SLA-Driven Simulation of Multi-Tenant Scalable Cloud-Distributed Enterprise Information Systems

Alexandru-Florian Antonescu[12], Torsten Braun[2]
alexandru-florian.antonescu@sap.com, braun@iam.unibe.ch

[1]SAP (Schweiz) AG
Products & Innovation, Next Applied Research
Althardstrasse 80, 8105 Regensdorf, Switzerland

[2]University of Bern
Communication and Distributed Systems
Neubrückstrasse 10, 3012 Bern, Switzerland

Cloud Computing is an enabler for delivering large-scale, distributed enterprise applications with strict requirements in terms of performance. It is often the case that such applications have complex scaling and Service Level Agreement (SLA) management requirements. In this paper we present a simulation approach for validating and comparing SLA-aware scaling policies using the CloudSim simulator, using data from an actual Distributed Enterprise Information System (dEIS). We extend CloudSim with concurrent and multi-tenant task simulation capabilities. We then show how different scaling policies can be used for simulating multiple dEIS applications. We present multiple experiments depicting the impact of VM scaling on both datacenter energy consumption and dEIS performance indicators.

Categories and Subject Descriptors: C.2.4 [**Distributed Systems**] Cloud Computing; K.6.4 [**System Management**] Scaling; I.6.7 [**Simulation Support Systems**] Parallel

General Terms: Simulation

Additional Key Words and Phrases: Cloud Computing, Service Level Agreement, Scaling

1. INTRODUCTION

Cloud Computing [Mell and Grance 2011] is an enabler for delivering large-scale, distributed enterprise applications with strict requirements in terms of performance. It is often the case that such applications have complex scaling and Service Level Agreement (SLA) management requirements. As example, distributed Enterprise Information Systems [Woods 2003] often interact with large-scale distributed databases, requiring distributed processing of results coming from multiple systems. Specific to cloud environments is the distribution of available cloud resources among multiple tenants, each running a specific set of applications with different workload patterns and SLA requirements.

The advent of Internet applications created new requirements for distributed software running in cloud environments, as the heterogeneity of physical computing infrastructure and application workloads increased. Thus, there is a need for testing the impact of different task allocation and resource scheduling policies on the performance of distributed applications running in these distributed environments.

One way of achieving these goals of quantifying the performance impact of different resource allocation policies in cloud environments is by using simulations based on previously recorded performance monitoring traces. We extend our work in [Antonescu and Braun 2014b] by using the Distributed Enterprise Information System (dEIS) application model for simulating different SLA-based resource scaling policies in the CloudSim [Lab 2014] simulator environment.

Our main contributions can be summarized as follows. We extend the CloudSim simulator with support for (1) multiple tenants with a dynamic number of VMs, and for (2) concurrent-tasks simulation at Virtual Machine (VM) level. We also present how to simulate different SLA scaling policies for multiple cloud tenants.

The rest of our paper is organized as follows. Section 2 presents the related work in the field of cloud simulators as well as modeling and simulation of distributed cloud applications. Section 3 introduces the SLA scaling algorithms and CloudSim extensions required for supporting dynamic resource allocation and scaling. Section 4 presents the evaluation results, and finally, Section 5 draws conclusions and gives future research directions.

2. RELATED WORK

There are many publications describing modeling approaches at simulation of cloud infrastructures and applications, focusing on both infrastructure modeling and resource utilization in virtual machines. We present a short overview of some of these works, along with a short description of CloudSim.

Sandhu et al. [Sandhu et al. 2013] present an approach at modeling dynamic workloads in CloudSim by considering both random-non-overlap and workload-based profile policies, similar to our approach for modeling dynamic enterprise workloads. However, we extend this work by considering SLA-based VM scaling, multiple tenants and concurrent execution/simulation of application tasks.

Buyya et al. [Buyya et al. 2009] describe a model for simulating cloud resources using CloudSim. They focus on modeling physical and virtual cloud resources, such as physical servers, virtual machines, tasks, allocation and scheduling policies. However, they do not focus on describing application performance models, VM scaling policies or dynamic VM instantiation in CloudSim.

Long et al. [Long et al. 2013] present the requirements for evaluating cloud infrastructures using CloudSim. They describe VM CPU utilization models using monitoring information gathered from a set of experiments, and use a power model of representing the energy consumption model in physical hosts. While we also employ a similar approach at modeling VM dynamic workload, we extend this by considering also the concurrent application workload, as well as considering a dynamic allocation of cloud resources by using SLA scaling.

2.1 CloudSim Cloud Simulator

As we used the CloudSim for running the allocation and scaling simulations, we describe briefly its architecture and main components.

