TEM Journal. Volume 5, Issue 3, Pages 294-300, ISSN 2217-8309, DOI: 10.18421/TEM53-07, August 2016.

Efficient Dynamic Capacity Management Approach for Guarantee SLAs in Clouds

Mahfoudh Alasaly [1], Ahmed Al-Sanad [1]

[1] *Information System Department, King Saud University, Saudi Arabia*

Abstract **- Recently, Cloud computing has created new dimensions for computing. The process of providing service – enabled connection is considered to be the most important function of clouds. This paper which we have carefully thought about, takes into consideration dynamic capacity planning for cloud systems. The objective is to dynamically accommodate computing capacity in order to meet Service Level Agreements (SLAs) and, in addition, to minimize the total costs incurred in the cloud services. In this paper, we propose an improvement of a capacity planning scheme to make sure that perfect performance targets of SLAs are always met. The scheme employs threshold-based techniques to guarantee that the SLAs are met and the run-up costs are created to operate on cloud environments and thus has to deal with these environments specific obstacles.**

Keywords **- Cloud computing, Service Level Agreement CPU Utilization, Response Time, Load balancer, EC2, RBE.**

1. Introduction

Cloud computing is considered as one of the most development techniques in the field of science and engineering in the past few years. It plays the key role in the development of science and engineering that are directly associated in our daily life. Cloud computing is utilized to growing the participation of computing resources and minimize the cost of e-business services. It is done on pay-as-you-go such as water or electricity. Nowadays, there are more companies such as Google, Amazon, Microsoft, and IBM who are offering clouds for serving different services for business, research and teaching [14]. Cloud computing has brought about new aspects for computing. Research course which has been carried out lately raises the need for the usefulness of service – enabled businesses and hardware constituents. The service – oriented architecture can be best practiced under cloud computing environment. There are three service patterns in cloud [1]: Infrastructure – as – a – Service (IaaS), Platform –as- a- Service (PaaS) and Software –as-a-Service (SaaS). Using IaaS services clients will be able to utilize infrastructure in accordance with their needs for a specific time and they are changed just for what they utilize or the amount of what they utilize. Using PaaS, clients can improve, examine, host, deploy and maintain their applications especially when they utilize service in one incorporated environment. SaaS enables the cloud utilizers to lease a consummate software solution that is controlled and hosted by the cloud provider. Amazon EC2 and VCL are examples of IaaS service providers. Microsoft Azure and Google APP Engine are examples of PaaS service providers. Oracle and IBM provide SaaS services for database and other applications. For instance, Enterprise Resource planning (ERP) or Customer Relationship Management (CRM) applications. Cloud deployment links up numerous noteworthy issues [2], [3] such as safety, upgrade control, bulk cost, backup hedge, technical function's customization, Quality of Service (QoS) and solution incorporation. In this study, our utmost interest is in QoS and cost reduction through capacity planning tactics.

In addition, to establish contracts with a cloud service provider, QoS goals are usually expressed as Service Level Agreement (SLA). SLAs have a key function in capacity planning schemes in order to manage the resources with restricted cost. For example, the deployment of any application on an IaaS cloud can significantly depress its price by reasonably using the essential capacity in terms of size and type that meets the SLAs. The method is this

DOI: 10.18421/TEM53-07
https://dx.doi.org/10.18421/TEM53-07

Corresponding author: Mahfoudh Alasaly, Information System Department, King Saud University, Saudi Arabia
Email: Alasaly@ksu.edu.sa

new capacity management scheme. It utilizes threshold-based techniques which target meeting the SLAs while reducing the run-up costs by adding or deleting instances depending on the monitored performance metrics. This method is founded on a customer technique where the laptop represents the customer and the Amazon Web Service (AWS) represents the server. The link between the customer and the AWS is set up through a web service. AWS monitoring tool monitors the CPU utilization and response time in minute intervals. Depending on these readings, the tool adds or takes away instances. The methods adopted in the prior work depend on anticipating future workloads, whereas this method does not demand performance patterns.

The main contribution of this paper is to recognize an efficient capacity management scheme to improve performance (in terms of response time, service quality, etc.) of multi-tier web applications as negotiated in the SOLAs of the cloud platform. Such process should supply solutions to determine how many instances to use while decreasing the afford cost. The rest of this paper is organized as follows. Section 2 covers the related work. Section 3 explains the approach and methodology including the description of the proposed algorithms. Section 4 shows the experimental results. The conclusion is presented in Section 5.

