THREE LAYERS NETWORK INFLUENCE ON CLOUD DATA CENTER PERFORMANCES

Marco Gribaudo
DEIB
Politecnico di Milano
via Ponzio 51
20133, Milano, Italy
marco.gribaudo@polimi.it

Mauro Iacono
DSP
Seconda Università degli Studi di Napoli
viale Ellittico 31
81100 Caserta, Italy
mauro.iacono@unina2.it

Daniele Manini
DI
Università degli Studi di Torino
corso Svizzera 185
10129, Torino, Italy
manini@di.unito.it

KEYWORDS

cloud networking; data center performances; Markovian agents; performance modeling; virtualization

ABSTRACT

The effects of networks on the performances of cloud architectures are a very significant issue in designing a data center. The efficiency of data transfers and the overall traffic management are a critical factor that constitutes a potential performance bottleneck, potentially limiting the number of computing nodes that can be installed more than their cost issues. In this paper we present a modeling approach, based on Markovian agents, that allows a performance analysis of network effects in high scale cloud architectures.

I. INTRODUCTION

Virtualization is a key technology in the field of cloud computing. The use of virtual machines (VM) allows to exploit the enormous amount of computing power available in modern data centers, by decoupling the computing needs of the applications from physical processors and memory; moreover, VM are an efficient mean to neutrally save the state of a complex computation, to spawn different instances of the same computing environment or to implement safety and security related strategies and lower the overall risk in sensitive applications.

The drawback of using VM is their startup phase. When not in use, a VM is normally stored in the cloud storage system, with an (uncompressed) footprint that can be around several hundreds of megabytes. Additionally, a VM may use a persistent virtualized storage unit, that is logically mounted during the startup phase. Starting a VM, by using a standard predefined snapshot, or restarting a VM, previously stored as a snapshot at the end of the previous running period, is thus a time consuming task, due to data transfers from the storage subsystem of the cloud and the memory of the physical server chosen to run it.

As the schedule of the cloud depends on the workload, a snapshot (and, in case, its persistent storage) is not necessarily stored in the same node that can run it when needed (e.g. OpenStack): a new VM instance from a standard image has to be retrieved from the image repository; an existing stored snapshot may need to be moved on the node that offers enough physical resources and time slots; if the architecture is made of different nodes for computing and storage, the snapshot obviously needs to be properly sent to the computing node; if the storage subsystem uses a distributed file system (e.g. CEPH), the snapshot retrieval involves even more complex mechanisms.

As a consequence, VM startup relies on the efficiency of the network layer, that is the part of cloud architectures that grows at the lowest pace. In this paper we present a modeling technique to evaluate the impact of the network layer and its organization on the VM startup time in high scale cloud architectures based on a three-tier network and a standard, replication based distributed storage model.

The paper is organized as follows: the next Section presents related works; Section III introduces the reference scenario for this work; Section IV describes the modeling approach; Section V shows the application to a case study; conclusions follow in the last Section.

II. RELATED WORKS

The use of VM is a classic technique, known since the mainframe era, to optimize the use of a large amount of resources by smartly sharing them between different, independent and isolated complete software stacks, by running different operating systems on virtualized hardware. A good, performance evaluation oriented introduction is provided by [1], that also offers a good historical perspective. VM offer a great flexibility in the management of cloud resources, that may give great benefits if a proper performance analysis driven tuning is implemented [2], as many performance influencing factors arise from the complexity of the architecture and must be taken into account [3] [4] [5] [6] [7].

The most critical performance factor in a modern data center is the efficiency of the network (at the point that data dependent structures may be needed [8]): to get an idea of the needs of a large data center, [9] reports about how this problem has been studied and what solutions have been implemented in

Proceedings 30th European Conference on Modelling and Simulation ©ECMS Thorsten Claus, Frank Herrmann, Michael Manitz, Oliver Rose (Editors)
ISBN: 978-0-9932440-2-5 / ISBN: 978-0-9932440-3-2 (CD)

Google facilities. Several well spread architectures emerged, such as the three layer, the Clos, the fat tree [10] and the DCell ones [11] [12], but other solutions have been proposed as well (e.g. VL2 [13] and CONGA [14]). The problem is relevant also in distributed data centers [15] [16], but the scope of this paper is focused on the internal infrastructure of a single, high scale datacenter.

Performance evaluation studies on the main cloud network architectures have been performed by means of simulation: two very good examples are given by [11] and [12], that give a complete and comparative panorama. A simulative approach is potentially capable of allowing the analysis of very large scale clouds, at the cost of a long computational effort: simulation time is as much longer as much the system behaviors are variable and its scale is large. In this paper an analytical approach is preferred. An important issue is also the evaluation of energy consumption in cloud networks: this is out of the scope of this paper, but interested readers can find an interesting introduction and recent results in [17] and [18].

