Dynamic Capacity Planning with Comprehensive Formation of SLAs in Clouds

Mahfoudh Alasaly
Information System Department
King Saud University
Riyadh, Saudi Arabia

Hassan Mathkour
Computer Science Department
King Saud University
Riyadh, Saudi Arabia

Issam Al-Azzoni
Software Engineering Department
King Saud University
Riyadh, Saudi Arabia

ABSTRACT
The provision of service enabled connectivity is the significant art of clouds. The long-held dream of computing as a utility has become reality in the era of cloud computing. Cloud users are now able to run and access their applications from anywhere in the world on demand. The proposed research considers dynamic capacity planning for cloud systems. The aim is to dynamically adapt computing capacity such that Service Level Agreements (SLAs) are continuously met while minimizing the total costs incurred in running the cloud services. However, research in this regard is still at its infancy. Scalability, resource heterogeneity, workload dynamicity, resource sharing and virtualization are the main challenges that need to be overcome to have effective and trustworthy schemes for capacity management, that play a vital role in cloud computing. The work in this approach is based on developing a capacity planning scheme to ensure that high-level performance targets (SLAs) are continuously met. The scheme applies threshold-based techniques to ensure meeting the SLAs while minimizing the total incurred costs. The approach is designed to work on cloud environments and hence must address these environments specific challenges.

Keywords
Cloud computing, CPU Utilization, Response Time, Load balancer, EC2.

1. INTRODUCTION
Cloud computing has created new dimensions for computing. Recent research trend invokes the necessity of service enabled business functions and hardware components. The service oriented architecture can be best practiced under cloud computing environment. Cloud offers three service models [1] like Infrastructure-as-a-Services (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). With IaaS services customers can use infrastructure according to their requirements for particular time and they have to pay only for what they used or how much they used? With the help of PaaS, customers can develop, test, host, deploy and maintain their applications using services in the same integrated environment. Software as a Service (SaaS), allows the clouds users to rent a complete software solution that is managed and hosted by the cloud provider. Amazon EC2 and VCL are example of IaaS service providers. Microsoft Azure and Google App Engine are examples of PaaS service providers. Oracle and IBM provide SaaS services for data base and other applications.

Cloud deployment associates several noteworthy issues [2][3] such as safety, upgrade control, bulky cost, backup hedge, technical function's customization, Quality of Service (QoS) and solution integration. In this study, our ultimate concern is with QoS and cost minimization through capacity planning tactics.

To establish contracts with a cloud service provider, these QoS goals are usually expressed as Service Level Agreement (SLAs). SLAs play vital role in capacity planning polices in order to manage the resources with limited cost. For example, the deployment of any application on an IaaS cloud can significantly reduce its economic cost by wisely using the necessary capacity in terms of size and type that meets the SLAs. The approach in this work to address SLAs in a cloud, proposes a new capacity management scheme. It utilizes threshold based techniques which aim at meeting the SLAs while minimizing the incurred costs by adding or removing instances depending on the monitored performance metrics. This approach is based on a client server technique where the laptop represents the client and the Amazon Web Service (AWS) represents the server. The connection between the client and AWS is established through a web service. AWS monitoring tool monitors the CPU utilization and response time in minute interval. Depending on these readings, the tool adds or removes instances. The approaches adopted in the prior work depend on predicting future workloads whereas, this approach does not require performance models. The rest of the paper is organized as follows. Section 2 covers the related work. Section 3 explain the approach and methodology that will be used including the description of the proposed algorithms. Section 4 includes the experimental results. The conclusion presents in Section 6.

