frequency

The applied technique consists of the following 2 sub-techniques:

- Spatial Pre-computation of additions contributing to the critical path

- Temporal Pre-computation of some values that are needed in following operations

Unfolding the expressions of a_t, b_t, c_t, d_t, e_t as they are described in [8], it is observed that b_{t-1}, c_{t-1} , d_{t-1} , e_{t-1} values are assigned directly to outputs c_t, d_t, e_t, a_t respectively. In Eq. (2) the expressions of a_t, b_t , c_t , d_t , e_t are defined.

$$\begin{aligned} e_t &= d_{t-1} \\ d_t &= ROL_{10}(c_{t-1}) \\ c_t &= b_{t-1} \qquad (2) \\ b_t &= e_{t-1} + ROL_s[f_t(b_{t-1}, c_{t-1}, d_{t-1}) + a_{t-1} + X_i + K_j] \\ a_t &= e_{t-1} \end{aligned}$$

where ROLx(y) represents cyclic shift (rotation) of word y to the left by x bits and ft(z, w, v) denotes the non-linear function which depends on the round being in process.

From Eq.(2), it is derived that the maximum delay is observed on the calculation of the b_t, value from a_{t-1} and value b_{t-1}, value. Obviously the critical path consists of three addition stages as it can be seen observing Fig. 2 and a multiplexer via which the values pass each time to/and feed the operation block.

A notice that one can make observing the Eq. (2) is that some outputs derive directly from some inputs values respectively. So it can be assumed that is possible during one operation to pre-calculate some intermediate values that will be used in the next operation so as to achieve concurrent calculations.

Therefore, while the main calculations are in progress, at the same time some values that are needed in the next operation can also be in progress of calculation. Furthermore, moving the pipeline stage to an appropriate intermediate point to store these intermediate calculated values, the critical path is divided resulting in a decrease of the maximum delay without paying any worth-mentioning area penalty. This way higher operating frequency is achieved and consequently higher throughput

This technique introduces the spatial pre-computation and it is used in order to reduce the critical path. From the Eq. (2) we can observe that the outputs c_t, d_t, e_t, a_t derive directly from the values b_{t-1}, c_{t-1}, d_{t-1}, e_{t-1}, respectively, and it is possible to pre-calculate some intermediate values.

Thus, Eq. (2) is transformed to generate the intermediate values a^*_{t-1}, b^*_{t-1}, c^*_{t-1}, d^*_{t-1}, e^*_{t-1} and g_{t-1} as described in Table 1.

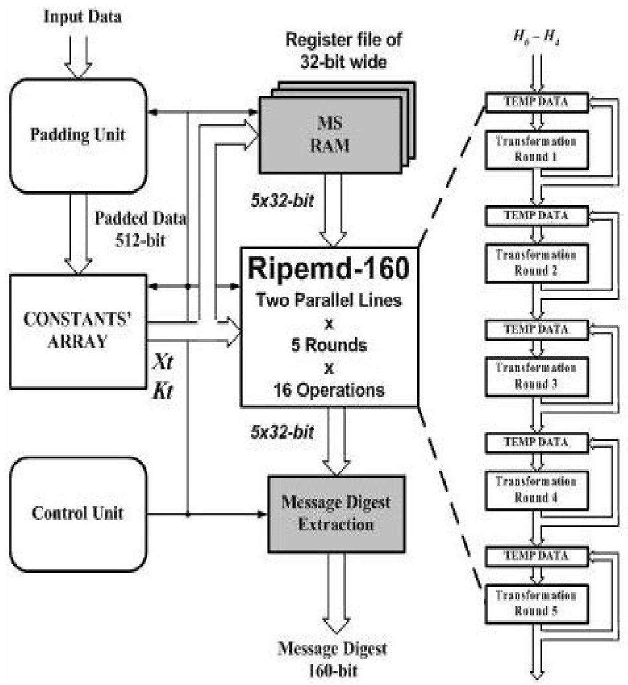
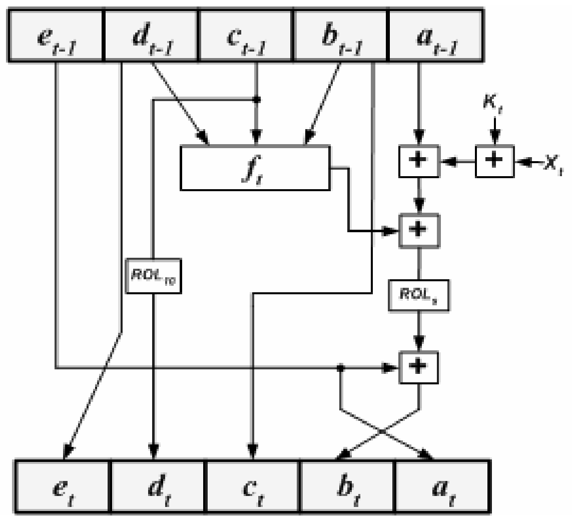


Figure 1: RIPEMD - 160 Architecture Core with Five Pipeline Stages Including a Single Operation Block



e_{t-1} d_{t-1} c_{t-1} b_{t-1} a_{t-1}
K_t
X_t
f_t
ROL_{10}
ROL_s
e_t d_t c_t b_t a_t

