Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

DISCRETE EVENT SIMULATION MODEL FOR ANALYSIS OF HORIZONTAL SCALING IN THE CLOUD COMPUTING MODEL

Joseph Idziorek

Iowa State University
2215 Coover Hall
Ames, IA 50011, USA

ABSTRACT

One of the distinguishing characteristics of the cloud model is the ability for the service users to horizontally scale computing resources to match customer demand. Because the cloud model is offered in a pay-as-you-go scheme, it is in the service user's best interest to maximize utilization while still providing a high quality of service to the customer. This paper describes a discrete event simulation model that is used to explore the relationship between the horizontal scaling profile configurations and the functionality of the cloud model. Initial results show that both a state-aware load distribution algorithm and the parameters that dictate the elasticity of the horizontal scaling ability are essential to achieving high rates of utilization. Through modeling and simulation, this paper presents both a framework and initial results to further explore the cloud model.

1 INTRODUCTION

The cloud computing model provides the user with the ability to exploit massively scalable computing environments which are distributed across geographically disperse data centers and are offered in a pay-as-you-go model. The seemingly infinite resources Cloud Service Providers (CSPs) offer supersede the capacity of costly in-house compute clusters or private grid architectures. Potential users now have access to computing power that was once only available to exclusive groups. The benefit of a highly scalable and flexible environment is that customers can implement applications within an infrastructure that is able to scale up resources as demand for the application increases and scale down when demand decreases and resources are no longer needed. This functionality, known as horizontal scaling, alleviates service users (customers of the cloud service) from the burden of up-front investment in capital expenses such as hardware and software licenses or complex usage forecasting.

One of the key characteristics that keeps the concept of cloud computing from being a case of "old wine in a new bottle" is the cloud model's inherent ability to self-provision resources (Voas and Zheng 2009). The responsiveness of the cloud model's ability to scale capacity to meet fluctuating demand is determined by user-specified horizontal scaling parameters. However, the ability afforded to the user does not guarantee that the resulting configuration of cloud-based resources will produce the favorable functionality. The horizontal scaling configurations are an essential step in this process. On one hand, a conservative approach will lend the cloud configuration under-utilized, costing the user unnecessary fees. On the other hand, an over-utilized cloud configuration will diminish the quality of service for all customers as the backend servers become saturated with service requests, potentially resulting in rejected connections. The goal of this study is to investigate the fundamental performance issues within the cloud computing model. The paper focuses on the benefits of using state-aware load distribution algorithms as

well as the effect that horizontal scaling profile configurations have on the trade-off between utilization and quality of service.

The paper is organized as follows. In Section 2, the background of the cloud computing model is presented. Section 3 provides a description of the cloud model by outlining the architecture of the model as well as the essential components necessary for a discrete event simulation model. Section 4 describes the simulation experiments and presents initial results. Finally, in Section 5, a summary is provided as well as direction for future work.

2 BACKGROUND

The information technology (IT) industry is on the cusp of a transformation that is changing the way people interact with computers and access Internet-based resources. The current path that computing management models have been following for the past two decades is likely to diverge as the economic advantages of renting computing resources increase. Furthermore, Internet user demographics show that there is a desire for pervasive access to an omnipresent persona in the Internet. This shift in both economic models and user expectations has brought about the advent of the cloud computing model. Cloud computing as a whole is not a new nor a distinct technology but an aggregation of existing technologies that have matured to a point where massively scalable and dynamic compute environments can be offered as a variety of services with a pay-as-you-go pricing model.

Cloud computing seeks to provide a massively scalable computing environment that can be abstracted and delivered as different levels of service (Mell and Grance 2009). Outsourcing IT services has provided economic advantages for some organizations in comparison to traditional in-house operations. Flexible business commitment duration and pay-as-you-go payment models are very attractive conditions under which to manage computing resources. Users of the cloud are not burdened with the up-front costs of infrastructure investment and forecasting of infrastructure demands. Furthermore, renters of cloud-based services also may benefit from the scalability and seemingly infinite resources made possible by the economies of scale achieved in a multitenant model.

When fundamentally broken down, cloud computing is a specialized distributed computing model. Building upon the desirable characteristics of cluster, grid, utility, and service-oriented computing, cloud computing introduces a compliment of unique features to create a new computing paradigm (Foster, Yong, and Raicu 2008). The technologies comprising much of cloud computing have been around for some time now (e.g. virtualization, broadband, high-density storage, multi-core processors); however, it has not been until recently that these technologies have together all matured to the point where a novel synthesis of these foundational components could be realized.

