Prioritized Service Scheme with QOS Provisioning in a Cloud Computing System

Mr. Prateek Jain
Research Scholar
JJT University, Jhunjhunu
Rajasthan
uniqueprateek10@gmail.com

Dr. Uma Shanker Pandey
Associate Professor
Delhi University (SOL)
India Delhi, India
uspandey1@gmail.com

Abstract:-Cloud computing is a compilation of existing techniques and technologies, packaged within a new infrastructure paradigm that offers improved scalability, elasticity, business agility, faster startup time, reduced management costs, and just-in-time availability of resources. It is based on the pay as you use policy and virtual servers are used in this technology. This technology is capturing the market at a rapid rate and is an advancement over the distributed computing technology. There is a scheduling issue in this technology as in case of normal scheduling the service with the more burst time blocks the service of less burst time hence we need to prioritize the service in the way that every service gets equal opportunity to execute. A priority scheme is proposed in which the prioritized customers are categorized into different priority queues. These prioritized customers have guaranteed Quality of Service (QoS) by the cloud computing system in terms of less response time. The concept of selection probability is introduced according to which the cloud metascheduler chooses the next query for execution. The priority queues are modeled as M/M/1/K/K queues and an analytical model is developed for the calculation of selection probabilities. Two algorithms are proposed for explaining the processing at the users' end and at the Cloud Computing server's end. The results obtained are validated using the numerical simulations.

Keywords: *Cloud computing, Quality of service, Prioritized queue, Mean response time.*

*****

I. Introduction

Cloud computing is complete new technique put forward from industry circle, it is the development of parallel computing, distributed computing and grid computing, and is the combination and evolution of virtualization, utility computing, Infrastructure-as-a-Service (IaaS), Platform-as-a- Service (PaaS) and Software-as-a-Service (SaaS) [2][3]. To users, cloud computing is a Pay-per-Use-On-Demand mode that can conveniently access shared IT resources through internet. Where the IT resources include network, server, storage, application, service and so on and they can be deployed with much quick and easy manner, and least management of and interactions with service providers [2][3][4]. Though Cloud Computing has made a strong footage in the various transactions but it has to face some of the challenges. One key issue and challenge to be addressed is the quality of service (QoS). So far little work has been done in the direction towards measurement of QoS provided to the end users. There are various parameters such as server delay, connection breakdown, speed etc., on which the QoS depends. Aurrecoechea et al. [6] gave a survey on QoS architecture. Barford and Crovella [7] evaluated server performance and its correlation with website workloads. Traffic intensity is an important parameter which affects the server delay. Bhatti and Freidrich [8] studied server QoS which is a key component in delivering end to end predictable, stable and tired services to the customers. They demonstrated that through classification, admission control and scheduling, they can support distinct performance levels for different users and maintain predictable performance. Load balancing in server helps in increasing the speed and efficiency of the Cloud Computing system. Researchers like Colajanni et al. [6][7], Menansce et al. [8], and Cherkasova and Phaal [9] addressed these problems. Cherkasova and Phaal discussed how an overloaded web server can experience a severe loss of throughput. Another important aspect of performance evaluation is the performance testing studied by Avritzer and Weyuker [1]. They discussed the role of modeling in the performance testing of Cloud Computing applications. Bandwidth optimization through optimal channel allocation is another important performance parameter discussed by Lin et al. [13]. Recently Awan and Singh [14] studied the performance of Cloud Computing system in wireless cellular network. Provost and Sundarajan [15] discussed various generic models of the structure of complex networks, and the probabilistic dependencies among networked entities. Job scheduling strategy is an important component/task/element for multiple job scheduling in cloud based system. It can provide a reasonable & efficient scheduling decision for submission of jobs by the end users. A FCFS job scheduling mechanism is provided in which execution of job is the basis of arrival of jobs & is the most basic, simple and common method of job scheduling in cloud as well as grid computing systems. According to the time order of jobs submitted by users, FCFS selects always the first job from the waiting job queue to map the appropriate computing resources and execute it. In this paper we have discussed a prioritized based model to provide QoS at the server processing end as well as to provide the information to the user in terms of tentative response time. Sensitivity of various parameters are analyzed which can be utilized to improve the performance of the system. Scheduling also plays a significant role at the backend of cloud computing hence a differentiated job scheduling algorithm was proposed. How to use Cloud computing resources efficiently and gain the maximum profits with job scheduling system is one of the cloud computing service providers' ultimate goals. Luqun [16] analysed the differentiated QoS requirements of Cloud computing resources. The non-pre-emptive priority M/G/1 queuing model for the jobs was used. Then, considering Cloud computing service providers' destination which is to gain the maximum profits by offering Cloud computing resources, we built the system cost function for this queuing model was developed. In order to improve the system resource utilization rate and shorten the response time, the author has presented PB-FCFS algorithm which is a task scheduling algorithm based on FCFS and backfilling strategy for cloud computing. The PB-FCFS algorithm combines backfilling and priority scheduling strategy, as well considers the resource recycling at the time of task accomplished [17]. The author proposed a cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm. The main contribution of author is to balance the entire system load while trying to minimizing the make span of a given tasks set [18]. In order to predict the job execution time in cloud environment the scheduler must be dynamic so the parallel job scheduling strategies like EASY, conservative backfill algorithms are failed to fill the resource gap fully. In combinational backfill algorithm (CBA) small jobs are getting high priority. To achieve QOS in cloud environment the author has proposed an improved backfill algorithm using balanced spiral (BS) method [19].The paper is divided into various sections. Section 2 gives the brief description of the model and its importance. Section 3 presents the mathematical formulation of the model along with various notations and assumptions. Section 4 provides numerical illustrations with sensitivity analysis and finally conclusion is drawn in section 5.