Figure 2: A Single RIPEMD - 160 Operation Block

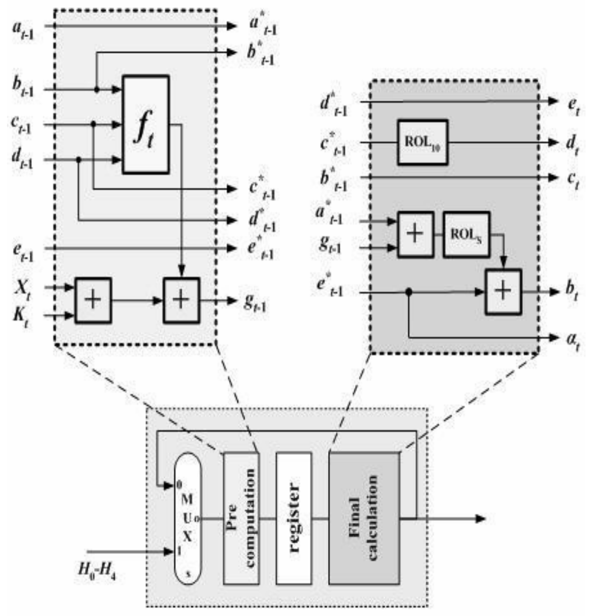
Table 1: Expressions for Spatial Technique

Pre-Computation	Final Calculation
$e^*_{t-1} = e_{t-1}$	$e_t = d^*_{t-1}$
$d^*_{t-1} = d_{t-1}$	$d_t = ROL_{10}(c^*_{t-1})$
$c^*_{t-1} = c_{t-1}$	$c_t = b^*_{t-1}$
$b^*_{t-1} = b_{t-1}$	$b_t = e^*_{t-1} + ROL_s(g_{t-1} + a^*_{t-1})$
$a^*_{t-1} = a_{t-1}$	$a_t = e^*_{t-1}$
$g_{t-1} = f_t(b_{t-1}, c_{t-1}, d_{t-1}) + X_t + K_t$	

In Fig.3 the pre-computation technique applied in RIPEMD-160 hash function is illustrated. Each operation block now consists of two units the “Pre-Computation” unit which is responsible for the pre-computation of the values that are needed in the next operation and the “Final-Calculation” unit which is responsible for the final computations of each operation.
Notice that in Fig.3 output b_t enters the multiplexer and feeds a no-load wire b_{t-1} which stores its value to the register as b^*_{t-1}. Also notice at the "Pre-Computation" unit that the inputs a_{t-1}, c_{t-1}, d_{t-1}, e_{t-1}, which is equal with the values a^*_{t-1}, c^*_{t-1}, d^*_{t-1}, e^*_{t-1} respectively, are fed through the multiplexer from the intermediate register outputs e^*_{t-1}, b^*_{t-1}, c^*_{t-1}, d^*_{t-1} respectively.
The introduced area penalty is small, only a single register for each “round”, that stores the intermediate value g_{t-1}.
Moreover, power dissipation is kept low and almost the same to that of the initial implementation as illustrated in Fig.2.In order to reduce the critical path by one addition level, we will continue with the application of the second technique, which introduces a temporal pre-computation of the values. From the “Final-Calculation” stage of Fig.3, one can observe that in every operation, from the current value of d_{t-1}, derives directly the value of e_t (at the next operation). Also, from the current value of e_t, derives directly the value of a_{t+1}. Consequently, the value of a, is the same as the value of was two operations earlier. So it is valid to write the following equation:

$$a_{t+1} = e_t = d_{t-1} \qquad (3)$$

Thus, we perform the temporal pre-computation of the sum $(X_{t+2} + K_{t+2}) + a_{t+1}$ two operations before it is used, by calculating the sum $(X_{t+2} + K_{t+2}) + d_{t-1}$ at the “Final-Calculation” unit, when the operation t is being executed. Then this sum at the “Pre-Computation” stage of the next operation (t+1) saved into the register h and represent the sum $(X_{t+2} + K_{t+2}) + e_t$.



a_{t-1} b_{t-1} c_{t-1} d_{t-1} e_{t-1} X_t K_t
f_t
a^*_{t-1} b^*_{t-1} c^*_{t-1} d^*_{t-1} e^*_{t-1} g_{t-1}
ROL_{10}
ROL_s
e_t d_t c_t b_t a_t
MUXs
Pre computation
register
Final calculation
H_0-H_4

Figure 3: The Modified Operation Block of RIPEMD-160 Hashing Algorithm

At the “Final-Calculation” unit of the same operation, the value of W derives directly from the value of h. The computed sum now of the value W represents the sum $(X_{t+2} + K_{t+2}) + a_{t+1}$. Finally at the “Pre-Computation” unit on the next operation (which is the operation t+2) the sum Z= W + f_t is calculated.
The computed sum now represents the value $(X_{t+2} + K_{t+2}) + f_t + a_{t+1}$. This sum is part of the computations needed for the calculation of b_{t+2} value. What remains for the computation of the value b_{t+2} is the rotation (Rol_s) of the value Z and then its addition with the value e^*_{t+1}, as is performed in the “Final-Calculation” in Fig.4.
Observing Fig.4 it can be realized that the critical path is not located any more in the computation of the b_t value but in the computation of the value of Z. This means that the critical path in Fig. 4 has been reduced from three addition stages, a Non Linear Function f_t and

a multiplexer in Fig.3 to two addition stages, a Non Linear Function f_t and a multiplexer.