3 DESCRIPTION OF THE SIMULATION MODEL

In this section, a simulation model and framework are presented for the cloud computing model. The use of modeling and simulation are appropriate in this context due to the lack of open source cloud computing software platforms that support horizontal scaling. Furthermore, the cost and inflexibility of performing research on CSPs' systems is not currently a feasible option. It follows that a simulation model provides a sound alternative to further explore the interactions and performance considerations of the cloud model, particularly when the cloud architecture is subjected to large amounts of service requests.

The simulation model discussed in this paper is represented in Figure 1. By its very nature, the public cloud computing model is Internet-facing, allowing clients to gain access to cloud-based resources from any Internet-accessible device, regardless of the client's location. As a result and similar to other Internet-facing resources, the supporting infrastructure of a cloud computing model must be able to handle dynamic loads created by persistent user demands.

Individual clients or customers initiate connections with the cloud-based resources via a public IP address assigned to the public facing portal of the load balancer. The load balancer accepts incoming requests from clients and appropriately forwards the requests to back-end Web servers (which are given

private IP addresses) depending on the given scheduling algorithm. The advantage of physically and logically separating the public-facing access portal of the load balancer and the back-end resources is that private IPs are not routable from the Internet, which provides a layer of security. Furthermore, the inclusion of a load balancer abstracts the pool of resources in the horizontal scaling group so that it appears to the client as though they are accessing a single server.

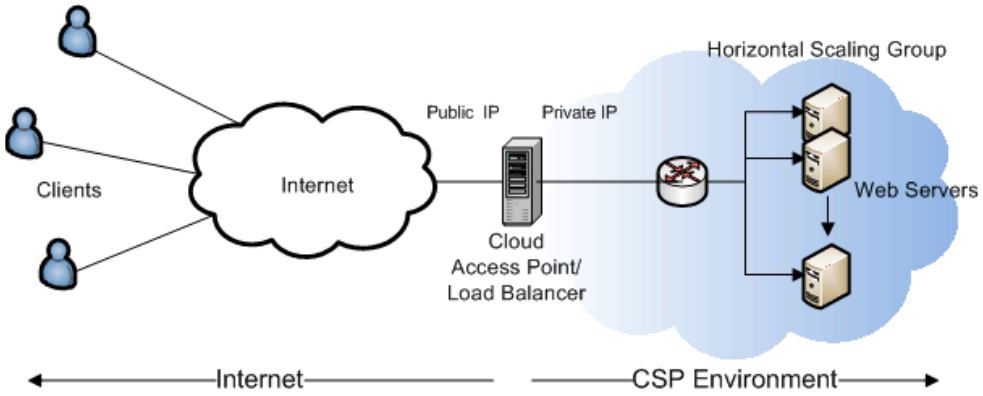

Figure 1: Cloud Computing Network Model

The horizontal scaling group consists of a pool of homogeneous Apache Web servers. Each of these servers possesses the same hardware and software configurations as well as an equal ability to process customer service requests. The number of servers in the available scaling pool as well as the number of active servers that can accept customer requests at any given moment are determined by the configuration settings which are in turn specified by the user.

3.1 Load Balancer

The primary tasks of the load balancer are to efficiently distribute incoming requests and to perform state-checking functionality to monitor the utilization of back-end servers. State monitoring becomes particularly useful for load-balancing algorithms such as least-connection, in which the next service request is routed to the server that is least utilized. Server load-balancing is an indispensible method for providing high-quality services to Internet-facing resources. In addition to the load balancer, the quality of the service, in part, depends on the overall capacity of the supporting system in respect to user demand. Effective scheduling performed by the load balancer seeks to ensure that the support infrastructure is efficiently utilized. While the back-end servers provide reliability through replication, the single load balancer does indeed present a single point of failure. Although this scenario is outside the scope of this paper, previous work has been conducted that seeks to increase the availability of load balancers through replication, active-standby, or active-active scenarios (Friedrich, Krahmer, and Schneidenback 2006; Bourke 2001).

The discrete event simulation model proposed in this paper examines a first-in, first-out (FIFO) queuing model for a cluster of web server instances with the load balancing architecture shown in Figure 2 (Ying-Wen and Yu-Nien 2006). The load balancer, much like the Web servers, handles incoming requests individually and thus can be modeled as a FIFO single-server queue in which customer arrival times are modeled as a Poisson distribution and service times are generated from an exponential distribution (Judge 1999).

