Journal of Theoretical and Applied Information Technology
15th October 2011. Vol. 32 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PERFORMANCE EVALUATION OF A CLOUD BASED LOAD BALANCER SEVERING PARETO TRAFFIC

AYMAN G. FAYOUMI
Department of Information Systems
Faculty of Computing and Information Technology
King Abdulaziz University
Jeddah21589, SAUDI ARABIA
E-mail: afayoumi@kau.edu.sa

ABSTRACT

Cloud resources represent an unforeseeable breakthrough in ITC industry. Load balancer is a key element in resource provisioning for high available cloud solutions, and yet its performance depends on the traffic offered load. We develop a discrete event simulation to evaluate the performance with respect to the different load points. The performance metrics were the average waiting time inside the balance as well as the number of tasks. The performance study includes evaluating the chance of immediate serving or rejecting incoming tasks. Pareto traffic was considered for the offered traffic.

Keywords: *Cloud Computing, Load Balancer, Pareto Distribution, Self-Similar Traffic, Discrete-Event Simulator*

1. INTRODUCTION

Network servers are resources used to host applications such as ERP, e-commerce, etc. These resources are expected to provide high performance, high availability, and secure and scalable solutions to support hosted applications. Many content-intensive applications show elastic load behavior that might scale up beyond the potential processing power of a single server. At the same time, enterprises organization and service providers need a dynamic capacity of server resources to deploy as many servers as needed seamlessly and transparently. Server load balancing structures multiple servers as a single virtual server by transparently distributing user requests among the servers, as shown in Figure 1. Nevertheless, depending on the offered load at the balancers, the performance might get degraded and hence the overall cloud solutions. This study evaluates the performance of the load balancers with respect to the average waiting time inside the balance as well as the number of served tasks against the offered load traffic.

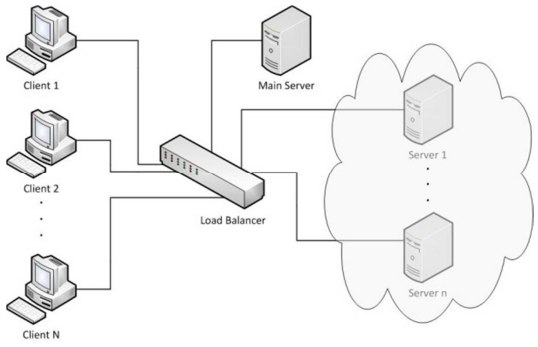

Figure 1. Load Balanced Server Farm

Authors in [15] present a comparative study among performance of four simulation models for four different load balancing algorithms (static, round robin, diffusive and short queue). This comparison was made in three different client-server environments (small-scale, intranet and internet). Diffusive load balancing was proven to be efficient in a dynamic environment.

Authors in [7] describe an algorithm for load balancing in cloud computing environment which takes into account the dynamic task requirements. The algorithm schedules the virtual machine to the host with the lightest load each time. Thus improve

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the resource utilization, which will enhance the overall performance of the cloud computing environment. They used the *CloudSim* toolkit to simulate this algorithm. The simulation results of both grid and cloud environments were compared. The resource utilization of the cloud environment was higher than that of the grid environment.

Since access to the infrastructure over the cloud incurs payments, *CloudSim* simulation-based approaches was used in [3] to simulate prototyping different services in repeatable and controllable environment free of cost where addressing the performance bottlenecks prior to deploying these services is applicable.

Authors in [2] employed Distributed Machine Simulation *DiMSIM* that is designed to simulate load balancing algorithm by implementing four load balancing algorithms (Random placement ,Threshold ,Join shortest queue JQS ,and the author algorithm BGQS that JQS enhanced) on local area network using real workload. The result shows that the BGQS and JQS are similarly performing.