2. RELATED WORK
Several research studies have addressed the capacity-planning problem for cloud computing systems. Herein, the focus was 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. Several policies exploit queuing network models to predict future performance under a given capacity configuration. One of the first contributions for capacity allocation and workload redirect has been proposed in [4]. The authors have proposed two algorithms for capacity allocation and load redirect of the requests. In the first algorithm they tried to determine the number of the running VM instances on a specific site in order to minimize the cost of these instances. They use optimization model to predict these numbers. The second algorithm concerns on determining the execution rate of incoming requests. If there is a problem in one site (e.g. server is down), requests will be redirected toward other sites taking in account that requests can be redirected only once and if not, this will affect the overall response time. Not far from what is aforementioned, in [5] the authors also tried to develop an optimization model using queuing model network to minimize the number of running VM with respect to the customer response time. unlike as we mentioned before, the authors developed an optimization model by using two queuing model M/M/c for the first tier and multiple queue M/M/1 for the other tiers based on Poisson distribution as shown in Figure. 1.

The authors in [6] discuss the challenges that encounter the auto scaling issues such as workload forecasting, identify resource requirement for incoming load, and resource allocation regarding cost factors.

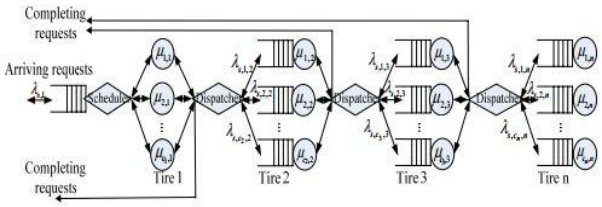

Fig 1. Open queuing model for virtualized multi-tier application

They develop a look-ahead RTA algorithm based on model predictive control which forecasts future workload with respect to the limitation and adjusts resources allocated to user's ahead-of-time. As well as, in [7] the authors exploit queuing system to determine the capacity of serving system. Requests coming into a tier are modeled as requests visiting a queue modeled as a G/G/1 queuing system. Each tier receives partially processed requests from the previous tier and feeds these requests into the next tier after local processing as shown in Figure. 2.

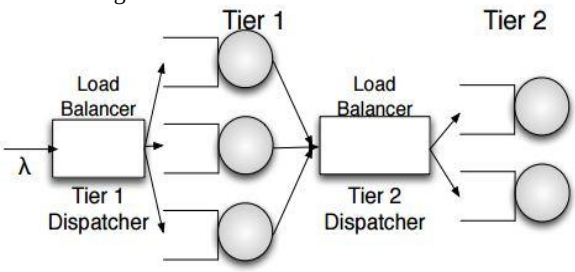

Fig 2. Multi-tier application model

Subsequently, using optimization methods, a capacity configuration is chosen such that the SLAs are met at minimal capacity and cost. These policies assume homogeneous capacity and the majority do not consider the economic costs of running the cloud application.

Recently Al-Azzoni and Kondo in their paper [8] have performed the Mean Value Analysis (MVA) to predict performance of multi-tier web applications running on multiple heterogeneous virtual machines over a public cloud (Amazon EC2). Their approach was shown to produce good performance predictions. The approach does not require intensive instrumentation and uses readily available server logs and system monitoring tools. In such environments, the different virtual machines running on the public cloud can potentially compete for resources, and hence it is important to be able to measure resource demands in an online fashion. Other policies apply concepts from control theory. These policies respond in a reactive way rather than other commonly used predictive and proactive approaches. The authors in [9] designed a novel architecture, which enables the resources of web application to grow or shrink on a cloud system with respect to the incoming requests (on-demand scaling) as depicted.

3. METHODOLOGY

The approach to address SLAs in a cloud, proposed a new capacity management scheme. It utilizes threshold based techniques which aim at meeting the SLAs while minimizing the incurred costs by adding or removing instances depending on the monitored performance metrics. This approach is based on client server technique where the laptop represent the client and the amazon web service (AWS) represents the server. The connection between the client and AWS is through a web service. The monitoring tool in client monitors the CPU Utilization and response readings every minute. Depending on these readings, the tool adds or removes the instances. The approaches adopted in the prior work depends on predicting future workloads whereas, this approach does not require performance models. In this case study, client-server architecture is used in which the client deploys this 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 cloud. The performance of any particular system is determined by carefully analyzing certain performance measures like response time and resource utilization. The response time (latency) is the time interval elapsed between the times at which a transaction 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, during a given time interval.