II. Model Description

We consider a cloud computing system which can be utilized by end users in retail shopping or the organization in business to business scenario. These systems can be maintained by the organization themselves or can be hired for information services. With increasing demand of customers, the systems are engineered as per their preference. For e.g., in a retail environment a customer visits a cloud computing website to purchase a product category or an item. Once a user selects a particular category, request is send to the web server which in turn connects to the Cloud Computing server to initiate the process of command execution or search. Cloud computing server run the query process and selects the data from the database server. It then fetches the data and provides it to the web server, which in turn is displayed to the customer. Thus the transaction process in the website includes a request or query, which is processed by the Cloud Computing server and the data storage area i.e. database server as depicted in the simplified architecture in Fig 1.

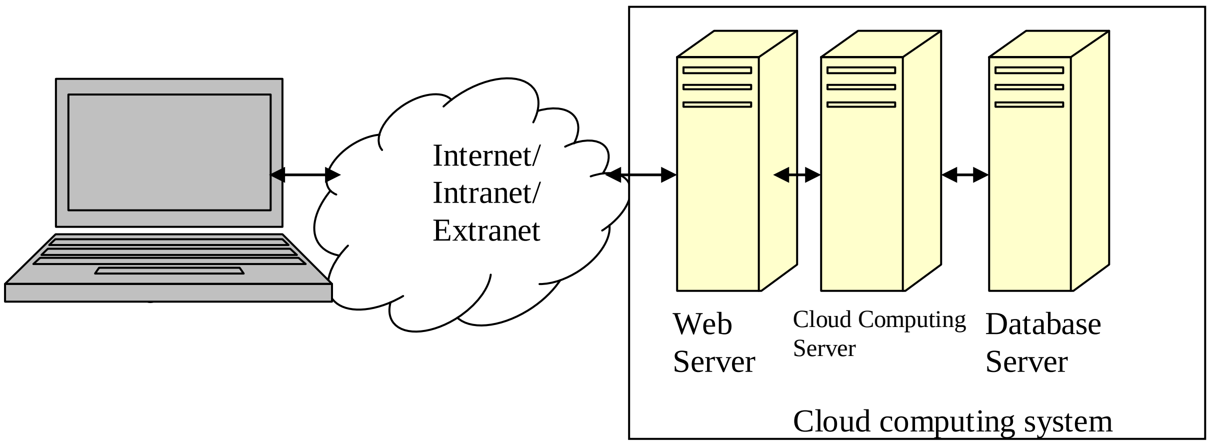

Fig. 1: Cloud computing system architecture

The figure depicts various components of a cloud computing system such as web server, cloud computing server and the database server. A web server serves web request with the help of a number of threads or processes (as in case of

UNIX). Each thread performs a user request. One can also say that as soon as the customer enters the site and requests information, a session is allocated to him by the web server by means of thread. After getting the request from the user and processing it, web server returns the result to user. Cloud computing server in turn processes the query or request with the help of number of commands. The commands