The authors already applied analytical and simulative methods to performance evaluation of cloud systems, both in small [19] [20] [21] [22] [23] and large scale [24] [25] [26] [27] [28]. Although analytical methods are known to be affected by the state space explosion problem, some approaches (e.g. product forms [29] and Markovian agents [30]) proved to be effective tools to overcome this limit. In this paper Markovian agents are exploited (as in [26], [27] and [28]) for their special suitability in modeling systems with very large number of states with increasing precision.

In this paper OpenStack cloud architecture is used as a reference. The management of VM images is documented at [31]; some technical information about typical VM images for OpenStack can be found at [32]. The network solicitation due to a VM is described at [33], while the integration of the operating system of a VM is described at [34] (using Ubuntu Linux distribution as an example). The main advantage of our approach woth respect to the rest of the literature is the capability, thanks to the adoption of Markovian agents, for seamlessly scaling up the models (and the dimensions and complexity of datacenters) to thousands of components, while keeping an analytical approach and increasing the precision of the approximation.

III. SCENARIO

In this work we focus on a datacenter of medium or large scale. Figure 1 shows a simplified architecture of the considered scenario. In a datacenter, the IT equipment is enclosed into fixed form factor cases called *rack units*. Units include *computing servers*, *storage servers*, *network equipments* and *power supply units* (PSUs). In this work we will not focus on PSUs; network equipments include switches, routers, firewalls and load balancers: in this work we will only focus on switches. Units are organized in columns, that we will simply address as *racks*. Racks are further organized in corridors, to improve the air circulation and the cooling of units. In particular, corridors are organized into *cold aisles* and *hot aisles*. The former ones present the front panels of the equipments, and allow technicians to access the controls of the units. The latter ones instead hold the backs of the units, and they are where cables interconnecting the units are placed. Cool air, produced by fans or air conditioners, enters the room from the cold aisles, flows through the units, cools them down, and exits the room from the hot aisles. Computing servers are usually special multiprocessor, multicore, and multithreaded power and network redundant x86 PCs. They are usually equipped with a relatively large amount of memory (currently in the range of 64-128 GB), and can run around 40-80 threads in parallel. They are however equipped with a limited disk space, and they relay to external storage to hold most of the persistent data. Storage units include both RAIDs (Rapid Array of Independent Disks) and JBOD (Just a Bunch Of Disks). The former are more expensive and require more advanced controllers: however they allow for both greater performance and reliability. The latter are much simpler and less expensive disk enclosures, whose task is just to allow computing units to mount them and use them as they were internal disks. Both units can be equipped with both rotational disks (HDD) or solid-state disks (SSD): usually a datacenter integrates all possible combinations of technologies to define different disk pools to be used for different purposes.

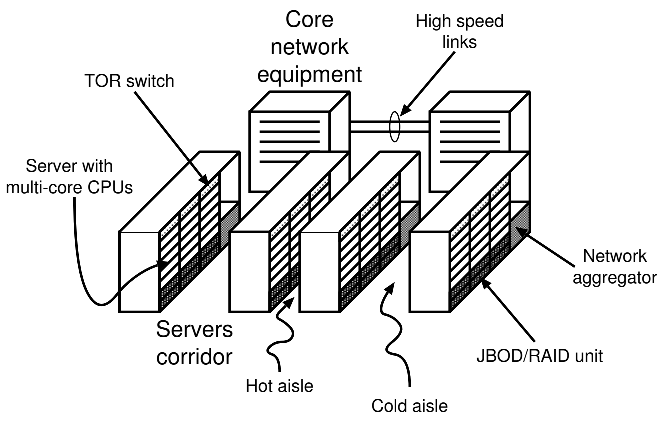

Fig. 1. Architecture of a datacenter.

Several network interconnection strategies for datacenters have been studied in the literature: a good survey can be found in [11]. In this work, we will mainly focus on the three-level network architecture as shown in Figure 2. Computing and storage units are directly connected to a switch that is defined as *Access switch* and that composes the so called *Access layer*. The switches can be positioned in two points that are usually addressed as *Top Of the Rack* (TOR) or *End Of the Line* (EOL). In the former, each rack has a switch, usually positioned in the top-most slot (for this reason it is called ”top of the rack”). It has the advantage of requiring a small number of shorter cables. However it can reduce performances by placing additional bottlenecks, and it can reduce the size of the infrastructure. In the other topology, switches are put at the end of each line of racks. EOL allows a greater scalability, but it is usually more expensive compared to the TOR solution. In our example we will focus on the TOR topology. Access switches are connected together using another level of switches, called *Aggregation layer*, that partitions the datacenter topology into a set of disjoint groups. In the example of Figure 1, aggregation switches are placed at the end of each corridor, and all the TOR access switches of the corresponding row of racks are connected to them. The connectivity of the datacenter is then completed by a further network, called the *Core layer*, that allows the communication between different aggregation switches. This organization however is affected by a problem known as the *bisection bandwidth*, which limits the maximum communication speed among nodes connected at different sides of the considered switch. Two techniques can be used to increase the available bisection bandwidth. Links that interconnect the different layers together might be characterized by different network technologies that might result in different link speeds. Since as the layer increases from access to aggregate, and from aggregate to core, the number of connected nodes increases as well, the bisection bandwidth can be increased by using faster communication technologies for links at higher levels. The second common way to increase the bisection bandwidth is by adding extra switches at each access layer, and using protocols like Equal Cost Multi-Path (ECMP) [35] to equally share the traffic among the different routes. For example, Figure 2 shows a 36 nodes architecture where each access switch is connected to three nodes (two for computing and one for storage), and access switches are grouped into bunches of three by the aggregation layer. Finally the four groups are connected together with the core layer. The bisection bandwidth is increased by using two aggregation layers per switch, and the by having three core layer switches.