Hence, the critical path is shortened by one adder level, which contributes approximately 30% to the overall maximum delay. Moreover, it has to be noticed that an initialization of the values of W and h is needed as illustrated in Fig.4. At the first operation of every round the current values of X_t and K_t contribute for the computation of the value b_{t+2}. Thus, before the first operation begins, the value of W must be equal to the sum $(X_1 + K_1) + a_0$, which will be used at the "Final-Calculation" of the first operation for the calculation of the value b_t . Also the value of h must be equal to the sum $(X_2 + K_2) + e_0$, which will be used for the calculation of the value b_{t+1} at the "Final-Calculation stage of the second operation. Therefore, another one modification that introduces two adders is needed. However, this change does not have any effect on the critical path.

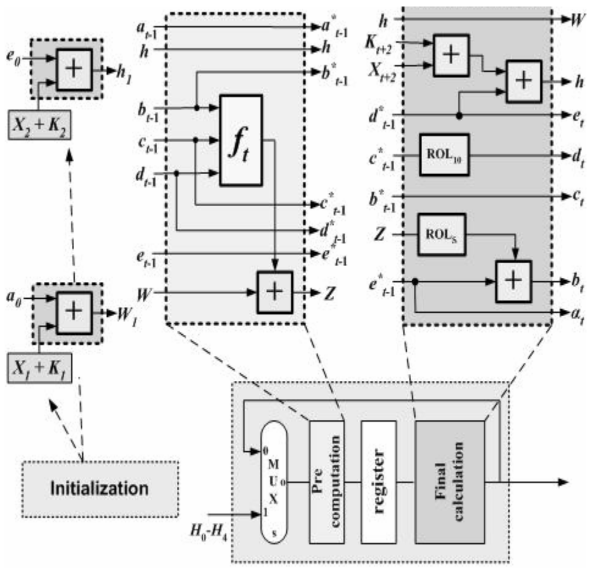


Figure 4: The Proposed Operation Block of RIPEMD-160 algorithm

The introduced area penalty is only two 32-bit registers, which are used for storing the intermediate results of the values W and h that are required. This area penalty sure enough is worth paying for an increase of throughput at about 36%.

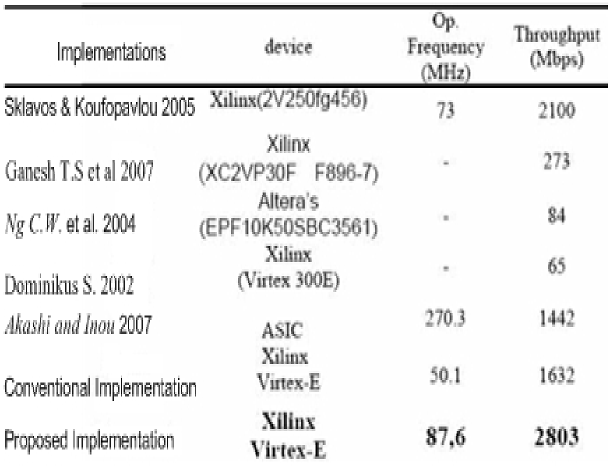
EXPERIMENTAL RESULTS

The proposed hashing cores that were presented as examples were captured in VHDL and were fully simulated and verified using the Model Technology's ModelSim Simulator. The designs were fully verified using a large set of test vectors, apart from the test example proposed by the standards. The synthesis tool used to port VHDL to the targeted technologies was Synplicity's Synplify Pro Synthesis Tool. Simulation of the designs was also performed after synthesis, exploiting the back annotated information that was extracted from the synthesis tool. Further evaluation of the designs was performed using the prototype board for the Xilinx Virtex device family.

Probing of the FPGA's pins was done using a logic analyzer. No scaling frequency technique was followed, selecting one master clock for the system, which was driven in the FPGA from an onboard oscillator. The behavior of the implementation was verified exploiting the large capacity of the FPGA device. The achieved operating frequency is equal to 87,6 MHz.

Furthermore, as far as it concerns the introduced area overhead for the RIPEMD-160 hash core, the proposed implementation presents an increase of approximately 8%. From the experimental results, it was proved that RIPEMD-160 proposed implementation was about 36% faster than the conventional implementation. From the above results, it derives that the proposed implementation is a worth-making optimization for the hashing core since the required area for the whole security scheme is much more than that needed for the RIPEMD-160 hashing core.