2. Related Work

Cloud computing is the new platform of computing that allows users to share resources. Although there are some important issues that should be addressed when moved to cloud computing. One of these important issues is the Quality of Service (QoS) where customers are choosing to deploy their applications on a cloud and are interested in achieving (QoS) goals. To establish contracts with a cloud service provider, QoS goals are usually expressed as Service Level Agreement (SLAs). SLAs can play a critical role in designing resource management and capacity planning schemes while reducing the costs of running cloud applications. For example, a customer deploying an application on an IaaS cloud can significantly reduce its economical cost by wisely using the necessary capacity in terms of size and type that meet the SLAs. Orthogonally, a cloud service provider providing SaaS can significantly reduce its economical cost and energy footprint by providing the necessary resources that meet the SLAs. Thus, designing effective schemes for capacity planning and cloud resource management that ensure meeting the SLAs while taking economical and energy costs into account has become a very hot topic [4-7], [21]. Several research studies have addressed the capacity-planning problem for cloud computing systems. Herein, we focus on such work, which proposes policies that dynamically scale resources up or down (e.g., computing, storage, and networking) in compliance with the SLAs. The resources can be heterogeneous in terms of performance and economical costs. Authors developed an algorithm to control the scaling process depending on threshold value called Scaling Indicator, which is the number of active sessions or logon sessions in each web application. Likewise, in [8] the authors also used the threshold value in a proposed algorithm to detect the bottleneck to guarantee the maximum average of the response time. Response time is the time for a serving demand to be pleased. This is the time it takes for a serving request to be carried out on the service provider's multiple resource sites [8]. To detect the bottleneck there are two cases, i.e. the static content response time saturation and dynamic content response time saturation. If the static content response time is above the threshold then the web tier will be scaled. If dynamic content response time is above the threshold then it acquires the CPU utilization of the Web server tier. If the CPU utilization of any instance in the Web server tier reaches the overload limit, the Web server tier will be scaled up; if not the database tier will be scaled up. The authors in [9] have conducted a comprehensive review on work that talks about dynamically scaling applications in the cloud. They also introduced the available mechanism for the scaling approaches as shown in Figure 1. The authors in [8-10] proposed a threshold-based policy, where lower and upper thresholds on performance measurements e.g., CPU utilization, are defined. When these thresholds are violated, the policy responds by allocating or de-allocating resources. However, these thresholds need to be set based on performance targets as determined in the SLAs and this can be non-trivial. Another policy has been suggested in [11], which uses reinforcement learning. This policy captures the online performance model of an application running on a cloud without any a priori knowledge. However, tuning reinforcement learning can be impractical. Another policy proposed by [12] implements an integral control technique called proportional thresholding. The policy focuses on elastic control of the storage tier in a multi-tier application running on a public cloud. The policy uses CPU utilization on the storage nodes as the sensor feedback metric and assumes homogeneous nodes. These policies assume homogeneous capacity and do not take economical costs into account.

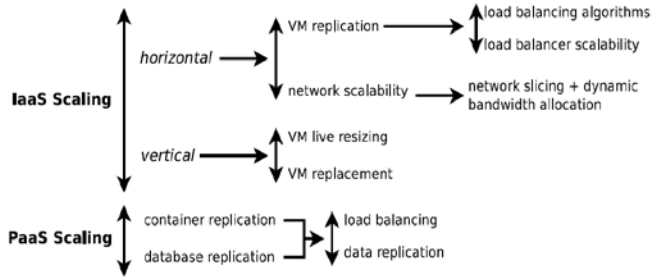

Figure 1: Summary of the available mechanisms for holistic application scalability.

In [13] the authors offered a new method that predicts borders on the response time performance of web APIs published by applications that are hosted in a PaaS cloud. The goal of authors is to grant a PaaS administrator to locate what response time service level agreement (SLA) ability to be achieved by all web API process published by the applications hosted in the PaaS. SLAs typically assign minimum level of service and a probability (commonly large) which the minimum level of service will preserve. SLAs typically assign minimum level of service and a probability (usually large) which the minimum level of service will be preserve.

The author in [14] believed how to assign adequate computing resources however not to over-save these resources to process and analysis auditing logs to ensure the guarantee of an SLA, indicate to the SLA-based on resource allocation trouble for high-performance cloud auditing. Also, the authors presented a study of the SLA-based on methods for resource administration in high-performance cloud audit.