In this paper, load distribution policies are examined in respect to the cloud model wherein the aggregation of backend web servers are homogeneous and appear to the user to function as a single, high-performance Web server. In attempt to achieve effective load distribution across available Web servers, three load balancing algorithms are considered:

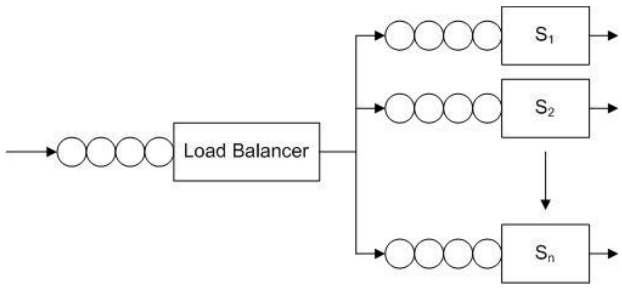

Figure 2: Load Balancer Configuration

Random – Service requests are randomly assigned by the load balancer to Web servers. Each server has an equal probability of receiving a request and the current state of the server is not considered.

Round-Robin – Service request are effectively cycled among the available Web servers. Only the previously routed decision is used to determine the next choice.

Least-Connection – Service requests are directed to the server with the least amount of active connections. State checking is performed to monitor the utilization of available Web servers.

There exist numerous scheduling algorithms with which the load balancer can distribute inbound requests to back-end servers. A survey of these algorithms can be found in Shirazi, Krishna, and Ali (1995). Previous works have analyzed and compared load balancing techniques for scalable web servers (Bryhni, Klovning, and Kure 2000). However, scalable in this context refers to the system's ability to expand capacity by possessing the inherent ability to support additional workstations. Although on the surface this work appears to have similar goals - the expansion of the capacity - as in the cloud model, it is not triggered by real-time utilization parameters. Despite the fact that Bryhni, Klovning, and Kure (2000) results show that the Round-Robin scheduling algorithm exhibits the best performance among homogenous servers that have the same processing capacity, these results are not applicable to the cloud computing model.

3.2 Web Server

The back-end Web servers in the cloud framework are modeled as separate processes and queues. Each Web or HTTP server is modeled as if it has its own network interface, CPU, memory and hard disk. In addition, the web server software is modeled as if it were an Apache server. In this setup, the HTTP daemon process listens for client requests, relayed via the load balancer. When a client initiates a service request, the listening processes spawn a new child process or awaken a dormant child process to handle the request. As specified in the HTTP/1.1 protocol, each Web page forks a new HTTP process that serves the request and all associated content. Although at any particular point in time multiple child processes may be concurrently active, the client requests are still processed initially by the HTTPD process, which is inherently restricted by the listening queue's first-in first-out (FIFO) queuing policy (Casalicchio and Tucci 2001; Laurie and Laurie 1997). As a result, the web server is modeled as a single server queue and service times are exponentially distributed.

3.3 Horizontal Scaling

The horizontal scaling ability afforded to the user by the cloud infrastructure is based on the needs of the system and can efficiently allocate and de-allocate computing resources (Beaty, Kochut, and Shaikh 2009). Those who employ cloud-based services no longer have to worry about anticipating appropriate amounts of computing resources. Similarly, service providers are better suited to actively monitor system utilization and more accurately assess capacity planning. The ability to run a business that has seemingly

infinite resources without making investments in hardware and software is unprecedented in the history of IT (Armbrust et al. 2009).

In static resource allocation configurations there inevitably exists a trade-off between capacity deployment and resource demand. On one hand, as seen in Figure 3, when the capacity of the system supersedes the demand, the supporting infrastructure is heavily under-utilized. As a consequence, the customer unnecessarily incurs financial costs. On the other hand, if the system capacity is not able to serve all of the user-generated service requests, as seen in Figure 4, the quality of service of the system diminishes as customers experience delays or unavailable resources.

