Enhancing Cloud Computing Scheduling based on Queuing Models

Mohamed Eisa
Computer Science Department,
Port Said University,
42526 Port Said, Egypt

E. I. Esedimy
Computer Science Department,
Mansoura University,
Mansoura, Egypt

M. Z. Rashad
Computer Science Department,
Mansoura University,
Mansoura, Egypt

ABSTRACT
This paper presented a proposed model for cloud computing scheduling based on multiple queuing models. This allowed us to improve the quality of service by minimize execution time per jobs, waiting time and the cost of resources to satisfy user's requirements. By taking advantage of some useful proprieties of queuing theory scheduling algorithm is proposed to improve scheduling process. Experimental results indicate that our model increases utilization of global scheduler and reduce waiting time.

Keywords
Cloud computing; Queuing models; Scheduling process.

1. INTRODUCTION
Cloud Computing (CC) can be defined as a new style of computing in which dynamically scalable and often virtualized resources are provided as a services over the internet [1]. Cloud computing has become a significant technology trend, and many experts expect that cloud computing will reshape Information Technology (IT) processes and the IT marketplace.

The goal of CC is to provide end users with a considerable processing power and computing resources that allow them to run the applications and other user's requirements. In general, CC depends on the power and resources of computer networks. With this architecture, clients have access to the resources provided by the cloud provider as described in their Service Level Agreement (SLA). Clouds using virtualization technology and data centers to allocate distributed resources for clients as they need.

Often traditional scheduling techniques [2, 3] and allocation strategies [4] cannot be used in cloud computing, in which the number of end users requests increases and decreases over time in an unpredictable way. This leads to difficulties of analysis and discover of information from incoming requests to distribute the available resources according to user requirements and constraints of cloud provider. Similarly, unpredictable requests due to the increased costs of server load, maximum the total execution time of the task and the difficulty of making an optimal decision in the whole group of tasks. Amazon EC2 [5], introduce cloud services that allow users to acquire and release resources on-demand. Amazon EC2 also allows workflow systems to increase and decrease the pool of available resources when the demands changing and unpredictable needs of users are involved in the allocation process.

Several approaches are used for calculating server's queue waiting time in cloud computing. In these traditional approaches of cloud computing , only a single server ,called broker, serves all the entire end users and so the overload on that single server increases which affects the system performance. Qiang Li and Yike Guo [6] proposed a model for resource scheduling in the art of cloud computing based on linear programming, but none of these papers have been taken into account the concept of server utilization, queue length and the response time of the system. K.Mukherjee and G.Sahoo, [7] presented a mathematical model based on a Bee and Ant colony system for market-oriented cloud computing. Buyya et al. [8] have proposed an optimization algorithm that minimizes the response time of end users request's and their cost in the context of cloud computing. Shirazi et al. [9] introduce several scheduling algorithms based on distributed systems that assign the requests to the backend servers. Bryhin et al. [10] compare load balancing techniques for scalable web servers. Therefore, various queuing models are introduced to address this problem based on waiting queues models for each broker in the cloud system that increase system performance, reducing the average queue length and waiting time than the traditional approach of having only one server. Also, the incoming request not wait for along period of time and also queue length need not be large. On the other hand, there are different strategies for effective distribution of the load among the available servers. Random, Round Robin and Least-connection are different strategies that balance the load among distributed servers to minimize waiting times and optimize system performance [11].

The most important problem is how to build a model that can maximize server utilization and minimize waiting time in queuing models. Therefore, a mathematical model is proposed to deal with multiple tasks and resources based on the basis of maximizing the benefit of the cloud provider and decrease the response time of the system.

The main objective of this paper is to improve the performance of cloud system using queuing models as a tool. Furthermore, proposed model verified experimentally in several models that achieve higher utilization and response times compared to other models. Finally, scheduling algorithm that compute the lower and upper waiting time for all jobs at the waiting queues is introduced.

The rest of this paper is structured as follows. Section 2 introduces the preliminaries and notations. Section 3 discusses our proposed model construction. Section 4 describes the experimentation carried out by discrete event simulator and presents the results. Section 5 concludes the paper.