In this work we mainly focus on a cloud datacenter, where all the computing nodes are used to host VMs. In our scenario, users are IaaS clients, that require the system to provide them a VM. Each user will then use the VM to run his software, and release it after use. As in a classical cloud scenario, VMs are started from *images*, that contain the filesystem of the OS plus all the other software that could be run in the VM. Persistent data are then stored using special block services set up by the cloud provider: they usually simulate the presence of a network connected disk that can be reached using the iSCSI protocol (a specific protocol that encapsulates SCSI data inside internet packets). For example, in *Openstack* [36], images are stored by a service called *Glance*, while persistent storage is provided by another service called *Cinder*. Both services use a lower-level block storage service (which in Openstack is called *Swift*). This file system architecture creates a high load over the network, which in many occasions becomes the real bottleneck of the system. In particular, the lifetime cycle of a VM, together with its storage access, is shown in Fig. 3. Initially, VM OS root disk images and persistent volume storage images are divided into blocks that are spread over the storage nodes of the datacenter (Fig. 3a). Root disk images size ranges from few tenth of MBs (for the smallest OS distributions) to several tenth of GB (for Windows based OS, or for more featured Linux installations). As soon as a VM is started, its root disk image is copied into a local drive of the computing node where the VM is run (Fig. 3b). This creates a strong utilization of the network, since GBs of data must be transferred inside the datacenter. After the image has been copied, the virtual machine manager can start the VM. Each OS running on a VM usually can access at least three different disks: the root disk that contain the OS, a fast but small local disk (called the *ephemeral storage*), and a remote persistent storage. The main characteristic of ephemeral disks is that they are not persistent: when the VM is released, they are cleaned, and all their content is lost. During normal operations, the VM accesses the locally connected disks: the root disk to install OS updates or other software that must be run on the VM; ephemeral disks to hold temporary data. In this case the network is usually accessed only to access the persistent storage (Fig. 3c). Even if the exact access pattern is cloud-architecture dependent, it is usually performed by locally caching the data, and only relatively large blocks of packed data are sent across the network. At the release of the VM, the resources required to hold both the root and the ephemeral disks must be released. The user might require to perform a snapshot of the root disk in order not to lose the OS updates that have been made during the VM execution. This process is called *shelfing* in Openstack terminology, and it requires that the new disk image must be transferred from the node that is releasing the VM to a storage node (Fig. 3d).

IV. MODELING APPROACH

Markovian Agents [37] are a formalism used to describe large system where elements can interact. Such models are solved using Mean Field Analysis [38]. In this case, agents do not communicate using messages, as ordinary Markovian Agents do, but they can influence each other via induction: the rate of jumping from one state to another can be influenced by the number of agents in a given state at a given location. Moreover, agents can increase in number or decrease (either spontaneously or induced by other agents), or they can multiply during the transitions.

The Markovian Agent based model depicted in Fig. 4 describes the behavior of a compute node. The system receives a total of Λ requests for activation of new VMs per time unit. Each node i receives requests at rate $\lambda_i(\Pi)$ (with $\sum_i \lambda_i(\Pi) = \Lambda$) where Π represents the count of agents in each state for each node. In particular, VMs are randomly assigned to nodes, with a probability that is proportional to the number of free VMs. Let us call $free_i(\Pi)$ the number of VMs that can still be assigned to node i, and let us call $\lambda_i(\Pi)$. We then have:

$$\lambda_i(\Pi) = \Lambda \frac{free_i(\Pi)}{\sum_j free_j(\Pi)}.$$

If the disk image of the starting VM is locally present, the agent goes in state *Local* with probability $p_{inCache}$ to simulate the immediate start of the computation. Otherwise, the agent goes in state $Startup_{ij}$ to represent the image transfer from storage node j to compute node i. The image is transferred at