Many empirical studies have shown that heavy-tailed Pareto distribution provides a good fit for a number of important characteristics of communication networks and Web servers. For instance, local and off the Internet file sizes could be modeled by Pareto distributions [5]. The sizes of user requests in the Internet were found to be heavy-tailed distributions as well [16]. More importantly, the service times of packets handled in network switches were also observed to satisfy heavy-tailed distributions [4,6].The authors in [6] develop a rule based mapping algorithm for virtual machines

2. LOAD BALANCED SERVERS

Figure 1shows load balancing within a server farm. Load balancer can provide superior performance by utilizing processing power of servers intelligently as to direct end-user service requests to least busy servers, thus providing fastest response. More importantly, the load balancer device should be capable of handling the aggregate traffic of multiple servers; otherwise it might represent a bottleneck.

Of course most appealing aspect of the load balanced resources is providing high available infrastructure [12]. If an application fails or a server goes down, due to either overload or planned outage, the load balancer can automatically redistribute end-user service requests to an available server within a local server farm or off the could. Distributed server on a load balanced environment can also provide disaster recovery services by redirecting service requests to a DR site as a result of a catastrophic event that might disables the primary site.

Cloud balancing approach extends the architectural deployment model of traditional load balanced servers to be used in conjunction with global load balanced servers off the cloud [12]. This approach increases the choices available from where a given application should be delivered and hence increases the application routing options. Presumably, these routing options are associated with decisions based on both technical and business goals of deploying load balanced global application. Technical goals can be related to the performance and the availability of the application while business goal be related to the incurred cost as the regulatory and security issues.

3. PARETO DISTRIBUTION AND SELF-SIMILAR TRAFFIC

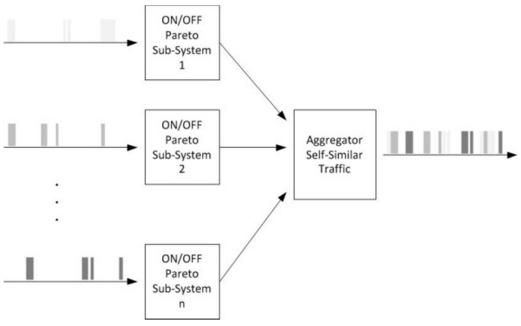

Figure 2. Generating Self-similar traffic

Pareto distribution, named after the Italian economist Vilfredo Pareto, is a power law probability distribution that coincides with the behavior of the Internet traffic. To generate a Pareto distributed sequence, one can generate a Pareto distributed sequence of interarrival times of simulating arrival of requests where each demands a Pareto distributed service time.

The Probability Density Function (pdf) of a Pareto distributed random variable*X* is:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$f(x) = \frac{\alpha b}{x^{\alpha+1}}$$

where α is the shape parameter (tail index), and *b* is the minimum value of *X*. When $\alpha \leq 2$,the variance of the distribution approaches infinite. When $\alpha \leq 1$, the mean value is infinite as well.

For self-similar Pareto traffic, α should be between 1 and 2 [10, 14]. The lower the value of α, thehigher the probability of an extremely large *X*.

Mean value of the Pareto distribution is:

$$E(x) = \frac{\alpha b}{\alpha - 1} \quad 1 < \alpha$$

And the variance of the Pareto distribution is:

$$V(x) = \frac{\alpha b^2}{(\alpha-1)^2(\alpha-2)} \quad 2 < \alpha$$

Given that the random variable*U* is uniformly distributed number on the interval (0,1), Pareto-distributed random variable *X* can be generated by:

$$X_{Pareto} = \frac{b}{U^{\frac{1}{\alpha}}}$$

4. CONCEPTUAL MODEL

As shown in Figure 1, the simulatedmodel used to study the characteristics of the arrival task occupancy of the load balancer buffer consists of a main server running a specific application. In case that the server is busy or down, the load balancer forward the arriving request to one of the secondary server(s) running and configured as the primary server and might be located off the cloud. In case that the primary and all applicable servers off the cloud are busy, the request gets buffered in a buffering area inside the balancer until a server becomes free.