The main objective of this research work is to find an effective capacity management scheme to improve performance (in terms of response time, service quality, etc.) of multi-tier web applications as negotiated in the SLAs of the cloud platform. Such approach should provide solutions to decide how many instances to use while reducing the incurred cost. Figure.3 shows the processes that should be followed by this approach.

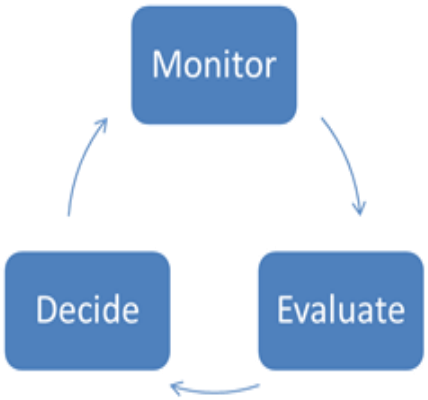

Fig 3. Capacity management scheme processes

1.1 Monitoring

The tool will develop constantly monitors the system. It monitors the readings for CPU utilization and 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.

1.2 Evaluate

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

1.3 Decide

In this stage the tool decides how many instances should be used. In this project 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 paper, 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 below shows the system deployment.

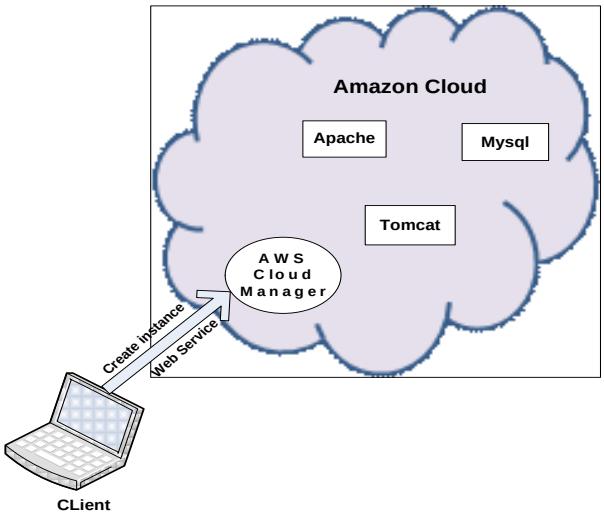

Fig 4. System Deployment

4. EXPERIMENTS

In this stage more experiments were carried out to measure the CPU Utilization and Response time for different numbers of clients (50,100, 150 and 200) and compared the results between them. The maximum number of clients is 200 because when the clients are more than 200, the error is occurred. In the same context also 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 numbers of clients.

4.1. Monitoring CPU Utilization for different instances

Figure. 5 shows TPC-W web application deployed on three instances: Apache, Tomcat and MySQL(S, S, and L). When TPC_W was run using different numbers of clients (N), the tool monitors the CPU-utilization of the three instances.

4.2. CPU UTILIZATION THRESHOLD-BASED SCHEME

The algorithm below compute the max CPU utilization threshold twice for every minute According to that add or remove instances.