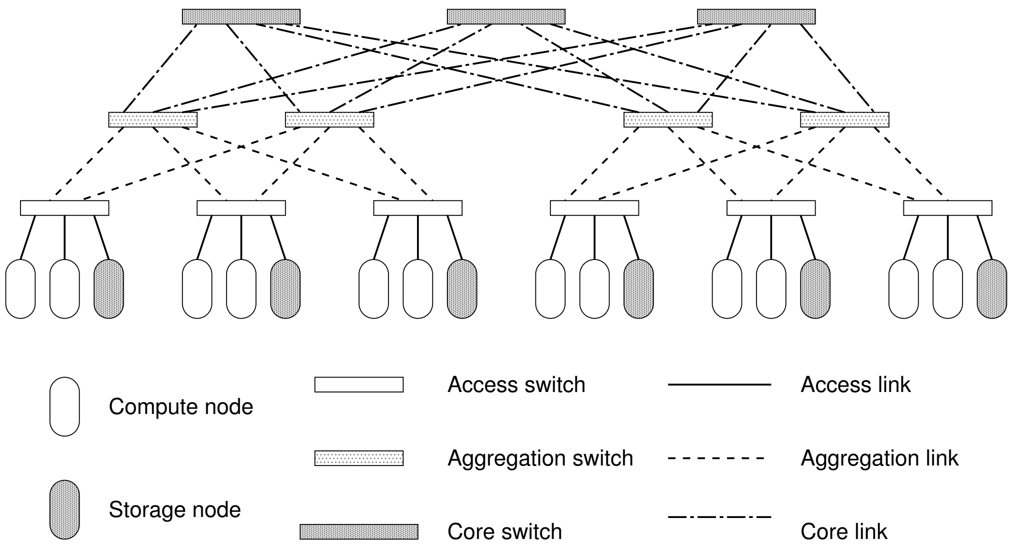

Fig. 2. Logical architecture of a three-tier datacenter interconnection network.

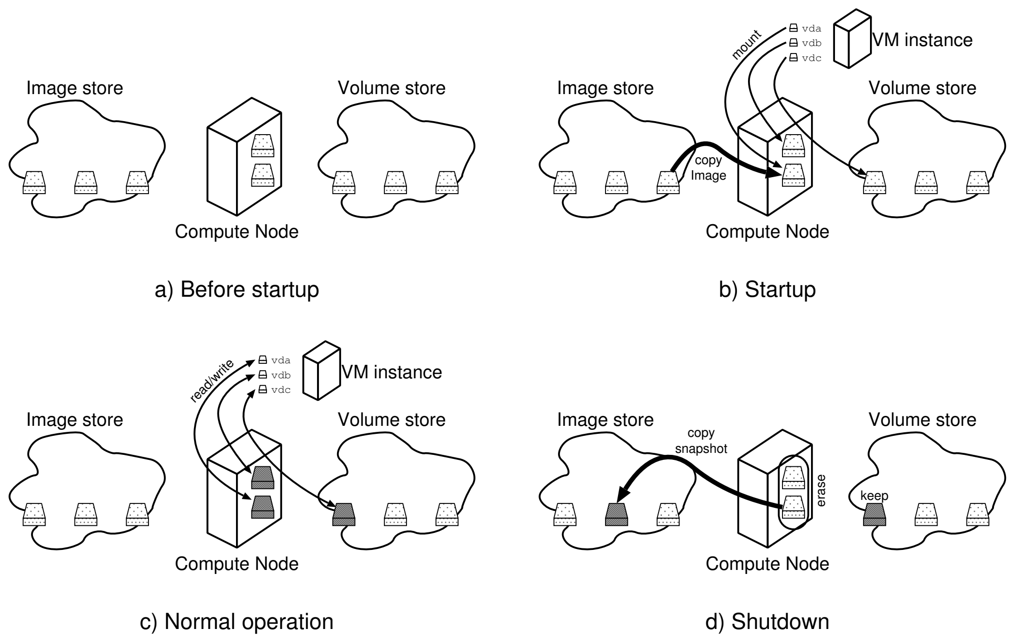

Fig. 3. Storage access during the lifetime of a VM: a) storage organization prior to startup, b) startup phase, c) disk access during normal operation, d) shutdown procedure.

rate $\sigma_{Startup_{ij}}$, that is equal to the speed of the link performing as bottleneck in the route connecting the compute node i to the storage one j. To be more precise, the computation of the speed of the route is computed in this way:

1) the total number of VMs $N_{R_{ij}}$ transferring data from each compute node i to each storage node j is computed;
2) let us call $\mathscr{R}_{l_k} = \{R_{ij}, \ldots\}$ the set of all routes R_{ij} that traverses a link l_k. The total number of VMs N_{l_k} using link l_k is computed by considering all the possible routes ij that traverses that link, and multiplied by the sharing factor $sh(R_{ij}, l_k)$ of that link in the communication: $N_{l_k} = \sum_{R_{ij} \in \mathscr{R}_{l_k}} N_{R_{ij}} \cdot sh(R_{ij}, l_k)$. Sharing factor $sh(R_{ij}, l_k)$ allows to model protocols like the ECMP;
3) for each link l_k, actual link speed σ_{l_k} is determined as $\sigma_{l_k} = \frac{C_{l_k}}{\max(N_{l_k}, 1)}$, where C_{l_k} is the maximum effective speed of link l_k measured in MB/s;
4) for each route R_{ij}, route speed $\sigma_{R_{ij}}$ is computed as the minimum capacity along the path: $\sigma_{R_{ij}} = \min_{l_k \in \mathscr{L}_{R_{ij}}} \sigma_{l_k}$, where $\mathscr{L}_{R_{ij}} = l_k$ is the set of link used by route R_{ij};
5) let us call $D_{VMimage}$ the average size of a VM image. Rate $\sigma_{Startup_{ij}}$ is then determined as $\sigma_{Startup_{ij}} = \frac{\sigma_{R_{ij}}}{D_{VMimage}}$.

When the transfer is completed, the agent goes in state *Local*. The VMs session has duration $1/\mu_{shutdown}$; once it is terminated the agent goes in state *Shutdown* with probability $1 - p_{noShelve}$ to model the user has performed the shelve action, otherwise the agent leaves the system. During the session, access to the remote disk can be requested by some applications at rate $\mu_{BlockIO}$. This behavior is modeled by the agent moving to the state *Block I/O*$_{ij}$. The agent returns to the normal state when the I/O is completed. This occurs at rate $\sigma_{BlockIO_{ij}} = \frac{\sigma_{R_{ij}}}{D_{Block}}$, where D_{Block} is the average I/O block size. If the VM requests to shelf the new image, the time spent to leave the system with rate $\sigma_{Shutdown}$ considers the copy of the image snapshot of size $D_{VMsnapshot}$. Also in this case the transfer speed depends on the bottleneck link between the computation and the storage nodes, and it is defined as $\sigma_{Shutdown} = \frac{\sigma_{R_{ij}}}{D_{VMsnapshot}}$

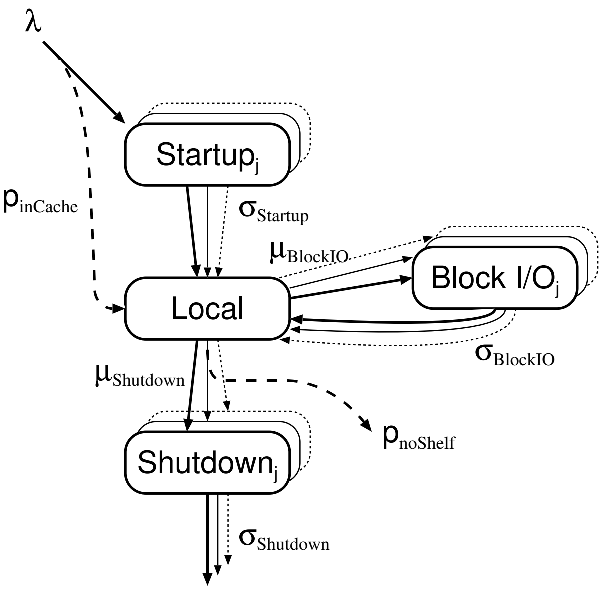

Fig. 4. Agent model of a compute node.

V. A CASE STUDY

To test the methodology, we apply the model proposed in Section IV to study the performances of four different patterns for placing computing and storage nodes inside a datacenter. In particular the considered scenarios are:

- **Distributed storage** (Fig. 5a). In this case there are no specific storage nodes, and disks are held directly inside the computing node. In this scenario, computing nodes have a slightly more limited capacity in term of VMs they can run, since they must also handle part of the storage requests.
- **Storage on rack** (Fig. 1). Each rack has its own set of storage nodes. This means that computing node might reach part of the storage nodes using just the switch at the corresponding access layer.
- **Storage on aisle** (Fig. 5b). In each aisle, there is a rack that includes all the storage nodes. In this way, computing nodes must use switches at the aggregation layer to reach their storage units. However they can be easier to maintain, since disks are located in a limited number of locations (i.e. one per aisle).
- **Storage in a given area** (Fig. 5c). Storage nodes are concentrated in a specific aisle (which might also be physically located in a different room with respect to the computing nodes). This has the disadvantage that storage nodes can be accessed only passing through the core layer. However it ensures a higher security, allowing the storage to be located in different places.