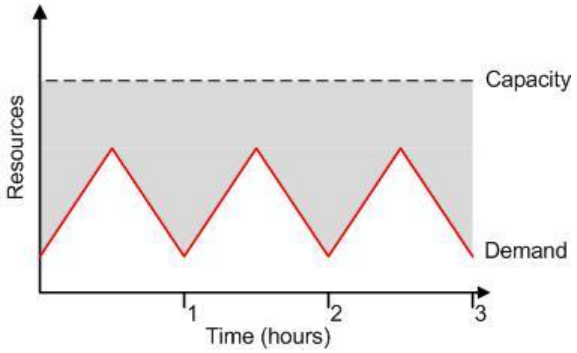

Figure 3: Capacity Over-Provisioning

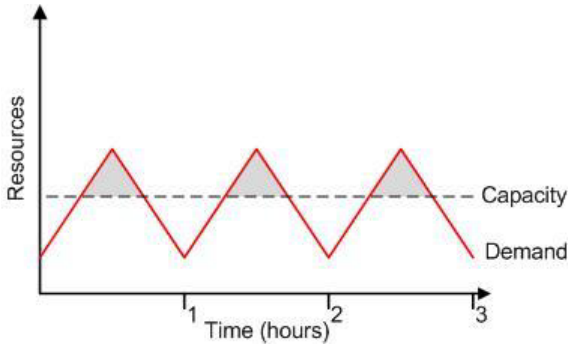

Figure 4: Capacity Under-Provisioning

If it is possible to predict peak-capacity, then the system can be designed to accommodate for peak-load as seen in Figure 5. Regardless of which one of the three previously described methods is employed, the system will inevitably experience either underutilized resources, resulting in less than desirable return on investment, or overutilization, in which the quality of service experienced by the customer diminishes. In reality, although flash-crowds present a formidable challenge, the more common of the two setbacks is under-utilization. It has been estimated that server utilization in data centers range from 5% to 20% (Armbrust et al. 2009).

Represented in the simulation model and the resulting framework presented in this paper are horizontal scaling functionalities that allow the cloud-based resources to respond dynamically to service requests by scaling capacity either up or down according to user-defined conditions, as illustrated in Figure 6. This type of capacity scaling is particularly useful to web applications that are subjected to a utility pricing model that experiences hourly, daily, weekly, or seasonal variability in usage. In addition, the advantage of the cloud model in this situation is that resource provisioning allows additional servers to be made available in a matter of minutes, rather than days or weeks.

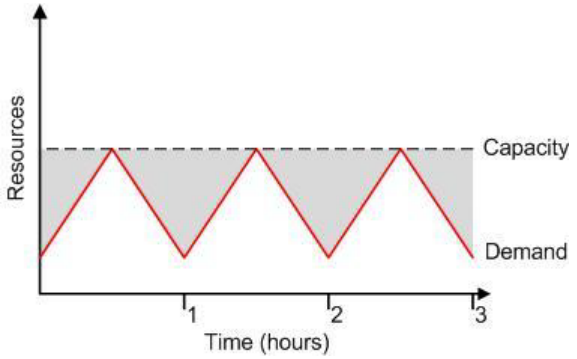

Figure 5: Peak-Load Capacity Provisioning

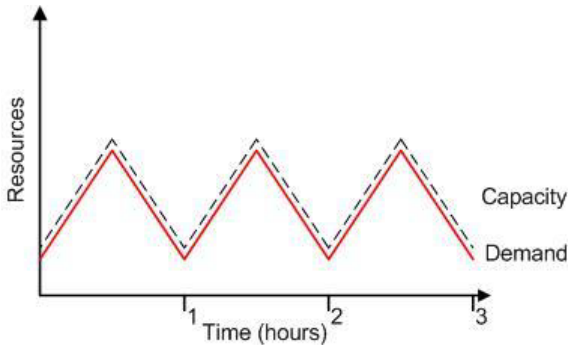

Figure 6: Capacity On-Demand Provisioning

3.4 Horizontal Parameters

The horizontal scaling profile is composed of parameters that define the cloud model metrics and thresholds that dictate that responsiveness of the system - namely, whether or not to increase or decrease resources in response to customer demand. Horizontal scaling profiles, defined by the customer, are the essential components of the cloud model. The descriptions of the horizontal scaling parameters as described by Amazon's Elastic Compute Cloud are as follows (Amazon 2010):

Table 1: Horizontal Scaling Parameters

Horizontal Scaling Parameters	Definition
Period	Frequency of time in which measurements are taken for the trigger metric.
Trigger	A metric that is monitored during a period (i.e. CPU utilization, network activity, or disk utilization).
Breach Duration	The amount of time (in seconds) used to evaluate and determine if a breach has occurred.
Horizontal Scaling Group	The number of available servers to be allocated or de-allocated.
UpperThreshold	The upper limit for the trigger.
UpperBreachScaleIncrement	The incremental amount of resources applied when the *UpperThreshold* has been breached.
LowerThreshold	The lower limit for the trigger.
LowerBreachScaleIncrement	The incremental amount of resources de-allocated when the *LowerThreshold* has been breached.