The work in [15] is devoted to bilateral SLA negotiation of PaaS services for hosting multi-tier Web applications in a scenario where the number of users is variable and the workload is not stationary but, typically, exhibits peaks and dips with daily, weekly or seasonal cycles.

The authors in [16] the proposed polices that are able to self-adaptive consideration when the workload variation. In [17] the authors offered DeSVi—the novel architecture to monitor and detect SLA infringements in Cloud computing infrastructures. The authors in [18] investigated how those problems could grow and collect the basic requirements for a service architecture that is capable to overcome these requests. They addressed these basic requirements for the development of functional and architectural for the SLA-based Service Virtualization and on-demand resource provision so that they are well-designed, and offered a novel way called Service-level agreement-based on Service Virtualization (SSV). Fig. 2 illustrates the proposed system, general architecture called SLA-based on Service Virtualization (SSV).

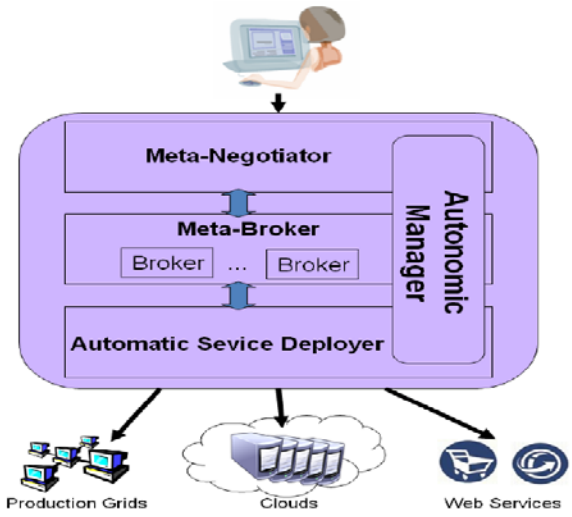

Figure 2: SSV architecture.

The authors in [19] offered two new algorithms for VM-scaling concentrated on dEIS systems, which are capable to be utilized by infrastructure management of cloud systems to optimally discover the most suitable scaling conditions use performance-models of distributed applications derived of fixed-workload benchmarks. In [22] the authors tested an MVA-based method to predict the average of response time that is used for a web application hosted on a cloud computing platform. An analytical model proposed by authors in [23] depend on queuing network to assessment of the aggregated QoS metrics of multi-tier applications in a virtualized datacenter.

3. Methodology

In this paper, we extend our previous study in [20] which discussed the conducted measurements for monitor CPU Utilization in various workloads through different number of clients and comparison of the results. In this paper, we applied the methodology that was used in [20] to deal with SLAs in a cloud, capacity management scheme. We utilized threshold-based mechanism that satisfies the SLAs with decrease costs incurred using addition or deletion instances that rely on the observed performance metrics. This approach is relying on client server mechanism where the laptop is representing the client and the amazon web service (AWS) is representing the server. The communication between the client and AWS is over a web service. Control tool monitors the CPU Utilization and response time readability each minute. Relying on these readability, the tool addition or deletion of the instances depends on the workload that is generated by TPC-W benchmark. TPC-W is a web benchmark that is vastly utilized to

assessment e-commerce systems. The method followed in the previous work relies on predicting future workloads while this method does not demand performance paradigms. In this status study, client-server architecture is utilized in which the client deploys this tool and the Amazon cloud manager appears the server, the communication between client and server is over web service. The web application works in Amazon cloud. The performance of any specific system is selected by carefully analyzing certain performance measures such as response time and resource utilization. The response time (latency) is the time period elapsed between the times at which a package is submitted to the system for processing until the answer begins to appear at the user's terminal. Utilization is the percentage of time the device is being used, through a specific time interval. Figure 3. shows the processes that should be followed by this approach.

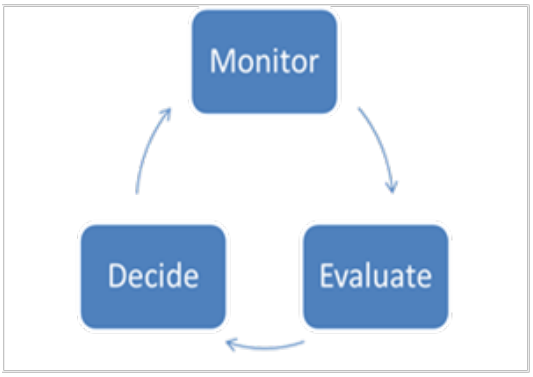

Figure 3: Capacity management scheme processes.