CloudSim [Lab 2014] positions itself as a simulator for both cloud applications and infrastructure. It accomplishes this by allowing modeling of hardware and software cloud resources. Among the modeled physical entities there are: hosts, network links and datacenters, while the modeled software entities are: virtual machines (VMs), brokers and cloudlets (tasks). This is achieved by offering the mentioned entities as Java classes that can be extended according to simulation requirements. The simulator is implemented using discrete events communication, fired at a specified minimum time interval. Cloud tasks (cloudlets) are created by *brokers*, which send them to VMs for execution on the resources of the hosts forming the datacenter. Upon completion of each cloudlet's execution, its parent broker is notified.

In CloudSim, a *datacenter* is composed of (1) a collection of *hosts*, (2) *storage* devices, (3) a policy controlling the allocation of *VMs* to hosts, and (4) resource utilization costs for comsumed computing time, memory, storage and network bandwidth. Each host is defined by (1) its number of *processing element*s and their Millions Instructions Per Second (MIPS) rating, (2) RAM memory size, (3) storage size, (4) network bandwidth, and (5) *VM scheduling policy*. As VM allocation policies it supports (1) time-shared, (2) space-shared, and (3) time-shared with over-subscription. The datacenter network is given using a BRITE [Medina et al. 2001] network topology specification.

VMs are described by their requirements in terms of (1) number of CPU cores and MIPS rating, (2) memory size, (3) network bandwidth, (4) virtual machine manager, and (5) cloudlet execution policy. There are four built-in cloudlet execution policies: (1) time-shared, (2) space-shared, (3)

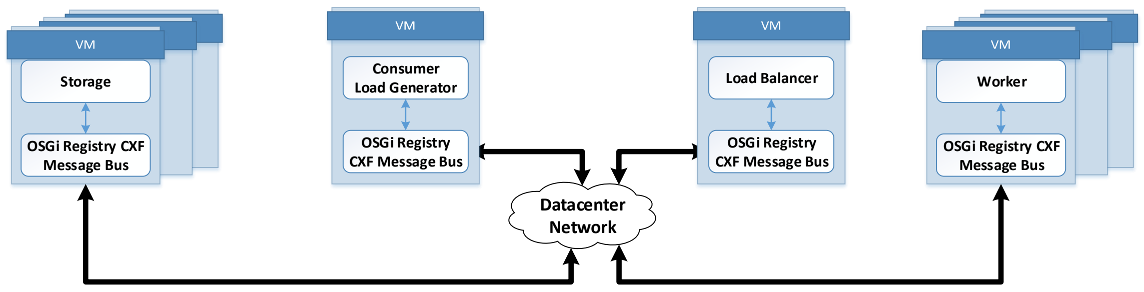

VM
Storage
OSGi Registry CXF Message Bus
VM
Consumer Load Generator
OSGi Registry CXF Message Bus
VM
Load Balancer
OSGi Registry CXF Message Bus
VM
Worker
OSGi Registry CXF Message Bus
Datacenter Network

Fig. 1. dEIS Architecture

dynamic-workload, and (4) network space-shared. The CloudSim API allows for easy development of new cloudlet execution policies.

CloudSim models cloud tasks as *cloudlet* entities, defined by (1) computing requirements given as number of processing elements, and computing task length given in MIPS, (2) network bandwidth consumption for input and output, (3) CPU utilization model, (4) memory utilization model, and (5) network bandwidth utilization model.

Cloudlets are generated by *Datacenter Brokers*, which are equivalent to cloud services. Each broker controls one or more VMs and it implements a selection algorithm for choosing which VM to receive a given cloudlet. The broker also implements the algorithm for reacting to the completion of various cloudlets it has generated.

For simulating a distributed application, one must create one or more cloud brokers and implement the algorithms for generating cloudlets, as well as handling their completion. Also, at least one datacenter needs to be defined, including its hosts and network. In Section 2.2 we present the architecture of such a distributed enterprise application.

2.2 Distributed Enterprise Information System Architecture

A typical dEIS application consists of the following tiers, each contributing to the SLA management problem: consumer/thin client, load balancer, business logic and storage layer. Fig. 1 provides an overview of the overall EIS topology. We shortly present the structure of the EIS system used, with more details found in [Antonescu et al. 2012], [Antonescu and Braun 2014a]. This class of systems is representative for core enterprise management systems, such as ERP [Leon 2008].

As representative dEIS distributed application we used the one described in ([Antonescu and Braun 2014a], [Antonescu and Braun 2014b], [Antonescu et al. 2013], [Antonescu et al. 2012], [Antonescu et al. 2013]). Targeted dEIS system is composed of four core services: one or more Thin Clients (CS), a Load Balancer (LB), one or more Worker services (WK), and one or more Database Storage services (ST). Each service runs in its own VM and communicates asynchronously with the other services using a distributed service messaging bus.