In this simulation model, for example, horizontal scaling is enabled when the defined trigger (e.g. CPU utilization) exceeds 70% capacity or falls below 30% capacity utilization. When the *Breach Duration* of the *Trigger* surpasses the *UpperThreshold* the *UpperBreachScaleIncrement* is reflected upon the system (e.g. increase available servers by one). Likewise, when the *Breach Duration* of the trigger dips below the *LowerThreshold* or the *LowerBreachScaleIncrement,* resources are subtracted from the horizontal scaling group.

The trigger metric used in the simulation model is the overall percent utilization of the system calculated as the aggregation of the load experienced across all the active servers in the horizontal scaling group divided by the maximum load for each Web server. The trigger metric is calculated as follows:

$$U_t = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} load_{i,j} / n}{\max \ load}$$

4 SIMULATION EXPERIMENT AND RESULTS

4.1 Metrics

The performance of the load balancing algorithms and horizontal scaling profile configurations of the cloud model are analyzed by means of the Load Balance Metric (LBM) proposed in (Bunt, Eager, and Oster 1999). The measure of load balancing between nodes is calculated by means of the peak-to-mean ratio of Web server load based on the current state of Web server at periodic times throughout the simulation. The LBM is then calculated by taking the weighted average of the peak-to-mean ratios experienced across the Web servers. The load of the Web server is defined as the number of requests being simultaneously processed or in queue among the active Web servers.

The load of Web server i (of n servers) at the j^{th} observation (of the observation period m) is represented as $load_{i,j}$ and $peak\_load_j$ is a measure of the highest load experienced by any of the active

servers at the j^{th} sampling point; where q_i is the weighted average of each Web server. The LBM is defined as follows:

$$LBM = \frac{\sum_{j=1}^{m} peak\_load_j}{\sum_{j=1}^{m}\sum_{i=1}^{n} load_{i,j} \cdot q_j}$$

The value of the LBM can range from one to the number of n Web servers. The smaller values of the LBM indicate better load balance performance, with an LBM of one being an ideally-balanced load. The weighted average of each Web Server is necessary when evaluating the performance of the cloud computing model due to the imbalance of time each server experiences throughout the simulation.

When the threshold of capacity of the available Web servers in the cloud computing model is exceeded, the resulting action is packet loss. The amount of packet loss experienced by the system is calculated as the ratio between rejected and received packets for all servers. Where f_j is the number of packets rejected by server$_j$ and r_j is the number of packets served by server$_j$. The Rejection Ratio (*RRatio*) is defined as follows:

$$RRatio = \frac{\sum_{j=1}^{n} f_j}{\sum_{j=1}^{n} f_j + r_j}$$

In the following section, the behavior of the cloud computing model will be analyzed in respect to the described load balancing algorithms and also the horizontal scaling configurations defined by the cloud computing model in respect to the LBM and Rejection Ration metrics.

4.2 Preliminary Results

Initial simulation experiments were performed to evaluate the impact of the load balancing algorithms on the LBM for a standard configuration policy (Table 2). Analysis was performed on the simulation model as the *UpperThreshold* varied between 0.6 and 0.9 percent tolerance while the *LowerThreshold* remained fixed at 0.2 percent tolerance.

Table 2: Profile 1

Horizontal Scaling Parameters	Value
Period	10 Seconds
Trigger	% Utilization
Break Duration	200 Seconds
Horizontal Scaling Group	10
UpperBreachScaleIncrement	+1
UpperThreshold	0.6 – 0.9
LowerBreachScaleIncrement	-1
LowerThreshold	20 %

Figure 7 represents a graph of the resulting LBM for the described simulation. The Least-Connection distribution algorithm outperforms both the Round Robin and Random algorithms, achieving a LBM close to one. As the cloud model reacts to service requests, the load balancer is most effective when a state-aware algorithm is deployed. Because state-aware algorithms can serve as the bottleneck of the system, more sophisticated algorithms were not considered. Instead, both Round Robin and Random algorithms were chosen. However, the results of the study indicate that these two algorithms simply do not

effectively distribute the service requests, and as a result, both of these algorithms in the cloud model experiences a non-zero rejection rate when subjected to constant service requests.