All the scenarios share the same number of nodes $N_{nodes} = 36$, and the same maximum number of VMs that can be run on the infrastructure $N_{VMs} = 1920$. Scenarios 2, 3 and 4 are characterized by $N_{compute} = 24$ computing nodes, $N_{storage} = 12$ storage nodes, and each node has the capacity of running up to $N_{VMs \times Node} = 80$ VMs. In scenario 1, all nodes act both as computing and storage device: for this reason their capacity of running VMs $N_{VMs \times Node}$ has been reduced accordingly to keep the maximum capacity of the system $N_{VMs} = 1920$ as in the other scenarios. Links are characterized by the following speed: $C_{l_{Access}} = 500$ Mb/s at the access layer, $C_{l_{Aggr}} = 500$ Mb/s at the aggregation layer and $C_{l_{Core}} = 500$ Gb/s at the core layer[1]. The average VM image size has been set to $D_{VMimage} = 80$ GB, while the snapshot size for VMs that are shelved (i.e. the difference from their original image) has been set to $D_{VMsnapshot} = 50$ MB. VMs perform block I/O on the average $\mu_{BlockIO} = 1$ block/h, and the block size is $D_{Block} = 10$MB. Requests of new VMs activations arrive at a rate varying in the range $\Lambda = 5..30$ req./h, and each VM has an average duration of $1/\mu_{shutdown} = 25$h. The probability of having a VM in cache is $p_{inCache} = 0.1\%$, and the probability of not shelving a terminating VM is $p_{noShelve} = 90\%$.

Figures 6, 7 and 8 show the average, minimum and maximum utilization of the links respectively at the access, aggregation and core layer. At the access layer, Scenarios 3 and 4 have a higher utilization since nodes produce a higher traffic to obtain the VMs due to the distribution of storage nodes. At the access layer, the only one having a lower utilization is Scenario 4, that is also the only one producing traffic at the core layer. In this case, however, the smaller utilization is due to the fact that the system saturates and creates a bottleneck for some nodes at the access layer.

Figures 9, 10 and 11 show the number of free VMs, the number of VMs performing IO operations (i.e. VMs laying in Startup, BlockIO, and Shutdown states), and the number of VMs in normal activities, respectively. As explained above, Scenarios 1 and 2 are more stable, as a consequence of the distribution of computing and storage nodes. It follows that they have a higher number of VMs performing the normal activity whereas in Scenarios 3 and 4 there is a higher number of VMs executing IO operations.

[1]These speeds roughly corresponds to the maximum effective throughput that can be achieved on standard 1GB and 10GB Ethernet technologies

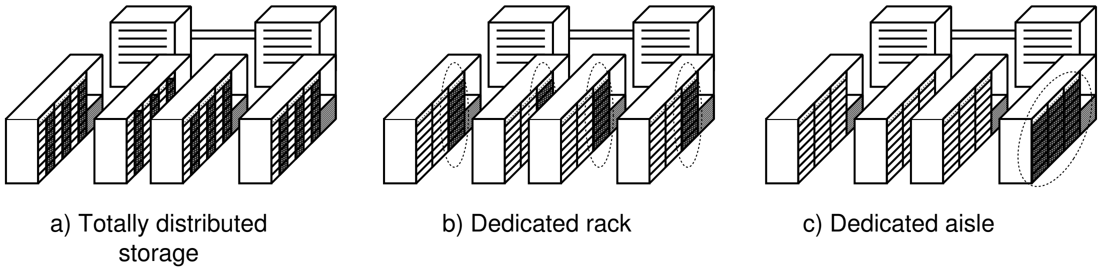

Fig. 5. Three alternative storage device organizations: a) storage is co-located with the computing nodes, b) storage is on a dedicated rack, c) storage is on a dedicated aisle.

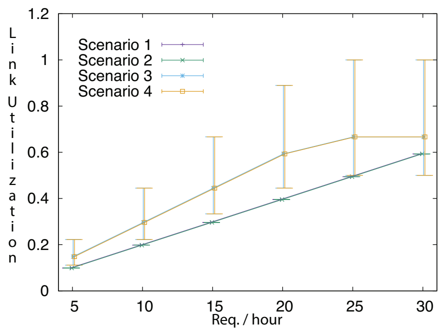

Fig. 6. Utilization of the links at access layer

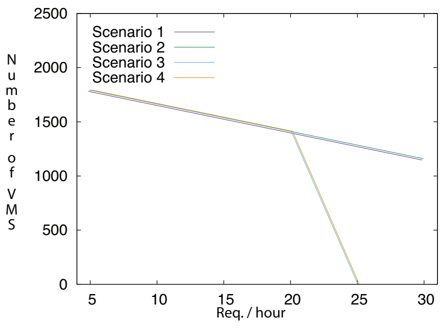

Fig. 9. Free VMs

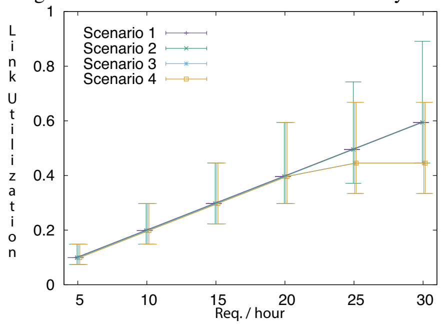

Fig. 7. Utilization of the links at aggregation layer

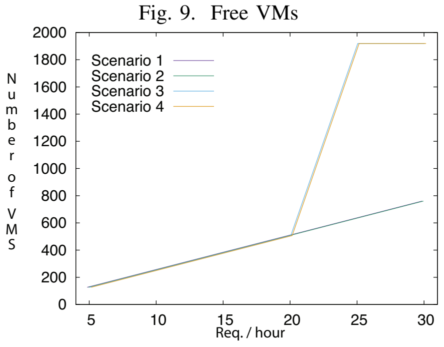

Fig. 10. VMs performing IO operations (Startup-Block-Shutdown)