The CS service contains the graphical user interface, as well as logic for initiating data sessions and issuing requests. The LB service provides load balancing logic, while also maintaining session information about connected clients. The WK services implement data queries, analysis, transactional and arithmetic logic for the application. The ST service contains interfaces and mechanisms for creating, reading, updating and deleting store data. A detailed presentation of the performance model of dEIS can be found in [Antonescu and Braun 2014b].

3. SLA-DRIVEN DISTRIBUTED SYSTEMS SCALING

In this section we present the SLA-based scaling approach for managing VMs. We first describe the Parallel CloudSim Cloudlet Scheduler (PCCS) in subsection 3.1, and then, in subsection 3.2 we present the SLA scaling manager used for dynamically creating VMs based on SLA policies and application performance indicators.

3.1 Parallel CloudSim Cloudlet Scheduler

Out-of-the-box, the CloudSim simulator does not support either simulation of time-based application level tasks, nor parallel simulation of multiple tasks in a VM. For these reasons, we developed a new time-shared CloudSim cloudlet scheduler, which works as follows. First, the application task's duration is converted from milliseconds to MIPS by considering the MIPS rating of the CPU originally running the task and the average CPU utilization during its execution, as defined by Eq. 1

$$cl_{MIPS} = \frac{1000 \cdot cl_{ms}}{CPU_{MIPS}} \cdot \overline{CPU} \tag{1}$$

where cl_{MIPS} is the calculated cloudlet's MIPS length, cl_{ms} is the cloudlet's duration in milliseconds when executed on a CPU with a MIPS rating of CPU_{MIPS} and an average utilization of $\overline{CPU}$.

Next, the PCCS will calculate the available CPU MIPS capacity cap of the current time slice ts using the formula presented in Eq. 2

$$cap = CPU^{VM}_{MIPS} \cdot cores \cdot ts \tag{2}$$

where CPU^{VM}_{MIPS} is the MIPS capacity of the CPU belonging to the VM executing the cloudlet, and $cores$ is the number of cores of the CPU.

Finally, the PCCS will evenly distribute the available MIPS resources between all running cloudlets, each cloudlet receiving a MIPS amount equal to cap/n, where n is the number of active cloudlets in the considered time slice. During our simulations the scheduling time slice was equal to 1 millisecond.

3.2 SLA-Based Scaling Manager

The SLA Scaling Manager (SSM) is responsible for dynamically adjusting the number of VMs for each of the services of the distributed applications and for each of the cloud tenants. It accomplishes this using invariant conditions formed with terms obtained from the performance indicators of the services running in VMs. An example of an invariant condition can be: "average distributed transaction execution time is below one second". The threshold contained in the SLA invariant is then used by the SSM for determining the conditions for performing either a scale-out action [Ferre 2004] (creating one or more VMs), or a scale-in action (terminating one or more VMs).

The SSM operates according to Algorithm 1, mainly by calculating the SLA ratio sr as the factor by which the average over the moving time window W of SLA metric m is approaching its maximum threshold $max_{SLA}(m)$. If sr is above a given threshold S^{UP} (e.g. 0.9) and sr is increasing from the last check then a scale-out operation is flagged. Similarly, if sr is below a threshold S^{DOWN} (e.g. 0.6) and sr is decreasing, then a scale-in operation is flagged. Either scale-out or scale-in operations will be executed only if the number of such operations ss is below a given threshold ss^{MAX} (e.g. 2) in the last W_S seconds (e.g. 40 sec, chosen as 1.5 times the time it takes for a VM to become fully operational), for ensuring system stability by preventing (1) fast-succeeding transitory scale-in and scale-out actions, and (2) oscillations in the number of VMs.

For the scale-in operation it is notable that the VM selected for shutdown (with lowest utilization value) is not immediately terminated, but first its broker is informed about the scale-in operation for

ALGORITHM 1: SSM Scaling algorithm

```
Data: per tenant: SLA scaling thresholds, monitoring information
Result: per tenant: scale-out, scale-in VM operations
while not at end of simulation do
    foreach tenant $e$ do
        calculate average value $\overline{m_{SLA}}$ of SLA metric $m$ over the sliding time window $W$
        $\overline{m_{SLA}} \leftarrow average\,(m(t), m(t-1), ..., m(t-W))$;
        calculate SLA ratio $sr$ for metric $m$: $sr \leftarrow \frac{\overline{m_{SLA}}}{max_{SLA}(m)}$;
        evaluate scale-out condition: $up \leftarrow (sr > s^{UP}) AND (sr(t) > sr(t-W_S))$;
        evaluate scale-in condition: $down \leftarrow (sr < S^{DOWN}) AND (sr(t) < sr(t-W_S))$;
        calculate scaling speed $ss$ as the number of scale-out or scale-in operations in the last time window $W_S$;
        if $(up = true)\, AND\, (ss < ss^{MAX})$ then
            create new VM;
        else if $(down = true)\, AND\, (ss < ss^{MAX})\, AND (count(VM) > 1)$ then
            select $vm$ for shutdown with lowest load;
            inform $VM$'s broker of imminent shutdown for preventing sending to workload to $VM$;
            wait for $T$ seconds before shutting-down the $VM$;
        end
    end
    schedule next scheduling check;
end
```

preventing new load being sent to the VM and then after a given time period T (e.g. 10 seconds), during which tasks running in the VM will get a chance to complete, the VM is finally terminated.