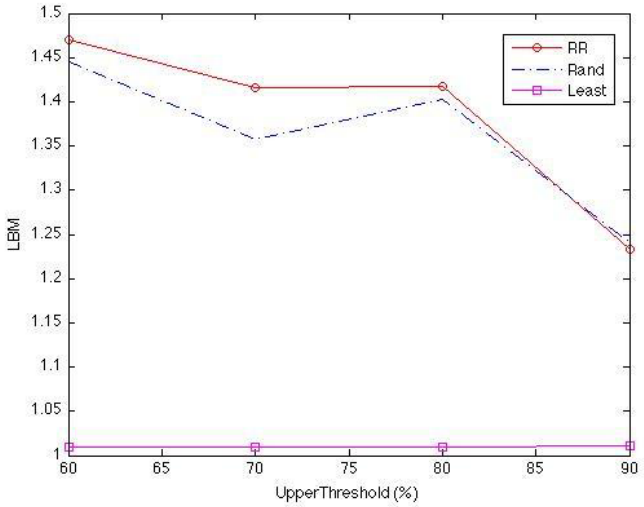

Figure 7: LBM Experimental Results

Analyzing the previously described simulation experiment further shows that the selection of the load balancing algorithm also has a significant effect on both the overall utilization of the servers (Figure 8) as well as the rejection ratio (Figure 9), which in turn directly affects the quality of service by the customers. In respect to overall utilization of the servers available in the horizontal scaling group for Profile 1 (Table 2), the least-connection distribution algorithm outperforms both the Round Robin and Random algorithms. This is no surprise considering that the results were obtained from the analysis of the LBM. Similarly, the rejection ratio experienced by the Least-Connection algorithm was zero while the other two algorithms under consideration performed poorly in comparison.

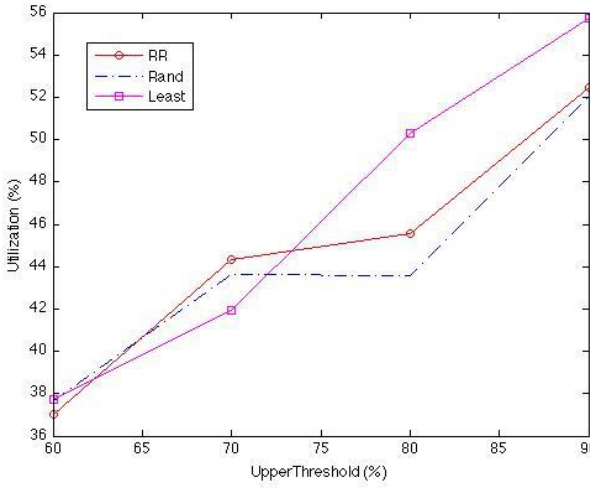

Figure 8: Utilization Comparison

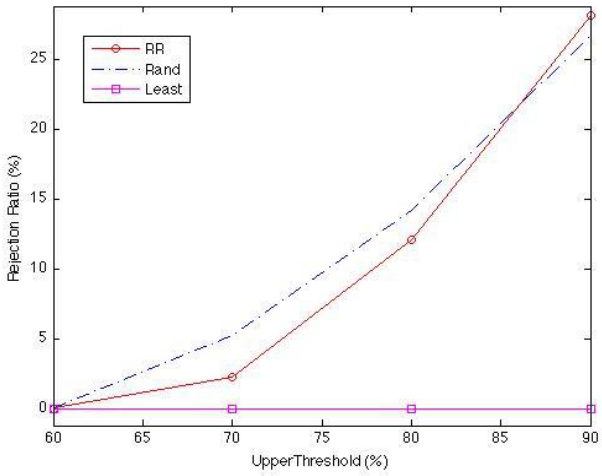

Figure 9: Rejection Ratio Comparison

The second set of initial simulations was performed to better understand the effect that the number of available resources in the horizontal scaling group had on both the utilization and rejection ratio as the upper threshold varied between 0.6 and 0.9 percent. From the previously described results, the Least-Connection algorithm was chosen for the following analysis. Both Profile 2 (Table 3) and Profile 3 (Table 4) have the same respective configurations while only deviating only in the size of the horizontal scal-

ing group, five and six servers, respectively. As seen in Figure 10, overall utilization significantly decreases as the number of available servers in the horizontal scaling group increases. Similarly, the rejection ratio of service requests increases; which is to be expected as the utilization approaches 98%. There is a clear threshold that is surpassed when the horizontal scaling group is increased by a single server. In the case of Profile 2 (Table 3), the service user experiences higher utilization at the cost of a higher rejection ratio, while in Profile 3 (Table 4) the service user is subjected to marginal utilization with a nearly zero rejection ratio.