Table 2: Performance Characteristics of RIPE-MD 160 Hash Function Implementations and Comparisons



Implementations	device	Op. Frequency (MHz)	Throughput (Mbps)
Sklavos & Koufopavlou 2005	Xilinx(2V250fg456)	73	2100
Ganesh T.S et al 2007	Xilinx (XC2VP30F F896-7)	-	273
Ng C.W. et al. 2004	Altera's (EPF10K50SBC3561)	-	84
Dominikus S. 2002	Xilinx (Virtex 300E)	-	65
Akashi and Inou 2007	ASIC	270.3	1442
Conventional Implementation	Xilinx Virtex-E	50.1	1632
Proposed Implementation	**Xilinx Virtex-E**	**87,6**	**2803**

It has to be added that the above comparisons concern hardware implementations mainly in FPGA boards. However due to the limited number of published work concerning RIPEMD all implementation results have been included regardless of the utilized FPGA family.

For this reason the evaluation board in each case is mentioned so as to be easy to adapt the results and make a fair comparison.

Beyond that however, the comparison results just confirm the efficiency of the proposed technique and this is accomplished with the comparison to the conventional implementation that has also been evaluated from our research time in the same FPGA board. From this comparison it can be inferred that the proposed implementation achieves its objective of higher throughput of about 35% with only 8% area penalty for evaluation in the same FPGA board (used in our research laboratory).

CONCLUSIONS

A novel hardware implementation of RIPEMD-160 hash function was presented in this paper. Two techniques were evaluated so as to increase throughput and thus make it suitable for the corresponding server of data intensive applications. The proposed implementation has a throughput of about 2.8 Gbps, about 35% higher from the next better performing implementation. The experimental results showed that a small area penalty was introduced for a remarkable increase of throughput.
Therefore, the proposed implementation increases the throughput and frequency significantly and keeps at the same time the area small. This makes the proposed implementation suitable for server side cryptographic schemes as well as and for every new wireless and mobile communication application that urges for high-performance and small-sized solutions.
This methodology can be applied to all other hash functions such as MD-5, SHA-1, SHA-256, SHA-384, SHA-512 in order to increase their throughput.

ACKNOWLEDGEMENT

This work was supported by the project PENED 2003 No 03ED507, which is funded in 75% by the European Union- European Social fund and in 25% by the Greek state-Greek Secretariat for Research and Technology.

REFERENCES

Akashi S. and T, Inoue. 2007. “ASIC-hardware-focused comparison for hash functions MD5, RIPEMD-160, and SHS” Intergration, the Vlsi Journal 40 , 3-10.

Dobbertin H., Bosselaers A. and Preneel B. 1996. “ RIPEMD-160: A Strengthened Version of RIPEMD”.

Dominikus S. 2002. “A hardware implementation of MD4-family hash algorithms,” Proc. 9th Int. Conf. on Electronics, Circuits and Systems, vol. 3, pp. 1143-1146.

Entrust Technologies. 1999. “RFC 2510 - Internet X.509 PKI - Certificate Management Protocols”,

Ganesh T.S, Frederick M.T, Sudarshan T.S.B. and.Somani A.K. 2007. "Hashchip: A shared-resource multi-hash function processor architecture on FPGA" Intergration, the Vlsi Journal 40 11-19.

Johnston D, and Walker J. 2004. “Overview of IEEE802.16 Security” , IEEE Security and Privacy.

National Instititute of Standards and Technlogy. 1994. “FIPS 186, (DSS), Digital Signature Standard”

National Instititute of Standards and Technlogy. 1995. “FIPS 198, The Keyed-Hash Message Authentication Code (HMAC)”

National Instititute of Standards and Technlogy. 2005. SP800-77 , “Guide to IPSec VPN’s”.

Ng C.W., Ng T.S. and Yip K.W. , 2004. “A unified architecture of MD5 and RIPEMD-160 Hash algorithms”, IEEE International Symposium on Circuits and Systems.

Sklavos N. and Koufopavlou O. .2005. “On the hardware implementation of RIPEMD processor: Networking high speed hashing, up to 2 Gbps”, Computers and Electrical Engineering Journal 31 361–379.

AUTHOR BIOGRAPHIES

HARRIS MICHAIL (S’04) received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of computer security, hardware design and reconfigurable architectures.

VASSILIS THANASOULIS is an under-graduate student in the Department of Electrical .Eng, University of Patras, Greece. He is currently working on his thesis that lies in the domain of security.

DIMITRIS SCHINIANAKIS received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of of computer security, hardware design

GEORGE PANAGIOTAKOPOULOS received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of embedded computers.

COSTAS GOUTIS (S’70-M’78) received B.Sc in Physics, Diploma in Electronic Engineering, M.Sc from the University of Heriott-Watt and Ph.D from the University of Southampton. He is currently a Professor with the ECE Department, University of Patras.

MASSIVELY PARALLEL SIMULATIONS OF ASTROPHYSICAL PLASMAS: STATUS AND PERSPECTIVES OF THE COAST PROJECT

B. Thooris, E. Audit, A. S. Brun, Y. Fidaali, F. Masset, D. Pomarède, R. Teyssier
Institut de Recherche sur les Lois Fondamentales de l'Univers
DSM/IRFU CEA/Saclay 91191 Gif-sur-Yvette France
email: Bruno.Thooris@cea.fr
http://irfu.cea.fr/Projets/COAST

KEYWORDS
Large scale computing, parallel computing, astrophysics, plasmas simulation, visualization.