3.1 Monitoring

The tool will constantly monitor the system. It monitors the readings for CPU utilization and Response time (latency) every minute (this is the maximum possible monitoring frequency in Amazon EC2). Depending the reading of CPU Utilization and Response time, the tool automatically decides adding or removing instance.

3.2 Evaluate

The tool assesses the situation depending on the readings of the CPU Utilization and response time.

3.3 Decide

In this stage the tool decides how many instances should be used. In this paper, using threshold-based schemes depending on monitoring the CPU utilization or response time, the tool dynamically adds or removes Tomcat instances to the load balancer depending on monitored CPU utilization and response time.

In this approach, client-server architecture is used in which the client deploys the tool and the Amazon cloud manager represents the server, the connection between client and server is through web service. The web application runs in Amazon loud instances Figure 4. shows the system architecture.

4. Experiments

In experiments, Amazon EC2 framework was used to check the performance of various arrangement that deploy TPC-W. Tomcat, and MySQL are utilized as the application, web, and database servers, sequentially. A load balancer is utilized to partition requests from an Apache instance to the Tomcat instances.

In this stage more experiments were carried out to measure the Response time for different numbers of clients (50, 200) and compare the results between them. The maximum number of clients is 200 because when the clients are more than 200, the error occurs. In the same context more experiments were carried out for a minimum number of clients, which is 50. TPC_W was used to run benchmark to generate the workload for different number of clients.

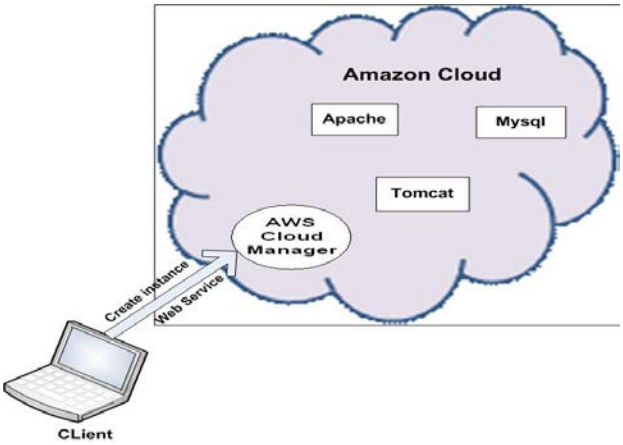

Figure 4: System's Architecture.

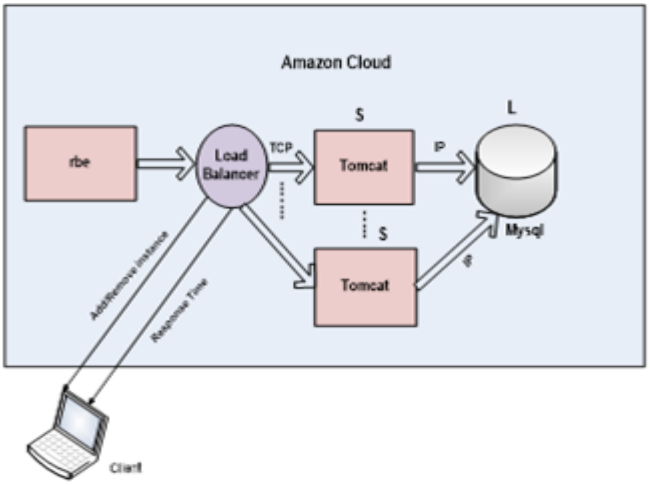

Figure 5: TPC-W deployment on Amazon Cloud Using Amazon Load balancer.

4.1 Monitoring Response Time (latency) for different instances

Fig. 5. shows TPC-W web application deployed on two Instances: Tomcat and MySQL(S, L) and load balancer used instead of Apache. We run TPC_W using different numbers of clients (N), the tools monitor the response time (latency) of the three instances.

Table 1: Load balancer Latency measurements for N=50.