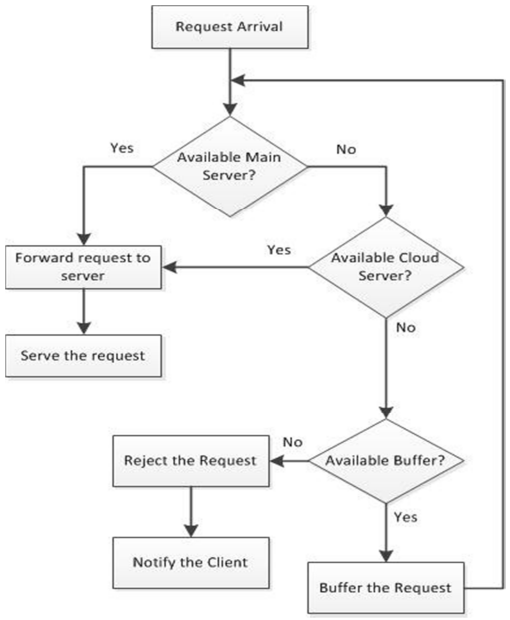

Figure 3. The flow chart of processing incoming requests

Figure 3 shows the flow chart of the requests upon arrival at the load balancer.The system's *QoS* metrics to be evaluated are the average waiting time in the load balancer, the distribution of number of requests waiting in the balancer's buffer, and the rejection probability at the load balancer.

5. SPECIFICATION MODEL

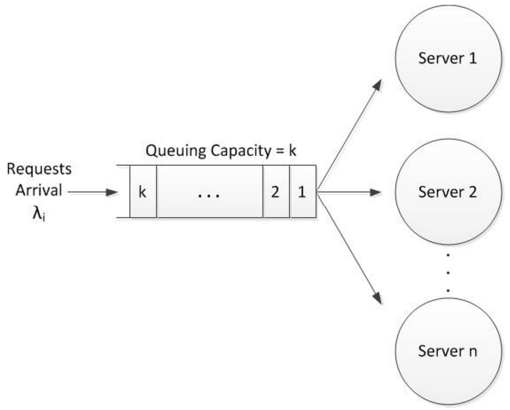

Figure 4. Queuing System

The described model in Conceptual Model Section is shown in Figure 4. Note that the arrival process is assumed to obey Pareto distribution of a rate of λ_i while the service duration of a request is assumed to follow Pareto distribution of a rate of λ_s. A reasonable estimation of the offered load at the system would be the ratio of the mean service time

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of the incoming requires to the mean inter-arrival time of the stream, where the rate is the reciprocal of the mean. Thus, the offered load, ρ, can be found as:

$$\rho = \frac{\lambda_i}{\lambda_s}$$

The model under consideration consists of two servers; namely, a main server and a backup server off the cloud. The load balancer can hold up to *k* requests where they are forwarded to a server according to their arrival; i.e., the incoming requests get server in a *FIFO* discipline. The servers serve requests conservatively; i.e., no server stays idle while there is a request waiting. The servers are non-preemptive; i.e., non-interruptible once start serving requests. Finally, in this study, we assume the minimum value of *X* is X_{min}=1.

5. SIMULATION

The stochastic processes of the simulated system are generated using Multi-Stream Lehmer Random Number Generation technique, where a designated steam is allocated for each of the stochastic process to insure un-correlated stochastic processes. Moreover the results were not collected until the system reaches the steady state as indicated by the mean waiting time in the system as well as the average number of requests within the system.