include adding items to shopping cart, processing, displaying specific item or the product page, updating the database, etc. To process the commands, cloud computing requires data repository where all the information regarding the cloud computing system are stored. Due to the increasing demand for web services a lot of companies and organizations are providing high end Cloud Computing solutions. The scenario of cloud computing has thus become very competitive hence the web surfers look forward to customized and prioritized services where they can demand QoS.

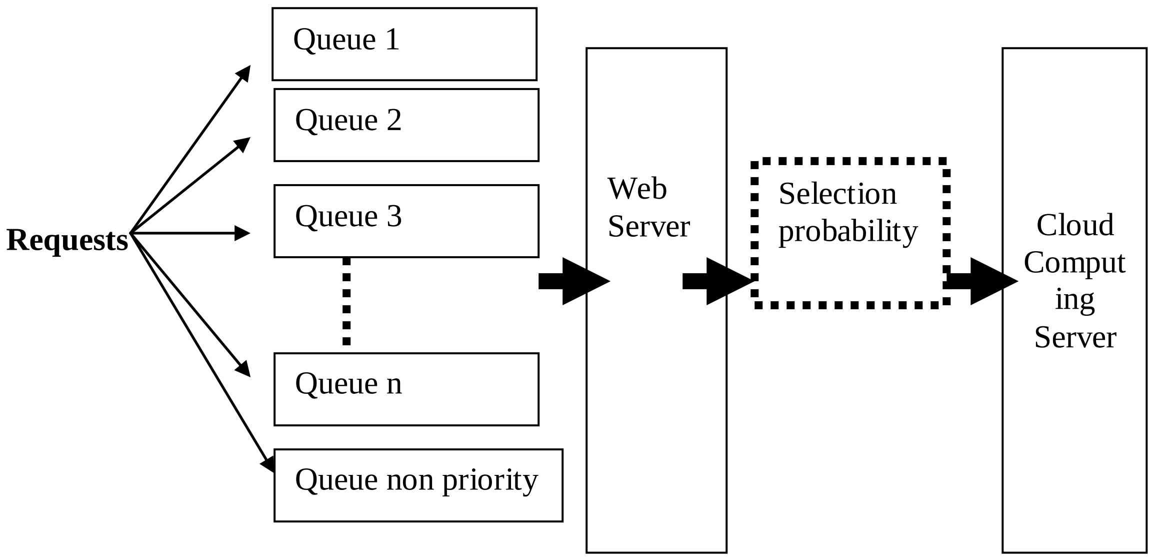

Fig. 2: Selection probability mechanism for the Cloud Computing server

We develop a model whereby user can demand QoS for his request. When a user comes to the website, he opts for a particular request/task (see Fig. 2). The system provides him the option for prioritized/non-prioritized service. A prioritized service guarantees the QoS in terms of lower response time. Every request/task has a pre-specified priority in the system. If the user opts for the prioritized service, system notifies the user whether the user can be guaranteed the QoS or not. In case the system can provide QoS then he/she will be served with that pre-specified priority as per the availability of server otherwise he/she is served as a non-prioritized user after notification. In our model, we have categorized the requests into k different priority sets. A separate queue for each priority set (queues 1, 2... k) and one queue for the non-prioritized users (queue NP) is formed. Hence, there are total k+1 queues in the Cloud Computing system. At the server side, the requests are fetched for processing from the k priority queues according to a particular selection probability p_i (i=1,2,…,k) which helps in ensuring QoS.