Let

Max- CPU: maximum CPU-utilization threshold

Min -CPU: minimum CPU utilization threshold

```
Count1=0;
        Count2 0;
        Every 1 minute:
        - Obtain monitoring data.
        - Compute average CPU utilization of the
active Tomcat instances (avgCPU)
        If (avgCPU >Max-CPU)    {
If (count=1)   {
      Add Tomcat instance to the list of active
Tomcat instances
  Count1=0;
} else count1=1;
        } else if (avgCPU<Min-CPU) {
If (count2=1) {
     Remove a tomcat from the list of active
Tomcat instances
Count2 =0;
} else count2 =1;
} else {
Count1=0;
Count2=0;
```

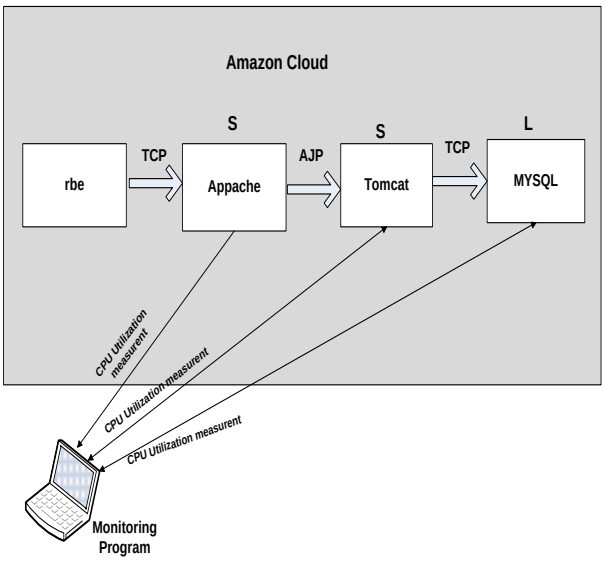

Fig 5. TPC_W deployment on Amazon cloud

Initially, in Figure. 6 the CPU utilization for the Tomcat instance is zero because there is no workload yet. When the TPC-W is running, Figure. 6 shows the CPU Utilization measurement for the cases of N =50 clients and N=150 clients.

The observation was when the time reaches 10, the CPU utilization is zero because requests for clients were finished Also, carried out more experiments using different numbers of clients such as (50,100,150,200) clients.

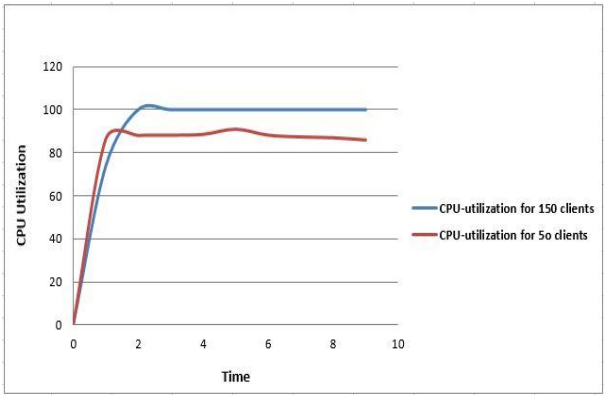

Fig 6. CPU utilization measurement for Tomcat instance under different numbers of clients

When the numbers of clients exceeded 100, was observed slight change in CPU utilization. Slight change was noticed in CPU utilization shown in Figure. 7.

Fig 7. Cloud Watch monitoring using different numbers of clients

4.3. Using CPU Utilization Threshold-based Scheme

In this case, Amazon load balancer was used instead of Apache and generate workload by using remote browser emulator (rbe) instance for different numbers of clients. CPU Utilization threshold between 70% and 90% was set. The monitoring program in the client allows to monitor the CPU utilization for each Tomcat instance, for example, if the CPU utilization increases more than 90%. In the start, the CPU-utilization for Tomcat1 and Tomcat2 is zero because there is no workload executing yet. When the rbe starts generating the workload, for example using 50 or 150 clients, we see the CPU Utilization increases. If the new reading for Tomcat1 instance becomes greater than 90% twice in a row, the monitoring program in the client dynamically adds another Tomcat instance to the load balancer. When the Tomcat2 instance is on, we see the new reading for the CPU in Tomcat1 decreases twice, in a row the monitoring program automatically adds the second Tomcat to the load balancer, and vice versa if the average CPU utilization for both Tomcat instances decreases to less than to 70%, the monitoring program automatically removes this instance from the load balancer. It is necessary the load balancer contains at least one tomcat instance.