4. EVALUATION RESULTS

In order to evaluate the integration of the SLA Scaling Manager and Parallel Cloudlet Scheduler into CloudSim simulator we ran three different simulations testing both the handling of multiple cloud tenants and the ability to scale the number of VMs according to high-level SLAs.

4.1 Simulation 1

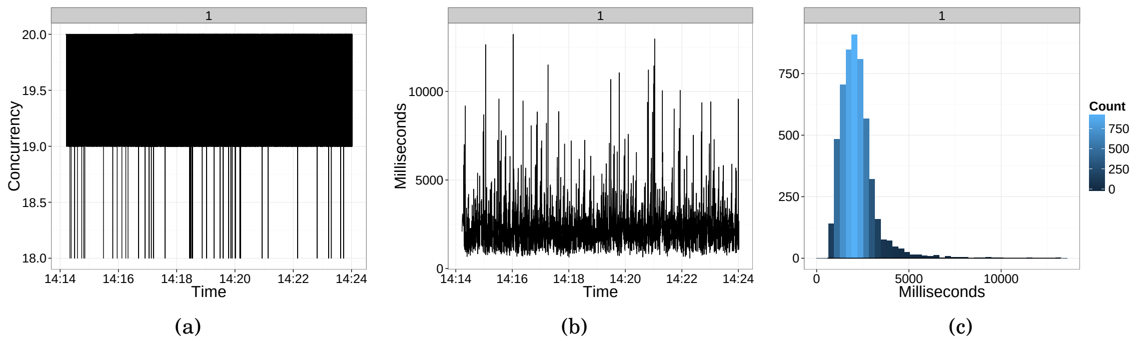
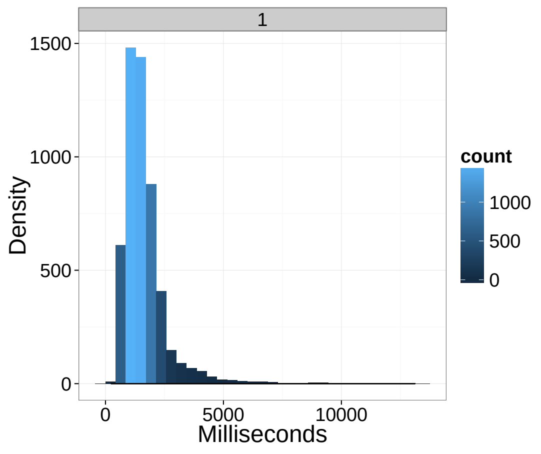
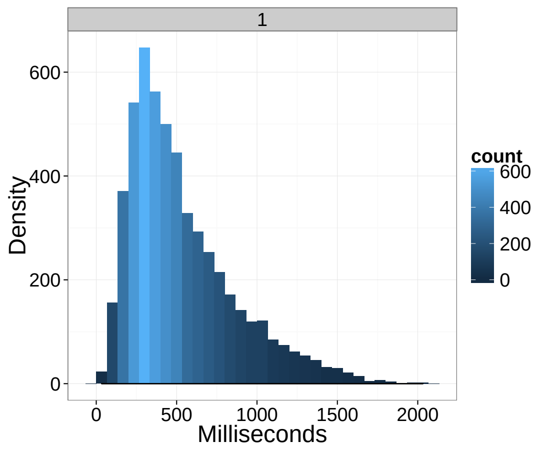


Fig. 2. Simulation 1 (a) CS Concurrent Load (b) CS Response Time (c) Histogram of CS Response Times

In this simulation we considered a single cloud tenant running a constant load of 20 concurrent distributed transactions, with the dEIS system configured to use only one VM for each service. We



(a)

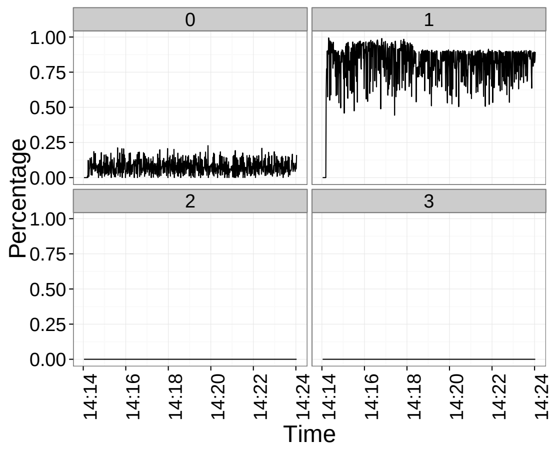


(b)