In this same experiment, it was also shown that percent utilization of the overall system increases as the *UpperThreshold* increases. Although this particular result should come as no surprise, the resulting side effect of connection rejections increases with the increased utilization as servers experience a workload closer to their overall peak capacity.

Table 3: Profile 2

Horizontal Scaling Parameters	Value
Period	10 Seconds
Trigger	% Utilization
Break Duration	200 Seconds
Horizontal Scaling Group	5
UpperBreachScaleIncrement	+1
UpperThreshold	Figure 10
LowerBreachScaleIncrement	-1
LowerThreshold	20 %

Table 4: Profile 3

Horizontal Scaling Parameters	Value
Period	10 Seconds
Trigger	% Utilization
Break Duration	200 Seconds
Horizontal Scaling Group	5
UpperBreachScaleIncrement	+1
UpperThreshold	Figure 10
LowerBreachScaleIncrement	-1
LowerThreshold	20 %

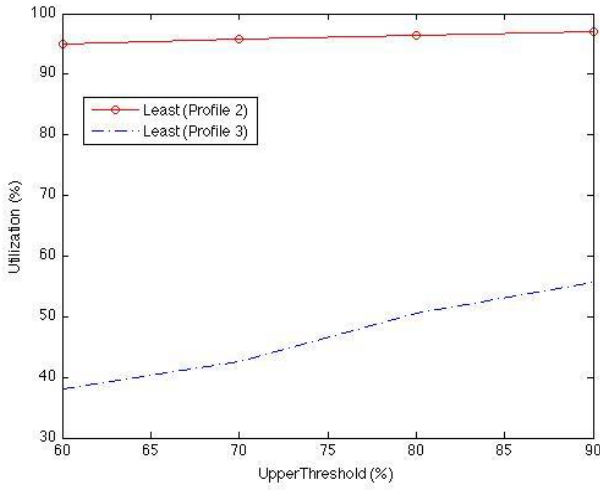

Figure 10: Utilization Comparison

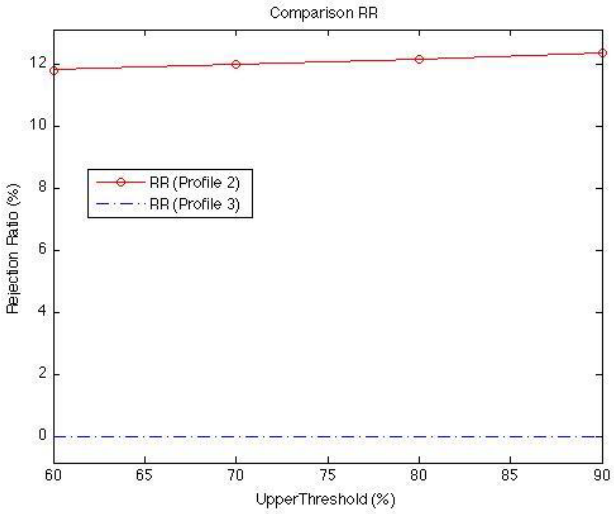

Figure 11: Reject Ratio Comparison

Overall, the Least-Connection algorithm in this given context outperforms both the Round-Robin and Random distribution algorithms. From the initial results presented in this paper, it can be seen that the horizontal scaling parameters have a significant impact on the inevitable trade-offs that occur between the overall utilization of the system and quality of service in the cloud model.

5 DISCUSSION AND FUTURE WORK

The limitations of this work are in the granularity and protocol complexity of the model. Because of these constraints and to provide more robust and application-specific analysis, future work will be conducted in the NS-2 network simulator. Further experimentation will also be performed on existing physical implementations of the cloud computing model such as Amazon's EC2, which served as the motivation for this paper. Although there has been significant work in areas closely related to the cloud model, to the best of the author's knowledge there does not exist any previous work that explores modeling the utilization and resource allocation in the pubic cloud computing model.

There exist a number of research questions to be answered with the cloud model and future work in this area will be significant. Areas of interest for future evaluation include price-based scaling parameters that allow the consumer to not only control their cloud resources based on consumption metrics but also on financial parameters. One of the largest motivations for adopting the cloud model is cost savings and it is in the CSP customer's best interest of obtain high utilization within a forecasted budget.