ABSTRACT

The COAST (for Computational Astrophysics) project is a program of massively parallel numerical simulations in astrophysics involving astrophysicists and software engineers from CEA/IRFU Saclay. The scientific objective is the understanding of the formation of structures in the Universe, including the study of large-scale cosmological structures and galaxy formation, turbulence in interstellar medium, stellar magnetohydrodynamics and protoplanetary systems. The simulations of astrophysical plasmas are performed on massively parallel mainframes (MareNostrum Barcelona, CCRT CEA France), using 3-D magnetohydrodynamics and N-body parallelized codes developed locally. We present in this paper an overview of the software codes and tools developed and some results of such simulations. We also describe the Saclay SD*vision* graphical interface, implemented in the framework of IDL Object graphics, our 3-D visualization tool for analysis of the computation results.

1. Introduction

The COAST project [1,2] is dedicated to high performance computing in astrophysics. The goal is the understanding of the formation of structures in the Universe, by developing advanced techniques in parallel computing and in applied mathematics to model galaxy formation and predict their observational signatures, as a function of physical parameters. Astrophysicists and software engineers collaborate to rationalize and optimize the development of simulation programs by creating a core of common specific modules and using common software tools for data handling, post-treatment, visualization, numerical methods, parallelization and optimization.

2. Overview of the simulation programs

Four major numerical simulation programs are used to cover different physics scales:

- The RAMSES code

RAMSES [3,4,5,6] is a hybrid, N-body and hydrodynamical 3-D code which solves the interplay of the dark matter component and the baryon gas for studying the structure and the distribution of galaxy clusters starting for the initial conditions of the Big Bang. The code is based on the Adaptive Mesh Refinement (AMR) technique, written in FORTRAN90 and parallelized with the MPI library [7]. Current developments focus on solving the full MHD set of equations.

- The HERACLES code

HERACLES [8,9,10,11,12] is a 3-D code which solves the equations of radiative transfer coupled to hydrodynamics. It studies thermal condensation in molecular clouds in the Interstellar Medium, radiative shocks, molecular jets of young stars and proto-planetary disks. It is written in FORTRAN90, parallelized with MPI and implemented in cartesian, cylindrical and spherical coordinates with regular mesh grids.

- The ASH [13,14] code

ASH (for Anelastic Spherical Harmonic) performs 3-D magnetohydrodynamics simulations in spherical geometry for the study of the turbulence and magnetic dynamo process in solar and stellar interiors. ASH, unlike the others codes presented which are completely developed in CEA/Saclay, is jointly developed at Saclay and at the University of Boulder.

- The JUPITER [15,16,17] code

JUPITER is a mutidimensional astrophysical hydrocode. It is based on a Godunov method, written in C and parallelized with MPI. The mesh geometry can either be

Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

cartesian, cylindrical or spherical. It allows mesh refinement and includes special features adapted to the description of planets embedded in disks.

3. Computing facilities

The COAST team uses local resources for development and post-treatment: the 256 cores 2.6 GHz opteron cluster DAPHPC, with an infiniband interface and four visualization stations with 16 to 32 Gb RAM, ~1Tb disk, 4 processors, 1Gb memory graphics cards, 30 inches screens.
Massive simulations are performed at CCRT (CEA National Supercomputing Center) on Platine, ranking 26th in the TOP500 world supercomputer list (November 2007): 7456 Itanium cores, total 23 Tb memory, 47.7 Teraflops (4 Mhrs computation in 2007).
Other resources for massive simulations (2 Mhrs for 2007) can be accessed on MareNostrum at the Barcelona Supercomputing Center, ranking 13th in the TOP500 world supercomputer list (November 2007): 10240 IBM PowerPC 2.3 GHz cores with 94.2 Teraflops, 20Tb of main memory.
Furthermore, the project will have access in 2008 to the IDRIS (French CNRS Supercomputing Center) new Blue Gene/P system with 40000 cores.

4. The recent COAST computational milestones

COAST members are participating in French or European founded collaborations (Horizon, Magnet, Sinerghy or STARS[2]).
In the context of the Horizon collaboration [18], COAST members succeeded in the HORIZON Grand Challenge Simulation at CEA/CCRT on Platine in September 2007, which is the largest ever N-body cosmological simulation performed. For the first time, have been performed a simulation of half the observable universe, with enough resolution to describe a Milky Way-like galaxy with more than 100 dark matter particles. The RAMSES code has been run on 6144 cores, 18 Tb RAM used for 2 months to simulate 4096^3□70 billions particles. This is an improvement of about an order of magnitude with respect to previous experiments, as illustrated in Fig.1. This simulation has been chosen to simulate future weak-lensing surveys like DUNE or LSST [19,20].
Another challenge in computing in astrophysics in 2007 was the HORIZON "galaxy formation" simulation at MareNostrum. The characteristics of the run are the following: 1024^3 dark matter particles, 4 billions AMR cells, box size 50 Mpc/h, resolution in space 2 kpc. 2048 processors were needed for computing, 64 processors dedicated to I/O, 3 weeks of computations so far, down to z=1.9, 20 Tb of data generated and stored. The run performed simulations from large scale filaments to galactic discs.