2. PRELIMINARIES AND NOTATIONS

2.1 Cloud Computing
Cloud computing allows users to run applications remotely, as shown in Figure 1, the first is the public cloud services which can be sold to anyone on the Internet (e.g., Amazon Elastic Compute Cloud (EC2) [5] and Google App Engine [12]). The second type of cloud is a private cloud that supplies hosted services to a limited number of customers (end users). In the type of hybrid cloud the infrastructure is a composition of two

or more clouds (private, community or public) that remain unique entities but are bound together by standardized or proprietary technology. In general, clouds are deployed to clients by giving them three access levels: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) [13]..

2.1.1 Software as a Service

Software-as-a-Service (SaaS) is a software distribution model in which applications are accessible through a single interface, like a web browser over the Internet. Users do not have to consider the underlying cloud infrastructure including servers, storage, platforms, etc.

2.1.2 Platform as a Service

Platform-as-a-Service (PaaS) provides a high level of integrated applications that control of distributed applications and their hosting environment configurations. In general, developers accept all instructions on the type of software that can be written to change built-in scalability.

2.1.3 Infrastructure as a Service

Infrastructure-as-a-Service (IaaS) provides users with computation processing, storage, networks and computing resources. IaaS users can implement an arbitrary application which is able to grow up and down dynamically .Also; IaaS sends programs and related data, while the cloud provider does the computation processing and returns the result.

2.2 Queuing theory

Queuing theory has emerged as a mathematical tool to deal with different types of queues [14].Waiting queue is an abstract representation whose goal is to isolate the factors that affect the system's ability to respond to service requests whose occurrences and durations are random. In general, models of simple queues are specified in terms of arrival process, service mechanism and discipline of the queue. The arrival process specifies the structure of the probabilistic way that service requests occur over time, the service mechanism describes the number of servers and the probabilistic structure of the period of time required to serve a user, and queue discipline specifies the order in which waiting users are selected from the queue for service.

The ultimate goal of waiting queues analysis is to understand the behavior of the underlying model so that intelligent and informed decisions can be made in their management. Therefore; the mathematical analysis of models produce formulas that measurers system performance, such as average waiting time, server utilization, throughput, the probability of exceeding buffer, the distribution of waiting time, the period of activity server, etc.

Waiting system was defined as, $WS = (S, R)$ where S is a set of servers $\{S = S_1, S_2, S_3, \ldots, S_n\}$, R is a finite set of requests $\{R = R_1, R_2, R_3, \ldots, R_n\}$, we assume that the types of requests and Sevres's queue are random, independent, identically distributed, adapted according to their order in the sequence of on a First-Come-First-Served (FCFS).

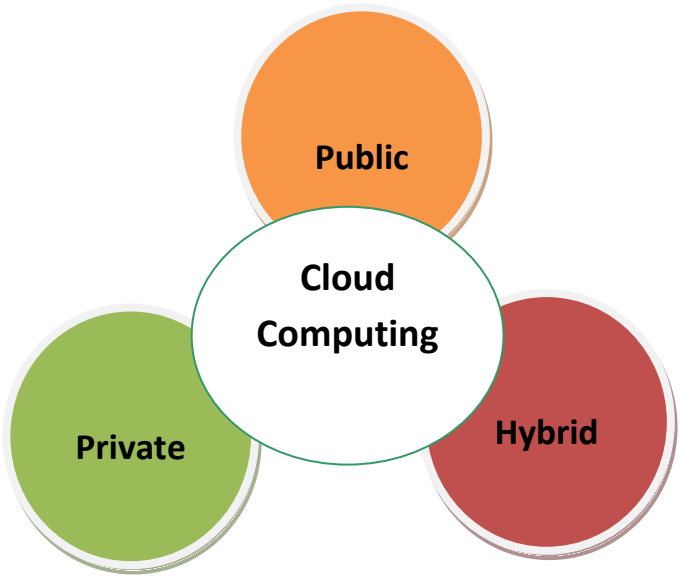

Fig. 1: Types of Cloud Computing.

The maximum processing time of the cloud server's queue provider can be computed using the following parameters;

D = Total demand rate,

R = Processing time rate,

c_o = Cost of processing unit, and

n = Number of waiting jobs.