While the utility pricing model presents a flexible payment option, it also is vulnerable to malicious resource use that is intended to run up the operating expenses for public cloud service customers. Research that focuses on Denial of Service attacks that seek to not only to disrupt cloud services but also to abuse the utility model on which cloud service are based will be necessary to ensure the long-term economic viability of the public cloud computing services.

6 CONCLUSION

The cloud computing model represents a significant shift in the computing paradigm. The discrete event simulation model described in this paper presents a framework with which to analyze the many facets of the cloud computing model. As evidenced in the experimental results presented in this paper, a state-aware load balancing algorithm is preferable to achieve this goal. The inevitable trade-off presents an interesting research area in which higher granularity of profile configuration settings can be explored to achieve higher efficiency and fewer rejections. Furthermore, to increase utilization, there exists a trade-off between overall utilization and the rejection ratio. This paper provides preliminary results to better understanding the cloud computing model and a framework within which further research can be conducted.

ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their valuable comments and suggestions and also Dr. Doug Jacobson and Mark Tannian for their support, feedback, and contributions to this work.

REFERENCES

Amazon. 2010. Amazon Elastic Compute Cloud. Available via: <http://aws.amazon.com/ec2/> [accessed April 2, 2010].

Armbrust, M., A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. 2009 Above the Clouds: A Berkeley View of Cloud Computing. Available via: <www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf> [accessed October 20, 2009].

Beaty, K., A. Kockhut, and H. Shaikh. 2009. Desktop to Cloud Transformation Planning. *IEEE International Symposium on Parallel and Distributed Processing* 1-8.

Bourke, T. 2001. *Server load balancing*. 1st ed. Sebastopol, CA: O'Reilly & Associates, Inc.

Bryhni, H., E. Klovning, and O. Kure. 2000. A Comparison of Load Balancing Techniques for Scalable Web Servers. *IEEE NETWORK* 14:58-64.

Bunt, R., D. Eager, G. Oster, and C. Williamson. 1999. Achieving Load Balancing and Effective Caching in Clustered Web Server. In *Proceedings of the International Web Caching Conference*.

Casalicchio, E., and S. Tucci. 2001. Static and Dynamic Scheduling Algorithms for Scalable Web Server Farm. In *Euromicro Workshop on Parallel and Distributed Processing*, ed. K. Klockner, 369-376.

Foster, I., Z. Yong, I. Raicu, and S. Lu. 2008. In *Grid Computing Environments WKSH*, ed. R. Buyya 1-10.

Friedrich, S., S. Krahmer, L. Schneidenbach, and B. Schnor. 2006. Loaded: Server Load Balancing for IPv6. In *International Conference on Networking and Services*, eds. P. Dini, C. Popoviciu, C. Dini, G. Lambert, G. Van de Velde, N. Pitatla, and I. Jayasumana. 8-13.

Judge, J. 1999. A Model For the Marginal Distribution of Aggregate Per Second HTTP Request Rate. In *IEEE Workshop on Local and Metropolitan Area Networks*, eds. D. Skellern, A. Guha, and F. Neri. 29-36.

Laurie, B., and P. Laurie. 2002. *Apache: The Definitive Guide*. 3rd ed. Sebastopol, CA: O'Reilly & Associates, Inc.

Mell, P., and T. Grance. 2009. The NIST Definition of Cloud Computing. Available via; <http://csrc.nist.gov/groups/SNS/cloud-computing/> [accessed March 21, 2010].

Shirazi, B. A., K. Krishna, and H. Ali. 1995. *Scheduling and Load Balancing in Parallel and Distributed Systems*. Wiley-IEEE Computer Society Press.

Voas, J., and J. Zhang. 2009. Cloud Computing: New Wine or Just a New Bottle? *IT Professional* 11:15-17.

Ying-Wen, B., and Y. Yu-Nien. 2006. An Analysis and Measurement of the Equivalent Model of Serial Queues for a Load Balancer and a Web Server of a Web Cluster With a Low Rejection Rate. In *IEEE International Symposium and Workshop on Engineering and Computer Based Systems*. eds. M. Riebisch, P. Tabeling, and W. Zorn. 486-487.

AUTHOR BIOGRAPHIES

JOSEPH IDZIOREK is a Ph.D. student in the Electrical and Computer Engineering Department at Iowa State University. His research interests lie in the field of computer and network security with a focus on the cloud computing model, Distributed Denial of Service attacks and network simulation models. His email address is <idziorek@iastate.edu>.