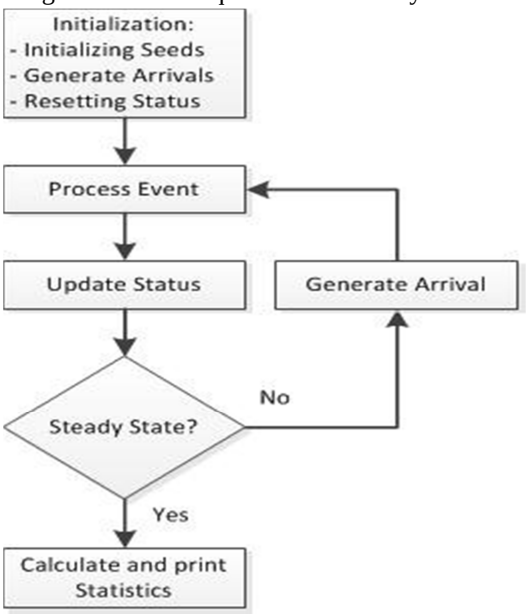

Figure 5. Simulator Flow Chart

The flow chart of the simulator is presented in Figure 5. At the beginning of the simulator, random number generator seeds are initialized. Also random requests arrivals at the load balancer are also generator. The simulator then starts processing the events in a *FIFO* manner. Each time an event is processed, it is the status of either the main server or the cloud server gets flagged with *BUSY* or *FREE*. A newly arrival request is generated each time an event is processed. Note that an even can be an arrival request, set a server to a *BUSY* status, or set a server to a *FREE*state. The simulator keeps running until the steady state is identified in terms of the average number of the requests in the balancer as well as the average waiting time of the request as to reach a deference that is not more than 0.1%.

A Discrete-Event Simulator was built to simulate a load balancer system with type of *Pareto/Pareto/2/k* queuing system, where two servers were considered; namely, main and could. Note that the arrival rate to the balancer is considered to follow Pareto distribution and the service rate of each of the incoming request is considered to follow Pareto distribution. The balancer's buffer can hold up to *k* requests to be forwarded to the main or the cloud server; i.e., the buffer capacity of the balancer, in terms of number of requests, is *k*. The main algorithm that the load balancer follows in forwarding the incoming is shown in Figure 4.

6. NOTATION AND VALIDATION

The main results investigated in this paper are summarized as follows:

- λ_m: The effective arrival rate to the main server
- λ_c: The effective arrival rate to the cloud server
- $\bar{w}$: average waiting time inside the system
- $\bar{d}$: average delay in the buffer
- $\bar{s}$: average server time in each server
- $\bar{l}$: average number of requests in the system
- $\bar{q}$: average number of requests in the buffer

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- $\bar{x}$: average number of requests per server (Server Utilization)
- P_r: probability of a request gets rejected
- P_d: probability of a request gets serviced without getting buffered
- P_b: probability of a request gets serviced after getting buffered

The simulator was validated by comparing its results with those of proven formulas of *M/M/c/k* queuing system. Results of the formulas of the average waiting times in the system as well as in the buffer were compared to that counterpart of the simulator that considers exponential distributed random variable for both of the inter-arrival time and service time. Additionally, the average number of requests in the systems as well as in the buffer of both theoretically proven formulas and simulation were also compared. Both results of the theoretical and simulation were almost identical. Figure 6 shows the comparison of the average numbers in the system. The average waiting times were too small to be presented.

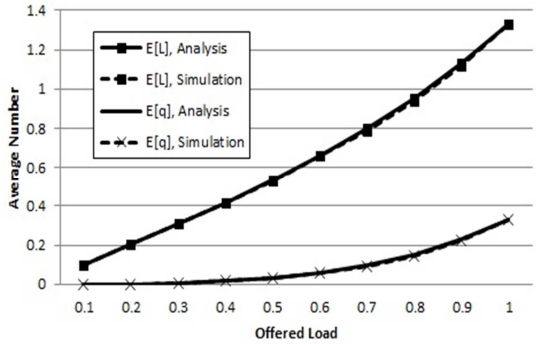

Figure 6. Simulation Validation, E[L]: expected number of requests in the system, E[q]: expected number of requests in the queue

Little's Law [11] was used to calculate the average number of requests in the system. The law stated that during the steady state of a system, the average number of requests is equal to their average arrival rate, multiplied by their average time spent in the system. Little's law was used in to derive the average number of requests in the system, buffer, and per server.

For another validating the results of the simulation, the follow equations were used and tested to hold true.

$$\bar{w} = \bar{d} + \bar{s}$$

$$\bar{l} = \bar{q} + \bar{x}$$

$$P_r + P_d + P_b = 1$$

7. RESULTS

In this section, we present the evaluation of the performance of the cloud based load balancer with Pareto distributed incoming and inter-arrival packets.