As cloud computing introduces infinite resources so we are assuming that available processing time R will be large than the total processing time requirements D. Therefore; the idle processing time available is $(R\text{-}D)$. If we run scheduling process for t times and place $(R\text{-}D)$ available processing time

in ideal queue, the ideal queue at the end of processing will be (R-D) t. will be

$$\text{Maximum ideal queue length} = \sum_{i=0}^{n} (R-D)\, t \quad (1)$$

The total processing time available from cloud provider Q at a time t , then

$$Q = Rt \quad (2)$$

, and the length of the ideal queue t must be

$$t = \frac{Q}{R} \text{ times} \quad (3)$$

Then,

$$\text{Maximum ideal queue length} = \sum_{i=0}^{n} (R-D) \left(\frac{Q}{R}\right)$$

$$\text{Maximum ideal queue length} = \sum_{i=0}^{n} \left(1-\frac{D}{R}\right) Q \quad (4)$$

If we know the cost of each processing time units is c_o (cost per processing unit)

$$\text{Maximum ideal queue length} = \sum_{i=0}^{n} \left(1-\frac{D}{R}\right) Qc_o \quad (5)$$

We find that the probabilistic model is appropriated to cloud provider whose demand for public services arrived unexpectedly. For many waiting queues the arrivals requests arrive randomly and independent of other requests, and we cannot predict when a request arrive. In this case, the Poisson probability distribution provides a good distribution of the configuration of arrival. The function of the Poisson probability gives the probability of requests that arrive within a period of time determined as follows.

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad (6)$$

Where x = the number of arrivals requests per time period,

λ = the mean number of arrivals per time period,

e = 2.71828.

2.2.1 Single- queue -server (M / M / 1 Model)

M / M / 1 model is a single server that has unlimited queue capacity and infinite requests, in which queue, service and arrivals are Poisson distribution which arrivals and service times follow the exponential distribution. The mathematical nature of the exponential distribution is a series of simple ratios can be calculated for different performance measures based on knowledge of arrival rate and service rate. The mathematical method used to calculate the formulas for the single waiting queue model with Poisson arrivals and exponential service times. The following formulas can be used to calculate M / M/ 1 characteristic model for a single server queue with Poisson arrivals and exponential service times, where

λ = the mean number of arrivals per time period (the arrival rate)

μ = the mean number of services per time period (the service rate)

Then, the average number of jobs in the waiting queue can be calculated as the following:

$$Lq = \frac{\lambda^2}{\mu(\mu-\lambda)} \quad (7)$$

Also, we can compute the probability that an arriving unit has to wait for service

$$P_W = \frac{\lambda}{\mu} \quad (8)$$

And P_W is called server utilization factor or traffic intensity

The probability that no jobs are waiting in the system

$$P_O = 1-\frac{\lambda}{\mu} \quad (9)$$

from the above formulas (8,9) , we can generalized formula to compute the probability of n waiting jobs in the queue by:

$$P_n = \left(\frac{\lambda}{\mu}\right)^n P_O \quad (10)$$

2.2.2 Multiple- queue- server (M / M/ S Model)

Multiple servers consists of two or more servers that are assumed to be identical in terms of service capability. In the multiple servers, arriving requests wait in a single waiting queue and then move to the first available server to be served. We assume that incoming requests follow a Poisson probability distribution, the service time for each server follows an exponential probability distribution, the service rate μ is the same for each server and the incoming requests wait in a single waiting queue and then move to the first idle server for service. The following formulas can be used to compute the operating characteristics for multiple servers. This model is depicted in figure 2, where

λ = The arrival rate for the system,

μ = The service rate for each server,

k = Number of servers,

$\overline{U}(X)$ = The upper bound of waiting time for each queue q_i ,

and

$\underline{L}(X)$ = The lower bound of waiting time for each queue q_i

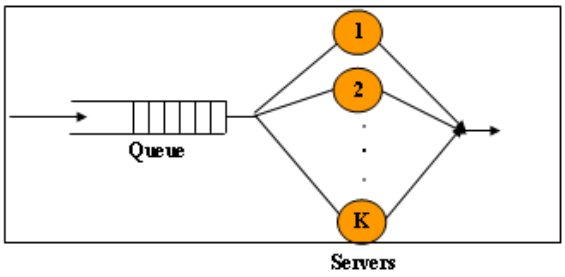

Fig. 2: Multiple queue servers, with service rate μ .