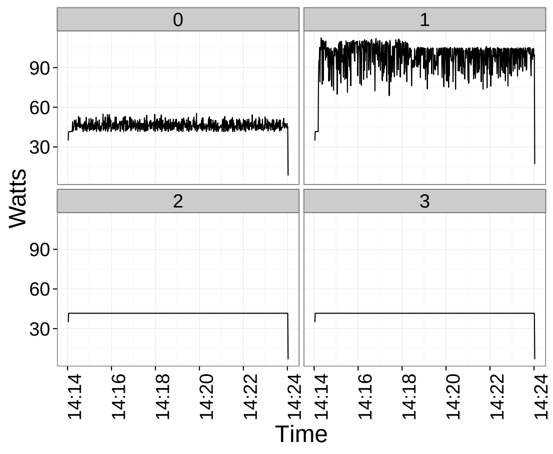
Fig. 3. Simulation 1 (a) Distribution of WK execution times (b) Distribution of ST execution times

used this simulation as a base for comparing the others simulations where we will introduce multi-tenancy and varying scaling conditions.

In Fig. 2a we display the simulated load, which varied between 19 and 20 concurrent requests due to asynchronous sampling of the number of active dEIS requests. Fig. 2b shows the response time measured at the CS service for each distributed concurrent transaction, under the given workload. As the simulation model uses datasets from a real distributed application [Antonescu and Braun 2014b], it has a rather large variance. Fig. 2c displays the distribution of response times at CS service. The average CS response time was 2200ms at the considered workload level of 20 concurrent transactions per second.

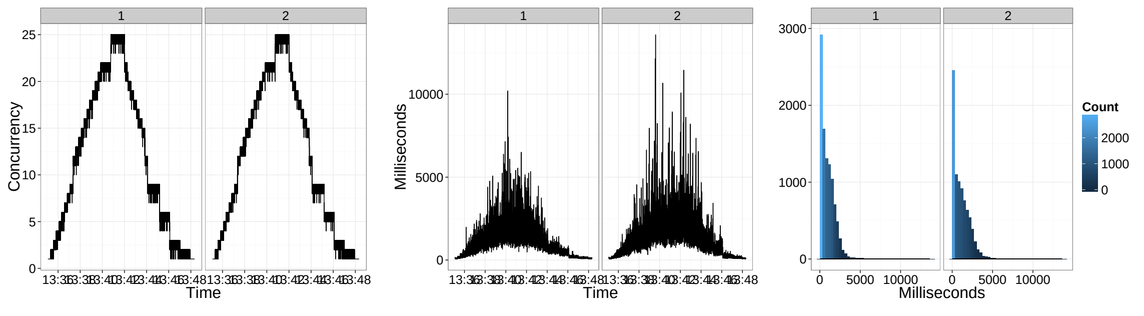


(a)



(b)

Fig. 4. Simulation 1 (a) Hosts CPU Utilization (b) Hosts Power Consumption



(a) (b) (c)

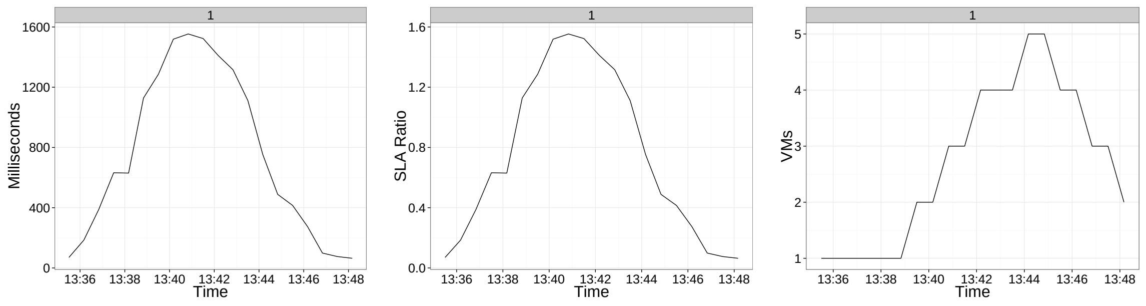
Fig. 5. Simulation 2 - Per tenant, CS service (a) Concurrent Load (b) Execution Time (c) Histogram of Execution Times

In Fig. 3a we display the execution times at the WK service, respectively in Fig. 3b at the ST service. This shows the breakdown of total transaction execution time between WK and ST services, with an average WK execution time of 1652ms and a standard deviation of 1025ms, and an average of 533ms and standard deviation of 329ms for the ST service respectively.

Fig. 4a displays the hosts' average CPU utilization as calculated from the VMs' CPU utilization, while Fig. 4b shows the energy consumption of the hosts, by considering a linear dependency model between the CPU utilization and host's power consumption. It is important to note the fact that hosts with no active utilization still consumed a large amount of energy.