III. Mathematical Formulation

For the mathematical formulation, we assume that the requests in each of the k+1 queues originate from a finite population source of size M_i (i=1,2,…,k+1) and are distributed in a Poission fashion with rate λ. The capacities C_i (i=1,2,..k+1) of each of the k+1 queues are also supposed to be finite. The command execution time D of the server is a random variable with exponential distribution. Let R_i (i=1,2,…,k+1) denote the mean response time of the users of the i^{th} queue. Let us consider a particular queue r with selection probability $p_r = p$. Let $M_r = C_r$ be equal to K. Then the queue can be modeled as M/M/1/K/K where the mean command execution rate equals p/D and the mean response time is R_r=R. The steady state probability that there are n requests in the queueing system can be obtained using the product type solution as

$$P_n = P_0 \prod_{i=0}^{n-1} \frac{\lambda(K-i)}{p/D}, \quad 0 \le n \le K \tag{1}$$

$$\text{or } P_n = P_0\left(\frac{D\lambda}{p}\right)^n \frac{K!}{(K-n)!} \qquad (2)$$

Where P0 is the probability that there are no requests in the system i.e. the server is idle and is obtained using normalizing condition

$\sum_{n=0}^{K} p_n = 1$, as

$$P_0 = \frac{1}{\sum_{n=0}^{K}\left(\frac{D\lambda}{p}\right)^n \frac{K!}{(K-n)!}} \qquad (3)$$

The server utilization U is 1-P_0 and the average rate of request completion E[T] is (1-P_0)p/D = Up/D. Now, on an average, a request is generated by a user in R + 1/λ seconds. Thus, the average request-generation rate of the web-server is K/(R + 1/λ). In the steady state, the request generation and completion (execution) rates must be equal. Thus we have,

$$\frac{K}{R+1/\lambda} = \frac{Up}{D} \qquad (4)$$

$$\text{or} \qquad p = \frac{\lambda KD}{U(\lambda R+1)} \qquad (5)$$

When a new user enters the website and requests a specified priority, the appropriate selection probability is recalculated and the inequality $\sum_{i=1}^{k} p_i \le 1$ is checked. If it is true after the selection probability recalculation, the user is guaranteed appropriate QoS (response time) during the whole session. If the inequality becomes false, after the selection probability recalculation, he will not be guaranteed QoS and he enters the website as a non priority user. The processing at the user's side is discussed below in algorithm 1.

Algorithm 1

```
when a new user arrives and makes a request
    if the user is a prioritized user then
        the appropriate priority related to the user's request is obtained;
        the length of the corresponding priority queue is updated and its capacity is checked;
        if the queue can still accommodate more users
            the selection probabilities p_i (i=1,2,…k) are recalculated;
            if $\sum_{i=1}^{k} p_i \le 1$
                the user is guaranteed the appropriate response time;
            else
                the user is served as a non-prioritized user;
            endif
        else
            the user is served as a non-prioritized user;
        endif
    else
        the user is served as a non-prioritized user;
    endif
end
```

At the server side, when the command execution is over, the next query has to be fetched form one of the k+1 queues. In order to make a decision from what queue the next query will be chosen, the selection probabilities are used. The queue having maximum choice probability is chosen for fetching the next query. The processing at the server's side is discussed below in algorithm 2.