Time (min)	Latency for Load balancer (msec)	Decision
0	0	Before running TPC-W
1	80.87154739	Use One Tomcat instance
2	80.89351794	NO Action
3	80.06155359	NO Action
4	80.21935349	NO Action
5	77.58310983	NO Action
6	84.33817575	NO Action
7	80.02068753	NO Action
8	78.99836054	NO Action
9	77.70095751	NO Action
10	80.83904144	NO Action

4.2 Response Time Scheme Algorithm

The algorithm below computes the max. response time threshold twice for every minute and according to that add or remove instances.

```
Input: Max-Latency. // maximum value of latency threshold.
       Min-latency. // minimum value of threshold.
       N. // Number of users.
Output: Tomcats added or removed or No Action.

Instances=1;
Count=1;
Timer=0;
While (Timer<10) then
 Begin
    L= Current latency for load balancer;
   If (L>Max-latency) then
     Begin
        If (count<2) then

     Count=Count +1;
  Else Begin
     Add Tomcat instance to the load balancer
      Count=0;
   End {if count}
Else
begin
  If (Instances>=2) then
    begin
       Remove a Tomcat instance from the list of
      action instances.
      Instances=Instances-1;
    End
End
Timer=Timer +1;
End;{While}
```

Initially, the load balancer latency is zero because there is no workload executing yet. When the TPC-W is running and workload generated by remote browser emulator (RBE) with number of users N= 50 we see not any change for all experiments because workload is low and not reached to threshold (threshold between 200-300) so the load balancer does not need any other instances and remaining have only one Tomcat instance (See Table 1).

In contrast, when the number of users N=200, we see the workload generated by RBE and reached the peak with reading value 472 exceeded threshold in this case, the monitoring tool automatically adds another instance to load balancer to become two instances in on state. On the other hand

when the reading value reached minimum 148, we see on the monitoring tool no action and the load balancer still has one instance (See Table 2. and Figure 6.). It is necessary that the load balancer contains at least one Tomcat instance.

Table 2: Load balancer Latency measurements for N=200.

Time (Min)	Latency for Load balancer (msec)	Decision
0	0	Before running TPC-W
1	343.3362922	Use two Tomcat instances
2	148.7699243	NO action
3	148.7699243	NO action
4	159.8360217	Remove one Tomcat
5	447.717502	Use two Tomcat instances
6	468.1052258	NO action
7	158.6571399	NO action
8	158.6571399	Remove one Tomcat
9	138.9171064	NO action
10	472.89342	Use two Tomcat instances

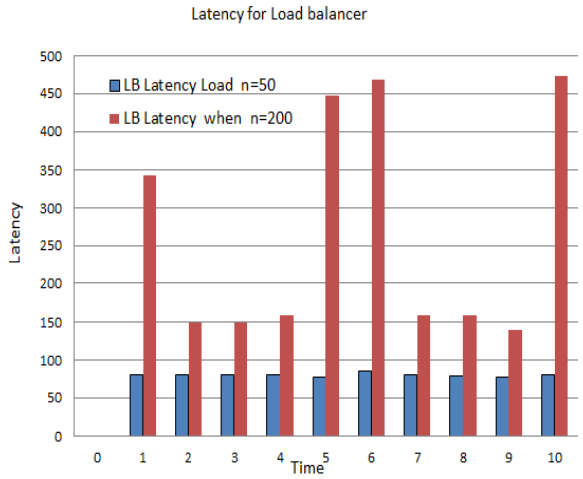

Figure 6. Load balancer latency measurement (N=50 and 200)

5. Conclusions

Dynamic capacity management method is proposed to guarantee the SLAs for customers with reducing the total cost in cloud environments. Our method increased and decreased the number of instances based on the current latency workload to keep the total cost as required all the time. We have built and verified a capacity management schemes for web applications on Amazon EC2 cloud system. The web application environment is installed on Amazon EC2. Furthermore, we have built a tool to monitor Amazon instances online and to manage the instances. The experimental results demonstrate that the proposed method is efficient and satisfies the customer requirements with the lowest cost where we benchmarked the workload for 50, 100, 150, and 200 users and we found that on a large number of users, 200, there is obvious difference in response times so that the number of instances are changed from time to time according to the corresponding workload.

References

[1] Mell, P., & Grance, T. (2010). The NIST definition of cloud computing.*Communications of the ACM*, *53*(6), 50.

[2] Shehab, M., Sharp, L., et. al. 2004. Enterprise resource planning, Business Process Management Journal, Vol. 10 No. 4, 2004, pp. 359-386, Emerald Group Publishing Limited 1463-7154. DOI 10.1108/14637150410548056.