Fig. 9a shows the total power consumption at datacenter level, summed over every second, for the first simulation. The total simulated energy consumption had a value of 335.9 KJ, distributed evenly across the entire simulation duration, as a consequence of the constant workload.

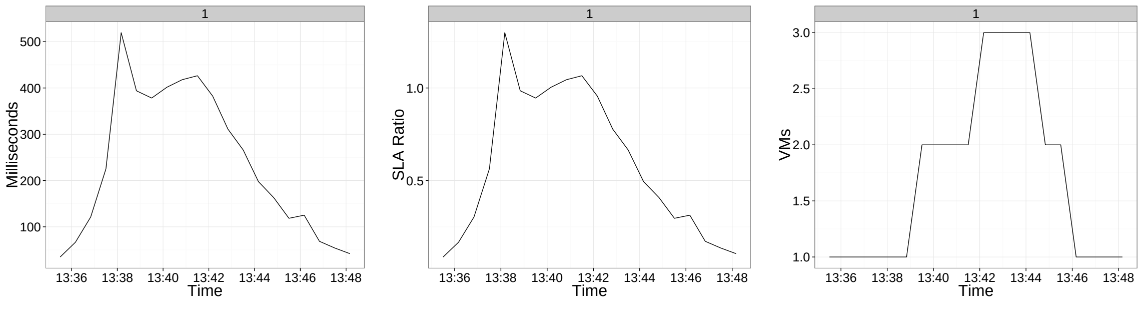
4.2 Simulation 2



(a) (b) (c)

Fig. 6. Simulation 2 - WK service (a) average execution (b) SLA Scaling Ratio (c) Number of VMs

In the second simulation we considered two cloud tenants (client-organization), each executing a varying workload as shown in Fig. 5a, which increased from 1 to 20 concurrent transactions and then decreased back to 1. The first tenant (#0) executed its workload on a fixed virtual infrastructure (static number of VMs/ no scaling), while the second tenant (#1) had scaling enabled at 1000ms for WK the service, and at 400ms for the ST service respectively.



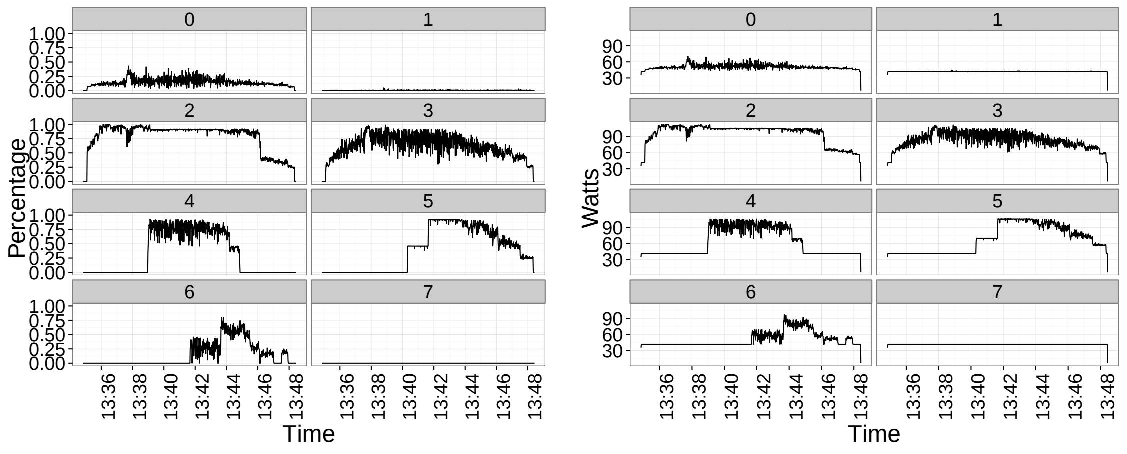
(a) (b) (c)

Fig. 7. Simulation 2 - ST service (a) average execution time (b) SLA Scaling Ratio (c) Number of VMs

As shown in Fig. 5b and Fig. 5c, tenant #1 (SLA scaling enabled) had a lower average transaction execution time of 965.8 ms, compared to tenant #0, who had an average execution time of 1144.1 ms. This shows the advantages of running the cloud workload under SLA conditions on a dynamically scaled infrastructure, compared to running it on a fixed-sized virtual infrastructure.