Algorithm 2

```
When the command execution is over
    the length of the corresponding queue is updated;

    the selection probabilities are recalculated;
    the queue with maximum selection probability is selected and the query form this selected queue is executed;
```

the length of the queue is decremented by 1;

End

IV. Numerical Illustration

Simulation and sensitivity analysis, based on the algorithms proposed in the previous section are performed to validate the authenticity of the analytical results. For the illustration purpose, 5 priority queues are assumed i.e. k=5 and one non priority queue is considered. The response times, and the capacities and population sizes of the queues are respectively fixed as

$R_1=5, R_2=10, R_3=15, R_4=20, R_5=25, R_6=30$ and $C_1=M_1=30, C_2=M_2=25, C_3=M_3=20, C_4=M_4=15, C_5=M_5=10, C_6=M_6=50$.

The model is simulated in Matlab 6.0 for 50 users by taking $\lambda=0.5$ and D=0.005. Table 1 exhibits the simulation results wherein the types of priorities demanded by the users are generated randomly. When a user demands a service of a particular priority, then the system checks whether it can provide the required QoS or not. If the QoS can be guaranteed, then 'Y' is shown which denotes response for acceptance of QoS provisioning by the server otherwise, priority service denial response is given to the user which is displayed by 'N'. At that particular instance priority queue served is also shown based on the selection probability computed by the server. We can see that till the 22nd user, priority 5 can be provided to the user but when 25th user opts for the same priority arrives, the server gives the priority service denial response due to capacity constraint. After some time, since the requests from the system are continuously served and the queue length decreases, users opting for priority 5 can be again served as shown in the table for the 50th user. Sensitivity analysis for various parameters is also carried out to validate the analytical model. Fig. 3 shows the deviation in the total number of users requesting various priority/ non priority services and the actual number of priority/ non priority requests served by taking 500 and 1000 users in the system. Graph reveals that for both the instances, the deviation in the total number of requests vs. served priorities is very less. This holds that the algorithm works for a large number of users without any deviation. Fig. 4 shows the effect on selection probabilities by varying the response time for different values of M for a particular queue. As the response time increases, the selection probability decreases which is quite obvious since greater response time indicates that the service can be delayed. This further means that the particular queue can be selected later i.e. a decreased selection probability. Further, selection probability for higher population size is greater as compared to the lower population size. Graphs in fig. 5 reveal the effect of varying λ on the selection probability for different values of D for a particular queue by assuming R=10 and M=C=100. As the arrival rate increases the selection probability of the queue increases. Also the selection probability increases with the command execution time D of the server.

V. Conclusion

In this paper we have proposed a cloud computing model to handle the prioritized and non prioritized requests of users. Quality of service is guaranteed to the prioritized requests after checking whether the server is capable of serving the request in priority at that instance or not. If the QoS cannot be guaranteed the user is intimated of that and he is served as a non prioritized user. A queueing model is developed for QoS provisioning based on the selection probability mechanism. The algorithms for the user's side priority validation and the server's side computation of selection probability to serve the incoming requests, so as to maintain QoS, are developed and illustrated numerically. The results reveal that the system can work efficiently as a large user base and is also a demand based system which ensures QoS. The model developed can be easily deployed in the Cloud Computing frontend and backend solutions. The system will enable the organization to provide services at a pre-specified response time and thus increasing the system reliability. Extension of our work can be done by developing a cloud computing multi server queueing model.