[3] Themistocleous, M., Irani, Z. and O'Keefe, R. 2001. ERP and application integration: exploratory survey, Business Process Management Journal, Vol. 7 No. 3, pp. 195-204.

[4] Chen, J., Wang, C., Zhou, B. B., Sun, L., Lee, Y. C., & Zomaya, A. Y. (2011, June). Tradeoffs between profit and customer satisfaction for service provisioning in the cloud. In *Proceedings of the 20th international symposium on High performance distributed computing* (pp. 229-238). ACM.

[5] Waheed Iqbal, Matthew N. Dailey, and David Carrera. SLA-driven dynamic resource management for multi-tier web applications in a cloud. In Proceedings of the IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 832-837, 2010.

[6] Hui Li, Giuliano Casale, and Tariq Ellahi. SLA-driven planning and optimization of enterprise applications. In Proceedings of the WOSP/SIPEW International Conference on Performance engineering, pages 117-128, 2010.

[7] Qian Zhu and GaganAgrawal.Resource provisioning with budget constraints for adaptive applications in cloud environments.In Proceedings of the ACM International Symposium on High Performance Distributed Computing, pages 304-307, 2010.

[8] Waheed Iqbal, Matthew N. Dailey, and David Carrera. SLA-driven dynamic resource management for multi-tier web applications in a cloud. In Proceedings of the IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 832-837, 2010.

[9] Luis M. Vaquero, Luis Rodero-Merino, and RajkumarBuyya. Dynamically scaling applications in the cloud. Computer Communication Review, 41:45-52, 2011

[10] Dutreilh, X., Rivierre, N., Moreau, A., Malenfant, J., Truck, I.: From data center resource allocation to control theory and back. In: Proceedings of the Conference on Cloud Computing. pp. 410- 417 (2010).

[11] Xavier Dutreilh, Nicolas Rivierre, Aur_elien Moreau, Jacques Malenfant, and Isis Truck. From data center resource allocation to control theory and back. In Proceedings of the IEEE International Conference on Cloud Computing, pages 410-417, 2010.

[12] Harold C. Lim, ShivnathBabu, and Je_rey S. Chase. Automated control for elastic storage. In Proceedings of the IEEE/ACM International Conference on Autonomic computing, pages 1-10, 2010.

[13] Jayathilaka, Hiranya, Chandra Krintz, and Rich Wolski. "Response time service level agreements for cloud- hosted web applications." Proceedings of the Sixth ACM Symposium on Cloud Computing. ACM, 2015.

[14] Xiong, Kaiqi, and Xiao Chen. "Ensuring Cloud Service Guarantees Via Service Level Agreement (SLA)-based Resource Allocation." Distributed Computing Systems Workshops (ICDCSW), 2015 IEEE 35th International Conference on. IEEE, 2015.

[15] Ranaldo, Nadia, and Eugenio Zimeo. "Capacity-driven utility model for service level agreement negotiation of cloud services." *Future Generation Computer Systems* 55 (2016): 186-199.

[16] Casalicchio, Emiliano, and Luca Silvestri. "Mechanisms for SLA provisioning in cloud-based service providers." *Computer Networks* 57.3 (2013): 795-810.

[17] Emeakaroha, Vincent C., et al. "Towards autonomic detection of SLA violations in Cloud infrastructures." *Future Generation Computer Systems* 28.7 (2012): 1017-1029.

[18] Kertész, Attila, Gabor Kecskemeti, and Ivona Brandic. "An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous Cloud environments." *Future Generation Computer Systems* 32 (2014): 54-68.

[19] Antonescu, Alexandru-Florian, and Torsten Braun. "Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications." *Future Generation Computer Systems* 54 (2016): 260-273.

[20] Alasaly, Mahfoudh, Hassan Mathkour, and Issam Al-Azzoni. "Dynamic Capacity Planning with Comprehensive Formation of SLAs in Clouds." International Journal of Computer Applications 117.19 (2015).

[21] Sara Bouchenak. Automated control for SLA-aware elastic clouds. In Proceedings of the International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks, pages 27-28, 2010.

[22] Suleiman, Basem, and Srikumar Venugopal. "Modeling performance of elasticity rules for cloud-based applications." *Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International*. IEEE, 2013.

[23] RahimiZadeh, Keyvan, et al. "Performance modeling and analysis of virtualized multi-tier applications under dynamic workloads." Journal of Network and Computer Applications 56 (2015): 166-187.