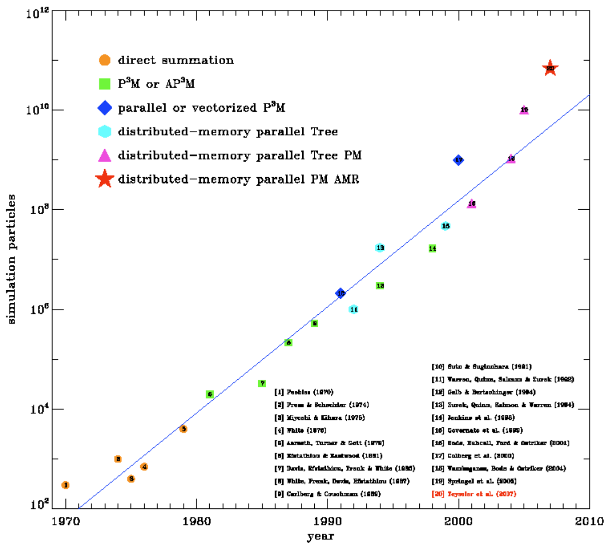


Fig.1 Size of the N-body simulations versus time (courtesy V. Springel [21]).

Two examples of MareNostrum data visualization are shown in Fig.2 and Fig.3, displaying the density distribution of the baryon gas at two different scales.

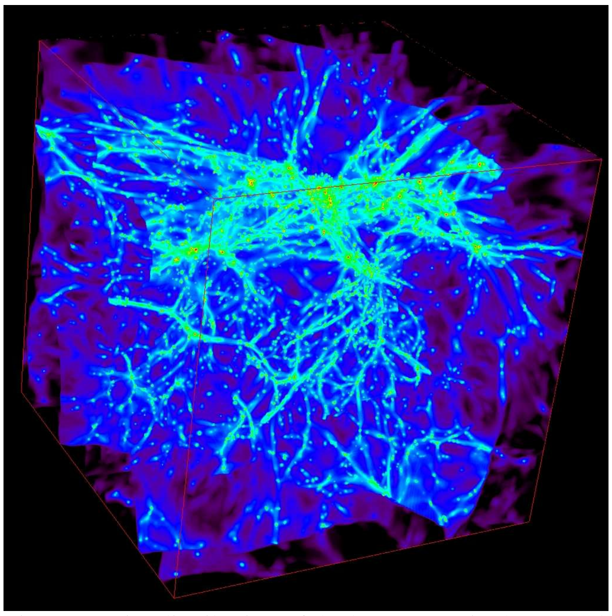


Fig.2 Density distribution of the baryon gas in the central region of simulation domain. Visualization made with SD*vision* software.

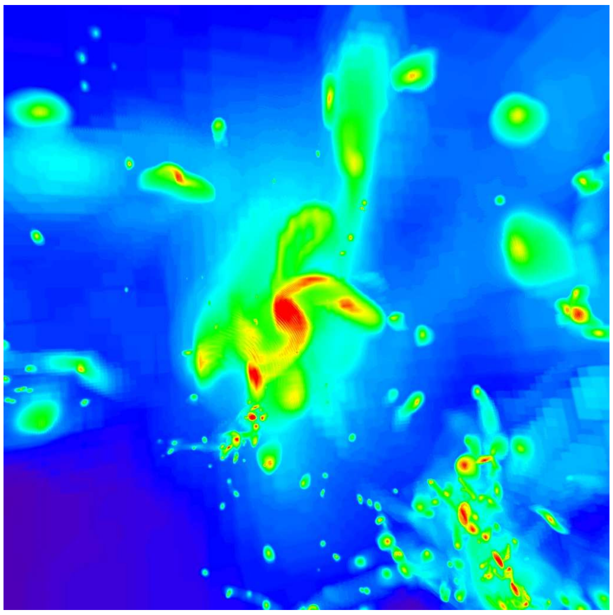


Fig.3 Zoom of galactic spirals, density distribution of the baryon gas. Visualization made with SD*vision* software

5. Data Handling

A unique format, HDF5 [22] (Hierarchical Data Format), has been chosen for the data produced by all our simulations codes. HDF5 is developed and maintained by NCSA (National Center for Supercomputing Applications). This library emphasizes storage and I/O efficiency in particular for data intensive computing environments. For instance, the HDF5 format can accommodate data in a variety of ways, such as compressed or chunked and the library is tuned and adapted to read and write data efficiently on parallel computing systems. NCSA maintains a suite of free, open source software, including the HDF5 I/O library and several utilities. The visualization tools developed is using IDL, which integrate a module able to read easily the HDF5 files.

6. Visualization

The visualization plays a very important role in the development of simulations codes. Fundamental aspects including domain decomposition, initial conditions, message passing and parallelization, treatment of boundary limits, can be controlled and evaluated qualitatively through visualization. Once in production phase, visualization is also used for the validation, the analysis and the interpretation of the results. A complete graphical interface named SD*vision* [23,24,25] has been developed in order to participate in the development of the simulation codes and visualize the large astrophysical simulation datasets produced in the context of the COAST program.