The probability that no jobs are in the system be calculated as the following:

$$p_o = \frac{1}{\sum_{n=0}^{k-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^k}{k!}\left(\frac{k\mu}{k\mu-\lambda}\right)} \qquad (11)$$

Also, we can compute the average number of jobs in the waiting queue

$$Lq = \frac{(\lambda/\mu)^k \lambda\mu}{(k-1)!(k\mu-\lambda)^2}\, p_0 \qquad (12)$$

Let the maximum number of job in the waiting queue is K, and the maximum queue length is k-1 then we can be computed Maximum number of jobs allowed in the system by computed the probability of the upper waiting time for each queue q_i as the following:

$$\bar{U}(X) = \sum_{n=0}^{k-1} \frac{(\lambda/\mu)^n}{k} + \frac{(\lambda/\mu)^k}{n!}\left(\frac{k\mu}{k\mu-\lambda}\right) \qquad (13)$$

Also we can compute the probability of the lower waiting time for each queue q_i as the following:

$$\underline{L}(X) = \sum_{n=0}^{k-1} \frac{(\lambda/\mu)^n}{k} + \left(\frac{k\mu}{k\mu-\lambda}\right) \qquad (14)$$

From the above formulas (11, 12), we can generalize formula to compute the probability of *n* waiting jobs in the queue by:

$$P_n = \frac{(\lambda/\mu)^n}{n!} P_o \qquad \text{for } n \le k \qquad (15)$$

$$P_n = \frac{(\lambda/\mu)^n}{k!k^{(n-k)}} P_o \qquad \text{for } n \succ k \qquad (16)$$

3. MODEL CONSTRUCTION

According to the analysis of the behavior of the cloud computing network with multiple servers and requests for service, the cloud can be considered as a pool of resources. Proposed model depicted in figure 3. It consists of four modules: multiple waiting queues for incoming requests, global scheduler based on SA algorithm, local schedulers and waiting queues for each local scheduler. Submit requests (R_1, R_2,…, R_n) from different sites are submitted to the Global Scheduler (GS). Each request is then transported to the local controller via the communication network and then sent to the local queue. Simulation studies to examine the effectiveness of different models within this framework are used. Suppose a model in which many users submit a request for execution of the work by any of a large number of sites. At each site, two components are placed: a global scheduler (GS), who determine where to send the jobs sent to this site, a Local Scheduler (LS), responsible for determining the order in which jobs is performed in this particular site.

3.1 Scheduling Algorithm (SA)

The goal of this algorithm is to reduce waiting time at cloud provider by calculating lower and upper waiting time for each queue as presented in Figure 4. The input of the algorithm is the multiple waiting queues and the output is the lower and upper waiting time for each queue. Also, SA algorithm begins with sampling original waiting queues into sets and assign each set to a local scheduler that compute lower and upper waiting time. These lower and upper calculated from all base local scheduler will be combined in a new decision system with low inconsistent. SA algorithm is able to compute lower and upper waiting time from waiting queues using definitions in 2.2.1 and 2.2.2. This algorithm have three steps, in the first step it assigns each one of this waiting queue to one local scheduler that work based on queuing models methodology. The second step is the core of algorithm where we compute the lower and upper waiting time for each queue. The last step start by mapping the waiting job to appropriate queue. The computation time of SA algorithm is $O(k n^2)$ for n waiting queues

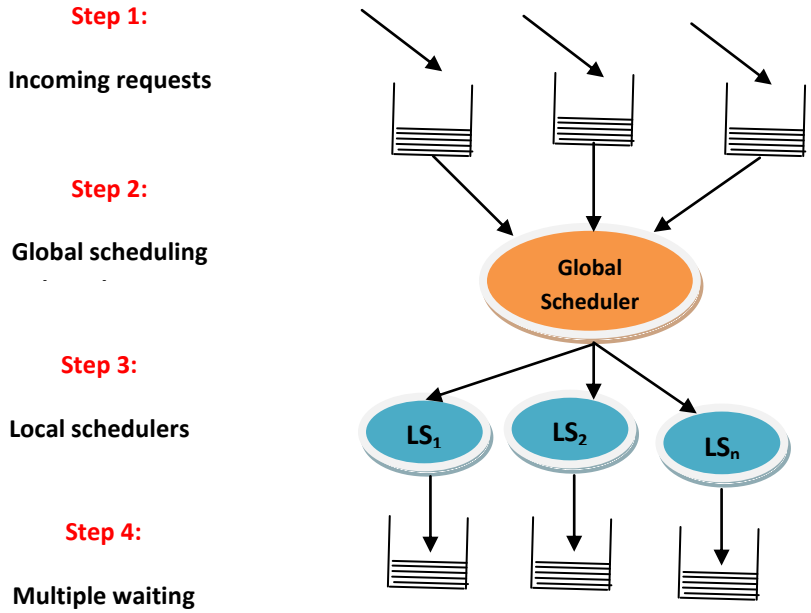

Fig. 3: Proposed Model.