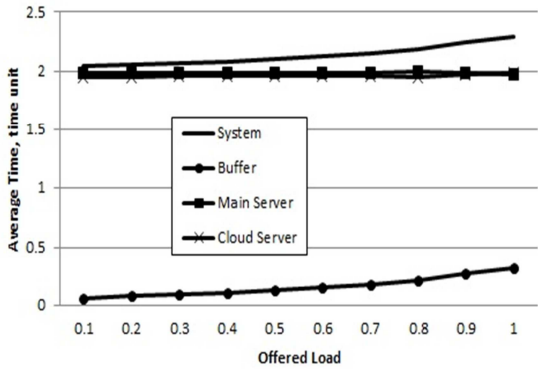

Figure 7. Average Time Spent

Figure 7 shows the breakdown of the average time spent in the system. The time spent in the servers is quite constant and it represents the average service time. The request response time is somehow dominated by the service time as the buffering time represents small portion of the total response time.

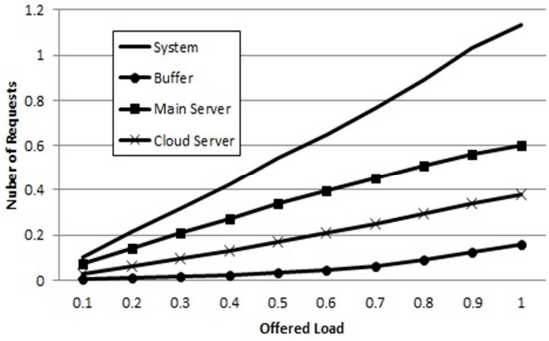

Figure 8. Average Number of Requests

The average number of requests is shown in Figure 8. The figure shows that the number of requests in

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the main servers is *30%* more than that of the cloud server. It also depicts that the requests occupancy of the buffer is noticeable only at higher offered loads $\rho \geq 70\%$. This shows that with specification model under consideration, the buffer plays remarkable role only when the system under stress.

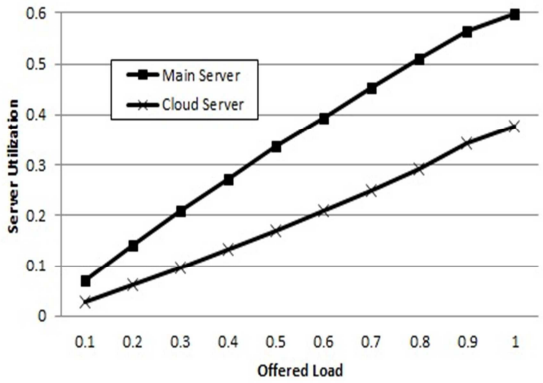

Figure 9. Server Utilization

Figure 9 presents that the main server is utilized at least *30%* more than the clouds based server. This results is helpful is sizing the hardware of load balanced main-cloud server system under certain workload.

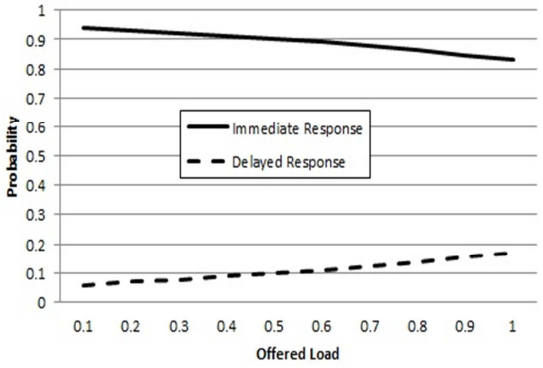

Figure 10. Probability of Delayed and Immediate Response

Figure 10 shows the probabilities of direct response versus delay response. It can be inferred that requests are mainly responded to immediately most of the time; i.e., the system is well sized according to the workload under consideration. Note that the system under consideration experiences no overload; i.e., no chance of requests rejection.