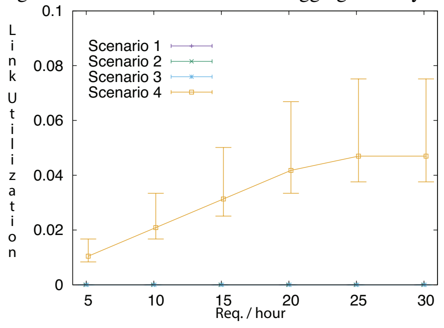

Fig. 8. Utilization of the links at core layer

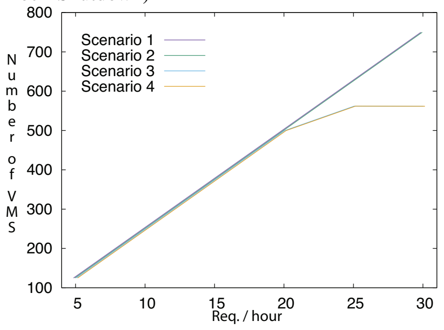

Fig. 11. VMs performing normal activity

VI. CONCLUSIONS

In this paper we proposed an approach for performance evaluation of the effects of networks in clouds. Our results, at the best of our knowledge, allow researchers and practitioners to model higher scale cloud systems with respect to previous literature, including architectures composed of more than one data center. Future works include the integration with our previous proposals for a detailed overall evaluation of all aspects of cloud systems. Moreover, we will study other network topologies that rely on commodity hardware such Fat-tree organizations, to see if alternative to the three layer architectures could improve the performance and reduce the cost of a data-center.

References

[1] D. A. Menasce', "Virtualization: Concepts, applications, and performance modeling," in *Proc. of The Computer Measurement Groups 2005 International Conference*, 2005.

[2] M. Gribaudo, P. Piazzolla, and G. Serazzi, "Consolidation and replication of vms matching performance objectives," in *Analytical and Stochastic Modeling Techniques and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7314, pp. 106–120.

[3] N. Huber, M. Von Quast, F. Brosig, and S. Kounev, "Analysis of the performance-influencing factors of virtualization platforms," in *Proc. of the 2010 international conference on On the move to meaningful internet systems: Part II*, ser. OTM'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 811–828.

[4] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang, "Probabilistic performance modeling of virtualized resource allocation," in *Proc. of the 7th international conference on Autonomic computing*, ser. ICAC '10. NY, USA: ACM, 2010, pp. 99–108.

[5] F. Benevenuto, C. Fernandes, M. Santos, V. A. F. Almeida, J. M. Almeida, G. J. Janakiraman, and J. R. Santos, "Performance models for virtualized applications." in *ISPA Workshops*, ser. Lecture Notes in Computer Science, G. Min, B. D. Martino, L. T. Yang, M. Guo, and G. Rnger, Eds., vol. 4331. Springer, 2006, pp. 427–439.

[6] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Koodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Generation Computer Systems*, vol. 51, pp. 61 – 71, 2015.

[7] A. Sfrent and F. Pop, "Asymptotic scheduling for many task computing in big data platforms," *Information Sciences*, vol. 319, pp. 71 – 91, 2015.

[8] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "A cluster-based data-centric model for network-aware task scheduling in distributed systems," *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 755–775, 2014.

[9] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, Aug. 2015.

[10] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[11] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. I. Iqbal, C. Xu, and A. Y. Zomaya, "Quantitative comparisons of the state-of-the-art data center architectures," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1771–1783, 2013.

[12] R. D. S. Couto, S. Secci, M. E. M. Campista, and L. H. M. K. Costa, "Reliability and survivability analysis of data center network topologies." *CoRR*, vol. abs/1510.02735, 2015.

[13] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, Aug. 2009.

[14] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "Conga: Distributed congestion-aware load balancing for datacenters," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 503–514, Aug. 2014.

[15] F. Palmieri, U. Fiore, S. Ricciardi, and A. Castiglione, "Grasp-based resource re-optimization for effective big data access in federated clouds," *Future Generation Computer Systems*, vol. 54, pp. 168–179, 2016.