The SD*vision* graphical interface

The interface is implemented as a graphical widget providing interactive and immersive 3-dimensional navigation capabilities. The baseline technology is the object-oriented programming offered by IDL's Objects Graphics [26]. It benefits from hardware acceleration through its interface to the OpenGL libraries. An example of the widget displayed in its running state is shown in Fig. 4.

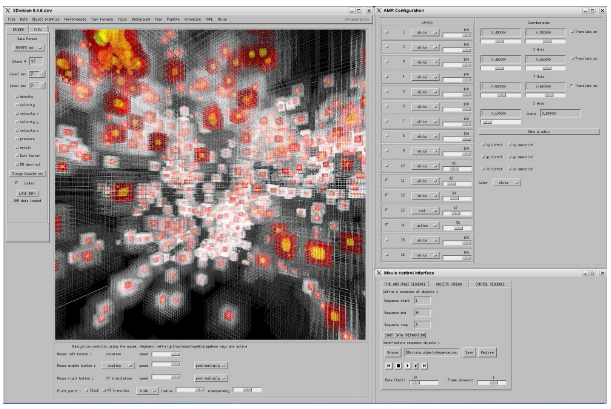


Fig.4 The SD*vision* widget used to visualize the levels 11,12,13 AMR cells of a galaxy cluster simulation mesh. □

SD*vision* has been developed to visualize the huge amount of data produced by the codes RAMSES, HERACLES, JUPITER and ASH. It allows the display of complex scenes with scalar fields (volume projection, 3D isosurface, slices), vector fields (streamlines) and particle clouds.

Two examples of such visual representations are shown in Fig.5 and Fig.6.

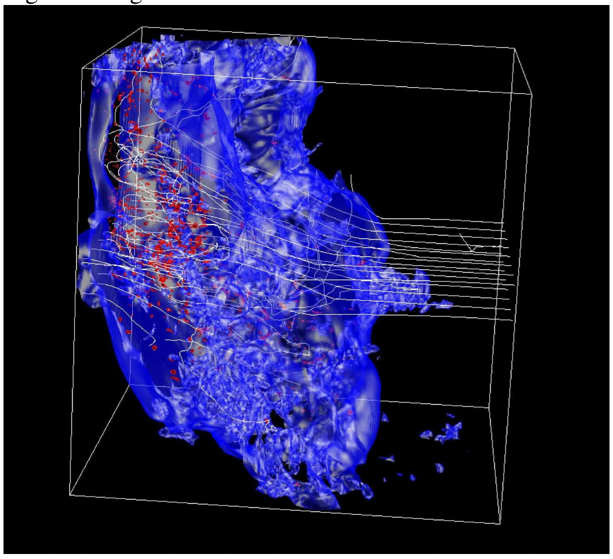


Fig.5 Visualization of the iso-density surfaces obtained in a high-resolution 1200^3 HERACLES 256-processors simulation of turbulences in the interstellar medium.

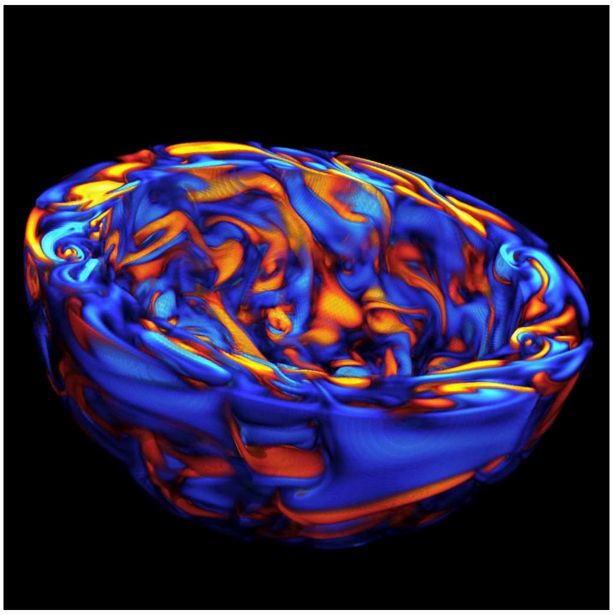


Fig.6 SD*vision* display of a 500x500x500 ASH simulation. This view of the azimuthal component of the magnetic field in the solar convection zone (positive in red, negative in blue) are obtained by the volume rendering technique with clipping.

7. Perspectives and conclusion

To be able to follow the improvements of the computers techniques, the COAST team dedicates a part of his human resources in training on software tools such as optimization techniques on recent mainframes (IDRIS Blue Gene) and R&D on new promising technologies for computing (GPU, Cells,…). Future challenges include the improvement by an order of magnitude in the N-body simulation of cosmological structure formation to improve the resolution. Using the next generation of high-performance centers, the objective is to simulate 8192^3 □550 billions particles

Computational astrophysics has a bright future, lying on the ever increasing performances of massively parallel mainframes. To achieve his ambitions, the project relies on a synergy between astrophysicists and software developers, local computing resources and access to supercomputers. Other major projects and initiatives are currently following the same approach: e.g. the FLASH Center at the University of Chicago [27], the ASTROSIM European Network for Computational Astrophysics [28], or the VIRGO consortium for Cosmological Supercomputer Simulations [29].