8. CONCLUSION

We evaluated the performance of a load balanced cloud server system under different offered loads. The results show that the buffer of the load balance plays marginal role except at very high loads. It also show that the main server handle at least *30%* as much requests at the cloud based server. It will be very informative to pursue the study of optimizing the buffer size that meets the minimal rejection probability. The future work is to compare the performance evaluation of systems considering different combinations of service time and inter-arrival time distributions.

REFERENCES

[1] R. Buyy, R. Ranjan, R. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities". Proceedings of the Conference on High Performance Computing and Simulation (HPCS 2009), June 2009

[2] J. Cao , G. Bennett , K. Zhang, "Direct execution simulation of load balancing algorithms with real workload distributed", Journal of Systems and Software, vol. 54, no. 3, p.227-237, November 2000

[3] Y. Cheng; K. Wang; R. Jan; C. Chen; C. Huang; ``Efficient failover and Load Balancing for dependable SIP proxy servers", IEEE Symposium on Computers and Communications, pp. 1153 - 1158, 2008

[4] A. Downey, ``Evidence for long-tailed distributions in the internet," Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, pp. 229-241, 2001.

[5] A. Downey, ``Lognormal and Pareto distributions in the Internet," Computer Communications, vol. 28, no. 7, pp. 790-801, 2005.

[6] D. Ersoz, M. S. Yousif, and C. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07), pp. 59-68, 2007.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[7] Y. Fang, F. Wang. J. Ge, ``A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing", Springer-Verlag Berlin Heidelberg , pp. 271-277, 2010.

[8] S. Ghani, F. Iradat, ``Loss Probability in Networks with Pareto Distributed Traffic," Second International Conference on Intelligent Systems, Modeling and Simulation (ISMS), pp. 355-360, 2011.

[9] C. Keller, A. Niehorster, O. Brinkmann, ``Rule-Based Mapping of Virtual Machines in Clouds", Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), February 2011

[10] Kramer, ``On Generating Self-similar Traffic Using Pseudo-pareto Distribution," A Technical Brief, Department of Computer Science, University of California, 2000.

[11] L. Leemis and S. Park: Discrete-Event Simulation: A First Course, Pearson, 2005

[12] L. MacVittie, ``Cloud Balancing: The Evolution of Global Server Load Balancing", F5 White Paper, www.f5.com/pdf/white-papers

[13] K. Park and W. Willinger, Self-Similar Network Traffic and Performance Evaluation. John Wiley & Sons, 2000.

[14] A. Reza, L. Hyotaek, ``Performance evaluation of hybrid buffering optical packet switch for Pareto traffic," 13th International Conference on Advanced Communication Technology (ICACT), pp. 470 - 474, 2011

[15] M. Soklic, ``Simulation of load balancing algorithms: a comparative study", ACM SIGCSE Bulletin, vol. 34, no. 4, December 2002

[16] C. Xia, Z. Liu, M. Squillante, L. Zhang, and N. Malouch, ``Web traffic modeling at finer time scales and performance implications," Performance Evaluation, vol. 61, no. 2-3, pp. 181-201, 2005.

AUTHOR PROFILE:

Dr. Ayman G. Fayoumi is assistant professor at the Faculty of Computing and Information Technology and Vice Dean of IT Deanship at King Abdulaziz University, Saudi Arabia. He chairs the Saudi national committee for ITC standardization besides his consultancy to IT sectors in many governmental and private organizations. His areas of expertise are Optical Networking, IT Strategic Planning, Smart IT, and Cloud Computing. He received his Ph. D. (2005) and M.S. (2001) in Computer Engineering from Colorado State University, MBA (2005) from University of Colorado in Business Administration. He has more than 22 research papers in refereed journals and conferences. He was listed in Chancellor's List of highest academic honors for graduate students in the USA in 2005. Dr Fayoumi is an editorial board member of the Journal of Computing and Applications, International Association for Computer Scientists and Engineers. He is also a founder of Saudi IT society.