Table 1: Simulation results for 50 users

User	Priority required	QoS guaranteed	p_1	p_2	p_3	p_4	p_5	Priority served
1	3	Y	0.0036	0.0021	0.0018	0.0011	0.0009	1
2	3	Y	0.0029	0.0021	0.0021	0.0011	0.0009	1
3	2	Y	0.0021	0.0025	0.0021	0.0011	0.0009	2
4	3	Y	0.0021	0.0021	0.0024	0.0011	0.0009	3
5	3	Y	0.0021	0.0021	0.0024	0.0011	0.0009	3
6	4	Y	0.0021	0.0021	0.0021	0.0014	0.0009	1

7	1	Y	0.0021	0.0021	0.0021	0.0014	0.0009	1
8	1	Y	0.0021	0.0021	0.0021	0.0014	0.0009	1
9	3	Y	0.0014	0.0021	0.0024	0.0014	0.0009	3
10	6	Y	0.0014	0.0021	0.0021	0.0014	0.0009	2
11	6	Y	0.0014	0.0017	0.0021	0.0014	0.0009	3
12	2	Y	0.0014	0.0021	0.0018	0.0014	0.0009	2
13	1	Y	0.0021	0.0017	0.0018	0.0014	0.0009	1
14	6	Y	0.0014	0.0017	0.0018	0.0014	0.0009	3
15	2	Y	0.0014	0.0021	0.0015	0.0014	0.0009	2
16	4	Y	0.0014	0.0017	0.0015	0.0016	0.0009	2
17	6	N	0.0014	0.0017	0.0015	0.0016	0.0009	2
18	4	Y	0.0014	0.0008	0.0015	0.0018	0.0009	4
19	6	N	0.0014	0.0008	0.0015	0.0018	0.0009	4
20	1	Y	0.0021	0.0008	0.0015	0.0014	0.0009	1
21	1	Y	0.0021	0.0008	0.0015	0.0014	0.0009	1
22	5	Y	0.0021	0.0008	0.0015	0.0014	0.0009	1
23	3	Y	0.0007	0.0008	0.0018	0.0014	0.0011	3
24	6	N	0.0007	0.0008	0.0018	0.0014	0.0011	3
25	5	N	0.0007	0.0008	0.0018	0.0014	0.0011	3
26	5	N	0.0007	0.0008	0.0018	0.0014	0.0011	3
27	3	Y	0.0007	0.0008	0.0009	0.0014	0.0015	6
28	1	Y	0.0014	0.0008	0.0009	0.0014	0.0015	6
29	1	Y	0.0021	0.0008	0.0009	0.0014	0.0015	1
30	2	Y	0.0014	0.0013	0.0009	0.0014	0.0015	5
31	3	Y	0.0014	0.0013	0.0012	0.0014	0.0013	1
32	6	N	0.0014	0.0013	0.0012	0.0014	0.0013	1
33	3	Y	0.0000	0.0013	0.0015	0.0014	0.0013	6
34	5	N	0.0000	0.0013	0.0015	0.0014	0.0013	6
35	3	Y	0.0000	0.0013	0.0018	0.0014	0.0015	3
36	3	Y	0.0000	0.0013	0.0018	0.0014	0.0015	3
37	3	Y	0.0000	0.0013	0.0018	0.0014	0.0015	3
38	3	Y	0.0000	0.0013	0.0018	0.0014	0.0015	3
39	6	N	0.0000	0.0013	0.0018	0.0014	0.0015	3
40	1	Y	0.0007	0.0013	0.0012	0.0014	0.0015	5
41	2	Y	0.0007	0.0017	0.0012	0.0014	0.0013	2
42	1	Y	0.0014	0.0013	0.0012	0.0014	0.0013	1
43	5	N	0.0014	0.0013	0.0012	0.0014	0.0013	1
44	4	Y	0.0000	0.0013	0.0012	0.0016	0.0015	4
45	6	Y	0.0000	0.0013	0.0012	0.0016	0.0015	4
46	4	Y	0.0000	0.0013	0.0012	0.0014	0.0015	6
47	3	Y	0.0000	0.0013	0.0015	0.0014	0.0015	5
48	2	Y	0.0000	0.0017	0.0015	0.0014	0.0013	2
49	4	Y	0.0000	0.0013	0.0015	0.0016	0.0013	4
50	5	Y	0.0000	0.0013	0.0015	0.0016	0.0013	4