Algorithm: Scheduling Algorithm (SA)

INPUT : Multiple waiting queues

OUTPUT : Mapping jobs based on queue models

//Constructing sampling waiting queues

1: For k=1 to N do

2: Create set Qi **by sampling** Q/N **// N is the number of waiting queues**

//Determine equivalence queues based on queue models

3: Let $X = \{q_1, q_2, ..., q_n\}$ **are the equivalence waiting queues**

4: Set $\underline{E}(X) = \varnothing$ **;** $\overline{E}(X) = \varnothing$

//Computing lower and upper waiting time for each waiting job

5: Set $\underline{L}(X) = \varnothing$ **;** $\overline{U}(X) = \varnothing$ **;**

6: For j = 1 to n begin

7: if $q_j \subseteq X$ **then** $\underline{L}(X) = \underline{L}(X) \cup q_j$ **;**

8: Else if $q_j \cap X = \varnothing$ **then** $\overline{U}(X) = \overline{U}(X) \cup q_j$ **.**

9: End if

10: End for

11: End for

12: $MJ = \bigcup \underline{E}(X) + \bigcap \overline{E}(X)$

Fig. 4: Scheduling Algorithm (SA) based on queuing models.

4. EXPERIMENTAL RESULTS

4.1 Simulation Settings

The proposed model based on discrete event simulation will simulate with Matlab [15, 16]. Requests enter the system randomly and form separate queues for each cloud server. The input flow of clients is of type Poisson, and that the service time distribution of the cloud servers is second order Erlang. The cloud center consists of 8 server independent that for performance issue can accept globally a limited number of requests in concurrent execution, i.e., they are allocated in a finite capacity region with the maximum number of requests. A load balancer distributes the requests among servers according to FCFS algorithm.

4.2. The discussion of the results

In this section, we perform simulations to evaluate the proposed model based on different queuing models. Table1 shows the queue length, residence time, utilization and throughput for each model based on the same constraints. We first compare the performance between the proposed model and other queuing models, in which the waiting time and utilization for waiting queues and servers are computed by proposed SA algorithm, where the arrival rates and service time for proposed model are allocated equally.

Comparison of the queue length, residence time, utilization and throughput between the proposed model based on SA algorithm and the different queuing models is shown in table 1. From table 1, we can see that the proposed model achieves much lower queue length compared to the M / M / 1 Model and M / M / S model under the same constraints. Also, the proposed model allocates a large number of waiting jobs in the computing servers, thus leading to a higher utilization. We next evaluate the system throughput between the proposed model and the queuing models in the cloud system.

We run simulation based on eight servers and the previous parameters to illustrate the performance of the cloud system. We first compare the performance between the proposed optimal model, in which the jobs for schedule and computation are allocated optimally by SA algorithm and the different queuing models, in which the jobs for schedule and computation are allocated equally. Figure 5 shows the comparison of service response time for each model.

Table 1. Comparison between proposed model and queuing models.

		Queue length [jobs]	Residence time [sec]	Utilization [%]	Throughput [jobs/sec]
M / M / 1 Model	Min	0.373	2.836	0.076	4.996
	Max	0.434	3.327	0.089	5.713
	Avg	0.403	3.082	0.082	5.331
M / M / S model	Min	0.181	1.326	0.159	7.496
	Max	0.216	1.495	0.185	8.606
	Avg	0.199	1.410	0.172	8.013
proposed model	Min	0.080	0.882	0.276	13.067
	Max	0.096	1.019	0.318	15.014
	Avg	0.088	0.950	0.297	13.973

Table 2. The response time and throughput of proposed system model

	#of jobs	System Response Time [sec]	System Throughput [job/sec]
Min	2.952	20.510	0.134
Max	3.047	21.413	0.149
Avg	3.00	20.961	0.141

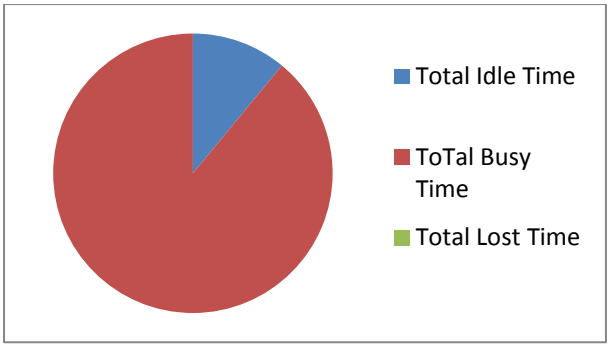

Fig. 5: Average response time for different combinations of scheduling and host utilization using queuing models.