[16] S. Spoto, M. Gribaudo, and D. Manini, "Performance evaluation of peering-agreements among autonomous systems subject to peer-to-peer traffic," *Perform. Eval.*, vol. 77, pp. 1–20, 2014.

[17] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Zomaya, "Performance and energy efficiency metrics for communication systems of cloud computing data centers," *IEEE Trans. on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[18] P. Ruiu, A. Bianco, C. Fiandrino, P. Giaccone, and D. Kliazovich, "Power comparison of cloud data center architectures," in *Proc. of the 2016 IEEE International Conference on Communications (ICC)*, 2016.

[19] E. Barbierato, M. Gribaudo, and M. Iacono, "A performance modeling language for big data architectures." in *ECMS*, W. Rekdalsbakken, R. T. Bye, and H. Zhang, Eds. European Council for Modeling and Simulation, 2013, pp. 511–517.

[20] ——, "Performance evaluation of NoSQL big-data applications using multi-formalism models," *Future Generation Computer Systems*, vol. 37, no. 0, pp. 345–353, 2014.

[21] ——, "Modeling apache hive based applications in big data architectures," in *Proc. of the 7th International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools '13. ICST, Brussels, Belgium: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2013, pp. 30–38.

[22] D. Cerotti, M. Gribaudo, M. Iacono, and P. Piazzolla, "Modeling and analysis of performances for concurrent multithread applications on multicore and graphics processing unit systems," *Concurrency and Computation: Practice and Experience*, pp. n/a–n/a, 2015.

[23] ——, "Workload characterization of multithreaded applications on multicore architectures," in *ECMS, Proc. of the 28th European Conference on Modelling and Simulation, ECMS 2014, Brescia, Italy, May 27-30, 2014*. European Council for Modeling and Simulation, 2014, pp. 480–486.

[24] M. Gribaudo, M. Iacono, and D. Manini, "Improving reliability and performances in large scale distributed applications with erasure codes and replication," *Future Generation Computer Systems*, vol. 56, pp. 773 – 782, 2016.

[25] ——, "Modeling replication and erasure coding in large scale distributed storage systems based on CEPH," in *Proc. of XII conference of the Italian chapter of AIS*, ser. Lecture Notes in Information Systems and Organisation. Springer Berlin / Heidelberg, 2016, vol. to appear.

[26] A. Castiglione, M. Gribaudo, M. Iacono, and F. Palmieri, "Exploiting mean field analysis to model performances of big data architectures," *Future Generation Computer Systems*, vol. 37, no. 0, pp. 203–211, 2014.

[27] ——, "Modeling performances of concurrent big data applications," *Software: Practice and Experience*, vol. 45, no. 8, pp. 1127–1144, 2015.

[28] E. Barbierato, M. Gribaudo, and M. Iacono, "Modeling and evaluating the effects of Big Data storage resource allocation in global scale cloud architectures," *International Journal of Data Warehousing and Mining*, vol. 12, no. 2, pp. 1–20, 2016.

[29] E. Barbierato, G.-L. D. Rossi, M. Gribaudo, M. Iacono, and A. Marin, "Exploiting product forms solution techniques in multiformalism modeling," *Electronic Notes in Theoretical Computer Science*, vol. 296, no. 0, pp. 61 – 77, 2013.

[30] M. Gribaudo and M. Iacono, "A different perspective of agent-based techniques: Markovian agents," in *Intelligent Agents in Data-intensive Computing*, ser. Studies in Big Data, J. Kolodziej, L. Correia, and J. Manuel Molina, Eds. Springer International Publishing, 2016, vol. 14, pp. 51–70.

[31] "OpenStack: Images and instances," http://docs.openstack.org/admin-guide-cloud/compute-images-instances.html, accessed: 2016-02-06.

[32] "Where to find OpenStack cloud images," https://thornelabs.net/2014/06/01/where-to-find-openstack-cloud-images.html, accessed: 2016-02-06.

[33] "OpenStack networking tutorial: Single-host FlatDHCPManager," https://www.mirantis.com/blog/openstack-networking-single-host-flatdhcpmanager/, accessed: 2016-02-06.

[34] "Ubuntu documentation: CloudInit," https://help.ubuntu.com/community/CloudInit, accessed: 2016-02-06.

[35] C. Hopps, "Analysis of an equal-cost multi-path algorithm," United States, 2000.

[36] "OpenStack web site," https://www.openstack.org/, accessed: 2016-02-06.

[37] M. Gribaudo, D. Cerotti, and A. Bobbio, "Analysis of on-off policies in sensor networks using interacting markovian agents." in *Proc. of the 4th International Workshop on Sensor Net-works and Systems for Pervasive Computing - PerSens 2008*, 2008.

[38] B. M. and L. J.Y., "A class of mean field interaction models for computer and communication systems." *Performance Evaluation*, vol. 65, no. 11-12, pp. 823–838, 2008.