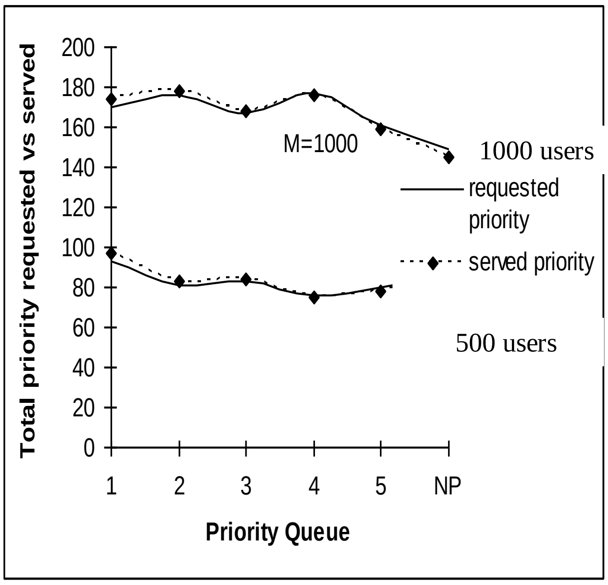

Fig. 3: Priorities requested vs. priorities served

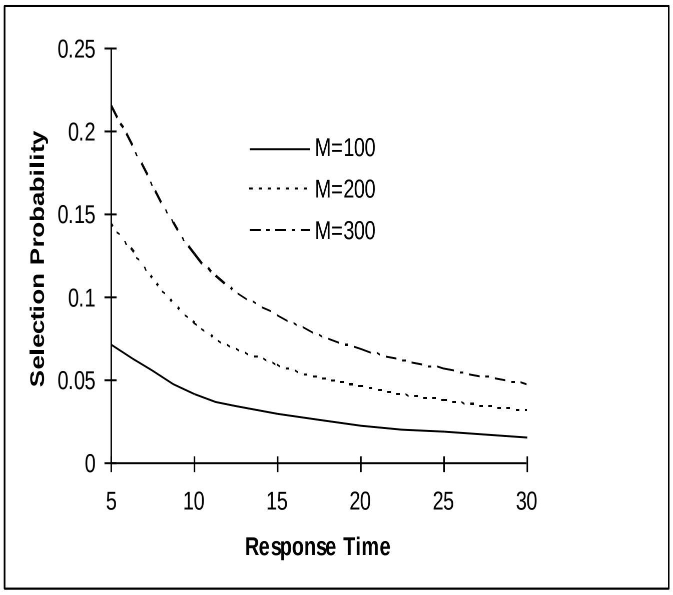

Fig. 4: Effect of response time (R) on selection probability (p)

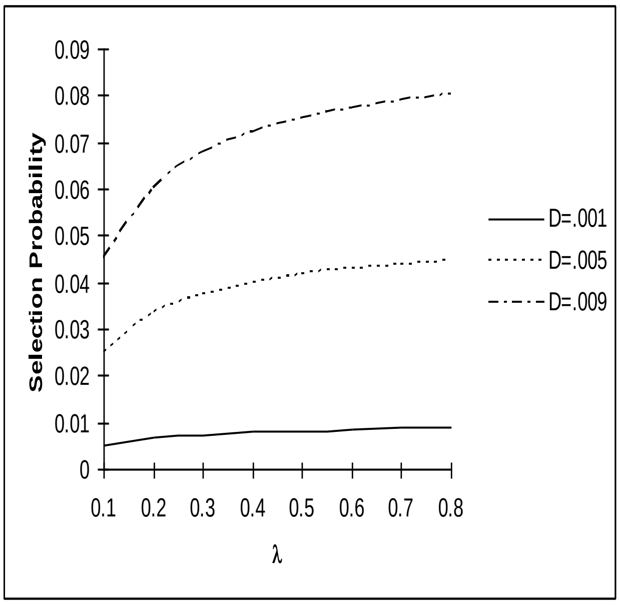

Fig. 5: Effect of arrival rate (λ) on selection probability (p)

REFERENCES

[1] Avritzer, A. and Weyuker, E. J.: The role of modeling in the performance testing of Cloud Computing applications, IEEE Trans. on Soft. Engg. , vol. 30, no. 12, pp. 1072-1083, 2004.