From figure 6, we can see that the proposed model takes less response time than the different queuing models under same constraints

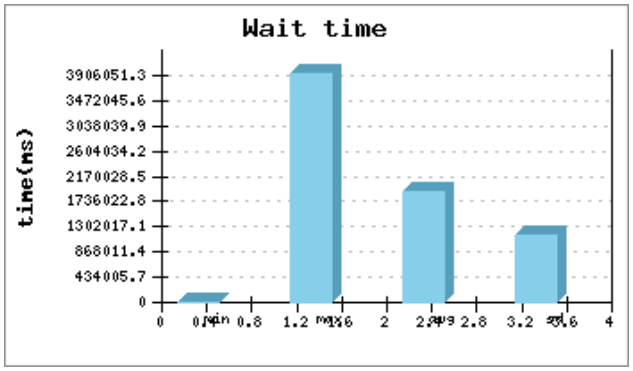

Fig. 6: Average response time for proposed model under same constraints.

Proposed model performance increased by reducing the mean queue length and waiting time. In addition, we observe that the proposed model is convenient for global scheduler in which we need to maximize the use, reducing waiting time and deal with an infinite number of applications for cloud resources. On the other hand, load balancing in global scheduler cannot be achieved by M/M/1 model that introduced a single channel for all requests.

5. CONCLUSION

In this paper, the proposed model based on queuing models. routing incoming requests to the queue with the smallest workload reduced workload, response time and the average length of the queue. These results indicate that our model increase utilization of global scheduler and decrease waiting time. The experimental results indicated that proposed model decrease waiting time at global scheduler in cloud architecture. In the future work, proposed model will use cloud computing algorithms based on parallel algorithms to decrease the time of routing end users requests.

6. REFERENCES

[1] B. Furht, A. Escalante (eds.), "Handbook of Cloud Computing", DOI 10.1007/978-1-4419- 6524-0_1,© Springer Science + Business Media , LLC 2010.

[2] Martin Randles, David Lamb, A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp. 551-556.

[3] T. Gopalakrishnan Nair, M. Vaidehi, K. Rashmi, V. Suma, "An Enhanced Scheduling Strategy to Accelerate the Business performance of the Cloud System", Proc. InConINDIA 2012, AISC 132, pp. 461-468, © Springer-Verlag Berlin Heidelberg 2012.

[4] B. Rimal, E. Choi, I. Lumb, "A taxonomy and survey of cloud computing systems," in Proc. IEEE Fifth International Joint Conference on INC, IMS and IDC, 2009, pp. 44–51.

[5] Amazon.com, "Elastic Compute Cloud (EC2)";

[6] Qiang Li, Yike Guo. "Optimization of Resource Scheduling in Cloud Computing", 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 978-0-7695-4324-6/10© IEEE, DOI 10.1109/SYNASC.2010.8.

[7] K. Mukherjee, G. Sahoo, "Development of Mathematical Model for Market-Oriented Cloud Computing", International Journal of Computer Applications (0975 – 8887), Vol. 9, No. 11, November 2010.

[8] R. Buyya, K. Sukumar "Platforms for Building and Deploying Applications for Cloud Computing", CSI Communication, pp. 6-11, 2011

[9] Shirazi, B. A., K. Krishna, 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.

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

[11] T. Helmy, A. Dekdouk " Burst Round Robin: As a Proportional-Share Scheduling Algorithm", IEEE Proceedings of the fourth IEEE-GCC Conference on towards Techno-Industrial Innovations, pp. 424-428, 11-14 November, 2007.

[12] Google App Engine. http://code.google.com/appengine/ (accessed on October 25, 2011).

[13] B. Furht, A. Escalante, "Handbook of cloud computing, Cloud computing fundamentals" written by B. Furht, Springer, 2010.

[14] L. Breuer, D. Baum "An Introduction to Queueing Theory", Springer Verlag, 2005.

[15] Integrating MATLAB, Simulink and State flow Components in a SimEvents odel: www.mathworks.com/wbnr15638

[16] Averill M. Law, W. David Kelton, *McGraw-Hill* 2000 "Simulation Modeling *and* Analysis" *(3rd Edition).*