If the average CPU-utilization of the two Tomcat instances decreases to less than 70% twice in a row, the monitoring program removes the Tomcat2 instance. This process continues until finishing the specified experiment time period. Figure. 8 and Figure. 9 show the CPU Utilization changes in Tomcat1 and Tomcat 2 when using 50 and 200 clients.

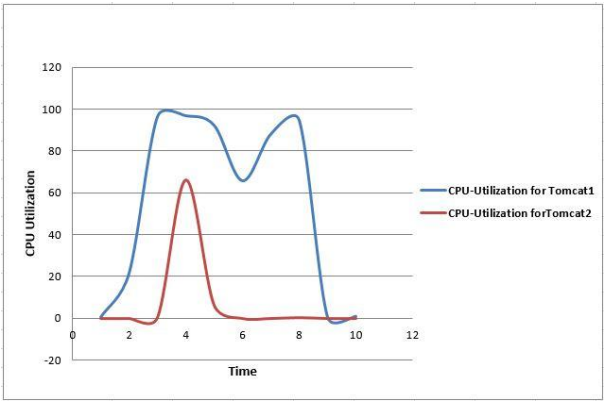

Fig 8. CPU Utilization measurements for N=50

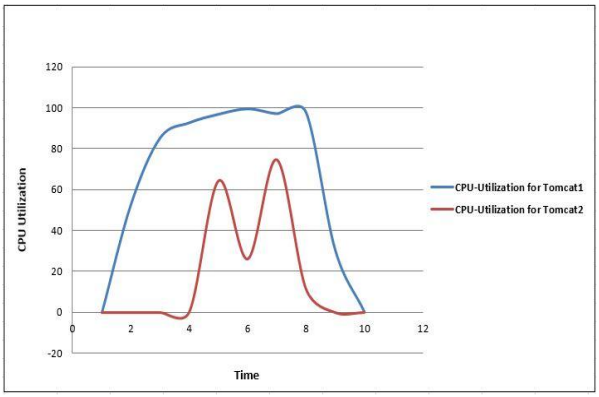

Fig 9. CPU Utilization measurements for N=200

5. CONCLUSIONS

In this paper, the primary focus is to find new methods to help monitor multi-tier web applications in a cloud platform. Threshold-based capacity management scheme was proposed to ensure SLAs are continuously met for multi-tiers web applications while reducing the total incurred cost. In this paper a capacity management schemes was built and verified for web applications on Amazon EC2 cloud system. Also a web application environment was setup in Amazon EC2. Furthermore a tool to monitor Amazon instances online and to manage the instances was built (i.e., dynamic capacity scheme was implemented).

6. REFERENCES

[1] Peter, M and Timothy, G. 2011. The NIST Definition of Cloud Computing.

[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] Danilo, A., Sara, C., and Barbara P.2011. Flexible distributed capacity allocation and load redirect algorithms for cloud systems. In Proceedings of the IEEE

International Conference on Cloud Computing, 2011, 163-170.

[5] Jing, B., Zhiliang, Z., Ruixiong,T., and Qingbo, W.2010. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In Proceedings of the IEEE International Conference on Cloud Computing, 2010, 370-377.

[6] Nilabja, R., Abhishek, D., and Aniruddha, G. 2011. Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting. In Proceedings of the IEEE International Conference on Cloud Computing, 2011, 500-507.

[7] Rahul, S., Upendra S., Emmanuel, C., and Prashant, S. 2010. Autonomic mix-aware provisioning for non-stationary data center workloads. In Proceedings of the IEEE/ACM International Conference on Autonomic computing, 2010, 21-30.

[8] Issam, A., and Derrick, K., 2012. Cost-Aware Performance Modeling of Multi-Tier Web applications in the Cloud. In proceedings of the international conference networked digital technology 2012.

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