[2] D. Nurmi, R. Wolski, etc., “The Eucalyptus Open-source Cloud computing System,” in Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, 2009, 124-131.

[3] B. Sotomayor, K. Keahey, I. Foster. Combining Batch execution and Leasing Using Virtual Machines, HPDC 2008, Boston, MA, 2008, 1-9.

[4] K. Keahey and T. Freeman, “Science Clouds: Early Experiences in Cloud Computing for Scientific Applications,” in proceedings of Cloud Computing and Its Applications 2008, Chicago, IL. 2008.

[5] Awan I. and Singh S.: Performance evaluation of Cloud Computing requests in wireless cellular networks, Information and Software Technology, vol. 48, no. 6, pp. 393-401, June 2006.

[6] Aurrecoechea, C., Campbell, A. and Hauw, L.: A survey of QoS architectures, ACM/Springer-Verlag Multimedia Systems J., special issue on QoS architecture, Vol. 6, no. 3, May 1998.

[7] Barford, P. and Crowell, M.: Generating representative web workloads for network and server performance evaluation, Proc. ACM SIGMETRICS '98, pp. 151-160, July 1998.

[8] Bhatti, N. and Friedrich, R.: Web server support for tiered services, IEEE Network J., vol. 13, no. 5, Sept./Oct. 1999.

[9] Cherkasova, L. and Phaal P.: Session-based admission control: A mechanism for peak load management of commercial web site, IEEE Trans. On Computers, vol. 51, no. 6, pp. 669-685, 2002.

[10] Cherkasova, L. and Phaal, P.: Session based admission control: A mechanism for improving performance of commercial web sites, Proc. Seventh Int'l Workshop Quality of Service, 31 May-4 June 1999.

[11] Colajanni, M., Yu, P., Cardellini, V., Papazoglou, M., Takizawa, M., Cramer, B. and Chanson, S.: Dynamic load balancing in geographically distributed heterogeneous web servers, Proc. 18th Int'l Conf. Distributed Computing Systems, May 1998.

[12] Jann, J., Burgula, R.S., Dubey, N., Patnaik, P.: End-to-end performance of commercial applications in the face of changing hardware, ACM SIGOPS Operating System Review, vol. 42, no. 1, pp. 13-20, 2008.

[13] Lin, M., Liu, Z., Xia, C.H. and Zhang, L.: Optimal capacity allocation for web systems with end-to-end delay guarantees, Performance Evaluation, vol. 62, no. 1-4, pp. 400-416, Oct. 2005.

[14] Menansce, D., Almeida, V., Fonseca, R. and Mendes, M.: Resource management policies for Cloud Computing servers, Proc. Second Workshop Internet Server Performance, June 1999.

[15] Provost, F., Sundarajan, A.: Modeling complex networks for electronic commerce, Proceedings of the 8th ACM Conference on Electronic Commerce, pp. 368-368, 2007.

[16] HUANG Qi-yi, HUANG Ting-lei, An Optimistic Job Scheduling Strategy based on QoS for Cloud Computing, 2010.

[17] PB-FCFS--A Task Scheduling Algorithm Based on FCFS and Backfilling Strategy for Grid Computing, Hong nANG, IEEE Conference, 2009.

[18] Kun Li e.t Al, Cloud Task scheduling based on Load Balancing Ant Colony Optimization, Sixth Annual China Grid Conference, 2011.

[19] Suresh.A et al, Improving scheduling of backfill algorithms using balanced spiral method for cloud Metascheduler, International Conference on Recent Trends in Information Technology, IEEE, 2011.