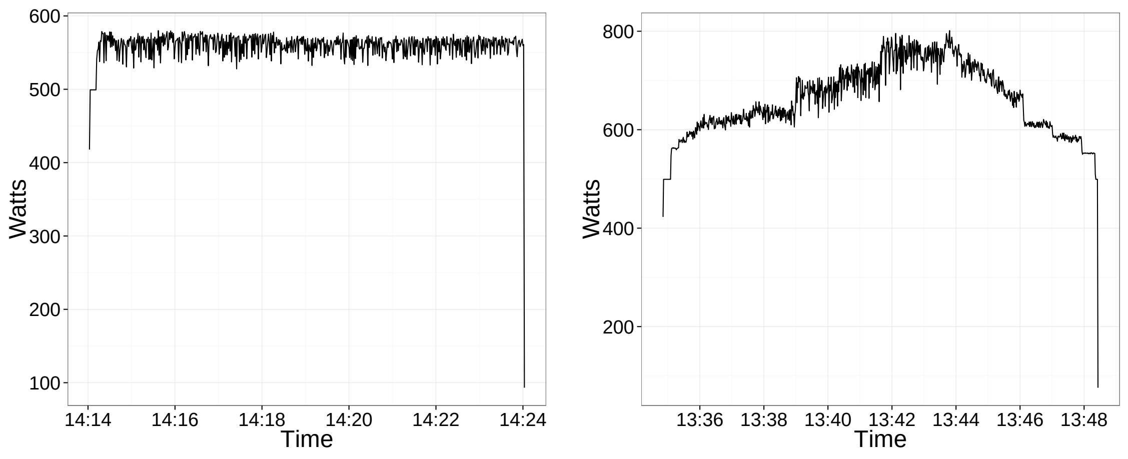
The scaling behavior for tenant #1 is described separately for WK and ST services. Fig. 6a shows the average execution time for the WK VMs calculated over a moving time window of 40 seconds, correlated with the concurrent workload presented in Fig. 5a. Fig. 6b shows the SLA ratio between the average execution time and the SLA threshold of 1000ms. As the SLA ratio approached the SLA scaling threshold (0.9 for scale-out, respectively 0.6 for scale-in), the SSM algorithm varied accordingly the number of VMs, as shown in Fig. 6c.

Similarly, the ST service was scaled based on the average execution time shown in Fig. 7a. The SLA scaling ratio for ST service is shown in Fig. 7b, while the actual number of ST VMs is displayed in Fig. 7c. The maximum number of ST VMs varied from 1 to 3 and then back to 1. It is important to note that



(a) (b)

Fig. 8. Simulation 2 (a) Hosts CPU Utilization. (b) Hosts Power Consumption.



(a) (b)

Fig. 9. Total Datacenter Power Consumption. (a) Simulation 1. (b) Simulation 2

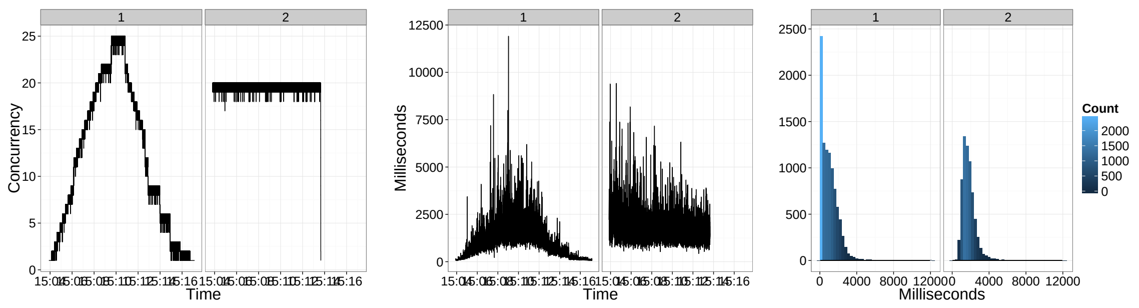
the system did not oscillate as the SLA scaling ratio approached the scaling threshold, because of the scaling speed limitation mechanism described in Section 3.2.

The effect of scaling VMs on the average CPU utilization of hosts can be observed in Fig. 8a, while the energy consumption per host can be observed in Fig. 8b. The total datacenter's power consumption can be visualized in Fig. 9b and had a value of 538 KJ. As the VMs' utilization increases the effect on datacenter's power consumption is an increase with approx. 20%, as the idle hosts still contribute significantly to the total power consumption.

4.3 Simulation 3

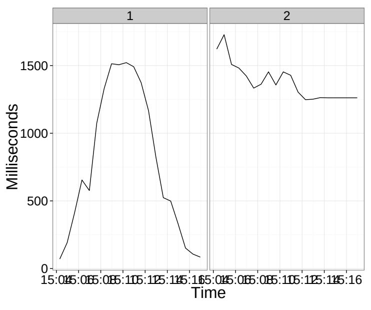
The third simulation consisted of two tenants, each with SLA scaling enabled. The SLA scaling thresholds were the same for both tenants, 1000ms for the WK services, respectively 400ms for ST services.