8. References

[1] http://irfu.cea.fr/Projets/COAST

[2] D. Pomarède, B. Thooris, E. Audit, R. Teyssier. Numerical Simulations of Astrophysical Plasmas. *Proceedings of the 6th IASTED International Conference on Modelling, Simulations, and Optimization (MSO2006)*, Gaborone, Botswana, September 11-13, 2006, ed. H. Nyongesa, 507-058, Acta Press, ISBN:0-88986-618-X

[3] R.Teyssier, Cosmological hydrodynamics with adaptive mesh refinement - A new high resolution code called RAMSES, Astronomy and Astrophysics, 385 (2002) 337-364

[4] S. Fromang, P. Hennebelle, R. Teyssier. A high order Godunov scheme with constrained transport and adaptive mesh refinement for astrophysical magnetohydrodynamics, A&A 457 (2006) 371

[5] R. Teyssier, S. Fromang, and E. Dormy. Kinematic dynamos using constrained transport with high order Godunov schemes and adaptive mesh refinement , J. Comp. Phys. 218 (2006) 44

[6] R. Osmont, D. Pomarède, V. Gautard, R. Teyssier, B. Thooris. Monitoring and Control of the RAMSES Simulation Program, *Proceedings of the CCP2007 Conference on Computational Physics*, Brussels, Belgium, September 5-8, 2007

[7] The Message Passing Interface (MPI) Standard, http://www-unix.mcs.anl.gov/mpi/

[8] M.Gonzalez, E.Audit, P.Huynh. HERACLES : a Three Dimensional Radiation Hydrodynamics Code. *Astronomy and Astrophysics,* 464 2 (2007) 429-435.

[9] E. Audit and P. Hennebelle, Thermal Condensation in a Turbulent Atomic Hydrogen Flow, *Astronomy and Astrophysics*, 433, 2005, 1-13.

[10]M. Gonzalez and P. Velarde. Radiative Shocks and Jets Simulated with the ARWEN and HERACLES codes. *Proceedings of the IGPP/DAPNIA International Conference on Numerical Modeling of Space Plasma Flows, ASTRONUM2007, Paris, France, June 11-15, 2007, to appear in the Astronomical Society of the Pacific Conference Series.*

[11]E. Audit. Fragmentation in the Interstellar Medium. *Proceedings of the IGPP/DAPNIA International Conference on Numerical Modeling of Space Plasma Flows, ASTRONUM2007, Paris, France, June 11-15, 2007, to appear in the Astronomical Society of the Pacific Conference Series.*

[12] E. Audit, V. Gautard, D. Pomarède, B. Thooris. Enabling Tools and Techniques for the Optimization of the HERACLES Simulation Program, *Proceedings of the 6th EUROSIM Congress*, Ljubljana, Slovenia, September 9-13, 2007

[13] J. Ballot, A.S. Brun, and S. Turck-Chieze. Simulations of turbulent convection in rotating young solarlike stars: differential rotation and meridional circulation. ApJ 669 (2007) 1190

[14] L. Jouve and A.S. Brun. On the role of meridional flows in flux transport dynamo models. A&A 474 (2007) 239

[15] F. Masset, A. Morbidelli, and A. Crida. Disk surface density transitions as protoplanet traps. ApJ 642 (2006) 478

[16] A. Crida, A. Morbidelli, and F. Masset. Simulating planet migration in globally evolving disks. A&A 461 (2007) 1173

[17] http://www.maths.qmul.ac.uk/~masset/index.html

[18] http://www.projet-horizon.fr/

[19] http://www.dune-mission.net/

[20] http://www.lsst.org/lsst_home.shtml

[21] V. Springel, et al. Simulating the joint evolution of quasars, galaxies and their large-scale distribution. Astro-ph/0504097 (2005).

[22] http://www.hdfgroup.org

[23] D. Pomarède, E. Audit, R. Teyssier, B. Thooris. Visualization of large astrophysical simulations datasets. *Proceedings of the Conference on Computational Physics 2006, CCP2006, Gyeongju, Republic of Korea, aug.29-sept.1 2006, ed. J.S. Kim, Computer Physics Communication*, 177 (2007) 263, doi:10.1016/j.cpc.2007.02.065

[24] D. Pomarède, Y. Fidaali, E. Audit, A.S. Brun, F. Masset, R. Teyssier. Interactive visualization of astrophysical plasma simulations with SDvision, *Proceedings of the IGPP/DAPNIA International Conference on Numerical Modeling of Space Plasma Flows, ASTRONUM2007, Paris, France, June 11-15, 2007, to appear in the Astronomical Society of the Pacific Conference Series.*

[25] D. Pomarède, E. Audit, A.S. Brun, V. Gautard, F. Masset, R. Teyssier, B. Thooris. Visualization of astrophysical simulations using IDL Object Graphics, *Proceedings of the Computer Graphics Imaging and Visualization 2007 Conference*, Bangkok, Thailand, August 14-17, 2007, IEEE Computer Society ISBN 0-7695-2928-3 p471-480

[26] http://www.ittvis.com/idl/

[27] http://flash.uchicago.edu/

[28] http://www.astrosim.net/

[29] http://www.virgo.dur.ac.uk/

HPCS 2008 AUTHOR INDEX