The workload executed by the first tenant was varying from 1 to 20 and back to 1 concurrent transactions, while the workload of the second tenant was constant at 20 concurrent transactions as shown in Fig. 10a. The execution time per request of each tenant at the CS service is displayed in Fig. 10b, and

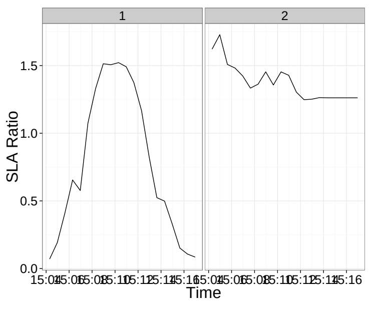


(a) (b) (c)

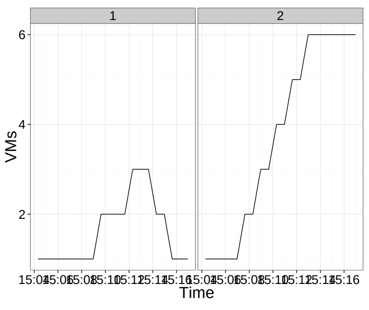
Fig. 10. Simulation 3 - CS service (a) Concurrent Load (b) Response Time (c) Histogram of Response Times



(a)



(b)



(c)

Fig. 11. Simulation 3 - (a) WK average execution (b) WK SLA Scaling Ratio (c) Number of VK VMs

had an average value of 1017.4 ms for tenant 1, and 1802.2 ms for tenant 2 respectively. The measured average values are consistent with the ones obtained in simulations 1 and 2. The histogram of tenants' execution times measured at the CS service is displayed in Fig. 10c.

The WK's average execution time per tenant is displayed in Fig. 11a, while the SLA ratios are displayed in Fig. 11b, and the number of WK's VMs is shown in Fig. 11c. The simulated values are consistent with the ones produced in the previous two simulations.

5. CONCLUSIONS

We have shown how CloudSim can be used as a simulation platform for testing SLA-based infrastructure scaling policies using application performance traces recorded from a small-scale cloud deployment. We described, implemented and validated a time-shared parallel cloudlet scheduler for CloudSim, which we used for building and evaluating a SLA scaling manager for VMs, by running three simulations of varying workloads in a multi-tenant cloud environment.

We have also proposed and validated a CloudSim model for translating application-level performance profiling information to VM-level CloudSim scheduler resource utilization level. We have also identified some possible optimization points in cloud infrastructure management, regarding energy consumption of idle servers. We have shown that SLA guarantees can be used for VM scaling purposes when it is possible to convert them to SLA ratios.

As future work we consider evaluating more complex scaling algorithms by using prediction of both the workload and the SLA usage ratios.

REFERENCES

A-F Antonescu and T Braun. 2014a. Improving Management of Distributed Services Using Correlations and Predictions in SLA-Driven Cloud Computing Systems. In *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, Poland, Krakow.

A-F Antonescu and T Braun. 2014b. Modeling and Simulation of Concurrent Workload Processing in Cloud-Distributed Enterprise Information Systems. In *Proc. ACM SIGCOMM Workshop on Distributed Cloud Computing (DCC 2014)*.

A-F Antonescu, A-M Oprescu, and others. 2013. Dynamic Optimization of SLA-Based Services Scaling Rules. In *Proc. 5th IEEE Internetional Conference on Cloud Computing Technology and Science (CloudCom)*.

A-F Antonescu, P Robinson, and T Braun. 2012. Dynamic Topology Orchestration for Distributed Cloud-Based Applications. In *Proc. 2nd IEEE Symposium on Network Cloud Computing and Applications (NCCA)*.

A-F Antonescu, P Robinson, and T Braun. 2013. Dynamic SLA Management with Forecasting using Multi-Objective Optimizations. In *Proc. 13th IFIP/IEEE Symposium on Integrated Network Management (IM)*.

Rajkumar Buyya and others. 2009. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *Int. Conf. on High Performance Computing & Simulation, 2009. HPCS'09*. IEEE.

Massimo Re Ferre. 2004. Vmware ESX server: scale up or scale out. *IBM Redpaper* (2004).

GRIDS Lab. 2014. Cloud Simulator cloudsim. http://code.google.com/p/cloudsim. (2014).

Alexis Leon. 2008. *Enterprise resource planning*. Tata McGraw-Hill Education.

Wang Long and others. 2013. Using CloudSim to Model and Simulate Cloud Computing Environment. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*.

A. Medina and others. 2001. BRITE: an approach to universal topology generation. *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on* (2001).

Peter Mell and Timothy Grance. 2011. The NIST definition of cloud computing. *NIST special publication* (2011).

Amandeep Sandhu and others. 2013. Modeling Local Broker Policy Based on Workload Profile in Network Cloud. *International Journal* (2013).

Dan Woods. 2003. *Enterprise Services: Architecture*. O'Reilly Media, Inc.