Prioritized Service Scheme with QoS Provisioning in a Cloud Computing System

Vidushi Sharma[1,*], Kriti Priya Gupta[2]

[1]School of ICT, Gautam Buddha University, India
[2]Symbiosis Centre for Management Studies NOIDA, Faculty of Management Studies, Symbiosis International University, India

**Corresponding email: vidushi@gbu.ac.in,svidushee@gmail.com*

Abstract

A priority scheme is proposed in which the prioritized customers get guaranteed Quality of Service (QoS) by the cloud computing system in terms of lesser response time. The concept of selection probability is introduced according to which the cloud metascheduler chooses the next query for execution. The prioritized customers are categorized into different priority queues which 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

1. INTRODUCTION

Cloud computing is complete the development in 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) [1,2]. To users, cloud computing is a Pay-per-Use-On-Demand mode that can conveniently access shared Information Technology (IT) resources through internet. The IT resources may include network, server, storage, application, service etc. which can be deployed in an easy manner and with least interaction with service providers [1,2,3]. 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. (1998) gave a survey on QoS architecture[4]. Barford and Crovella (1998) evaluated server performance and its correlation with website workloads [5]. Traffic intensity is an important parameter which affects the server delay. Bhatti and Freidrich (1999) studied server QoS which is a key component in delivering end to end predictable, stable and tired services to the customers [6]. 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 Aurrecoechea et al. (1998), Barford and Crowell (1998), and Menansce et al. (1999) addressed these problems [4,5,7]. Cherkasova and Phaal (2002) discussed how an overloaded web server can experience a severe loss of throughput [8]. Another important aspect of performance evaluation is the performance testing studied by Avritzer and Weyuker (2004) [9]. 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. (2005) [10]. Awan and Singh (2006) studied the performance of cloud computing system in wireless cellular network [11]. Provost and Sundarajan (2007) discussed various generic models of the structure of complex networks, and the probabilistic dependencies among networked entities [12]. Luqun (2010) analysed the differentiated QoS requirements of cloud computing resources. Kun Li et al. (2011) proposed a cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm [13]. Suresh et al. (2011) proposed an improved backfill algorithm using balanced spiral (BS) method to achieve QOS in cloud environment [14].

Scheduling plays a significant role at the backend of cloud computing. 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. In this paper, a differentiated job scheduling algorithm is proposed. 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. Sensitivities of various parameters are analyzed which can be utilized to improve the performance of the system. The paper is divided into 5 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.

2. MODEL DESCRIPTION

We consider a cloud computing environment, which consists of many datacenters. Datacenter is a collection of many resources viz. hardware, software, infrastructure and platform etc. Each datacenter has one or more Virtual Machines (VM) which in turn have one or more processing elements. Metascheduler maps the jobs/ requests of the user with the VM, where the jobs are scheduled to the processing element by a local scheduler. Figure 1 depicts the cloud metascheduler architecture. Three main tasks are involved in this architecture:

1. The request for the completion of the task is submitted to the metascheduler.
2. Metascheduler maps and schedules the job with the cloud cluster.
3. After scheduling the jobs are handed to the VM, which has processing elements . Local scheduler schedules jobs to the processing elements

Job scheduling is a challenging task in cloud computing systems and needs to be managed with utmost importance. The aim of job scheduling systems in cloud computing is to ensure the QoS. The two types of schedulers are available in a cloud are global metascheduler and local scheduler. Local scheduler focuses on determining how the processes reside on a single CPU and are allocated and executed whereas the global metascheduler interacts with the user and schedules jobs to VM. It uses information about the system to allocate processes among the clusters. It is impossible to predict the job execution time in cloud, hence scheduler need to be dynamic. In our model we have consider the above cloud computing architecture and proposed two algorithms to make the metascheduler dynamic providing the guaranteed QoS in terms of response time to the user. All the tasks received by the metascheduler and the decision for scheduling is made by querying the cloud about the time required to be served. This is done by finding out the length of the queue i.e. the number of requests pending in the category requested. Based on the number of requests lying in the category which user requested, selection probability is computed and the decision whether the customer can be served with that guaranteed response time is made. If the customer cannot be served then he can be given an option of being served as a non prioritized customer. Therefore, to predict the job execution time, the scheduler finds out the selection probability based on the type of request.

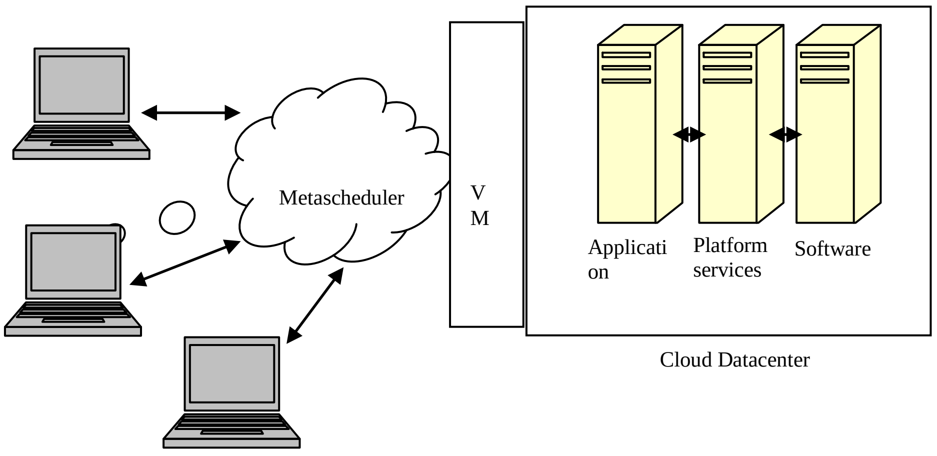

Figure 1: Cloud computing system architecture

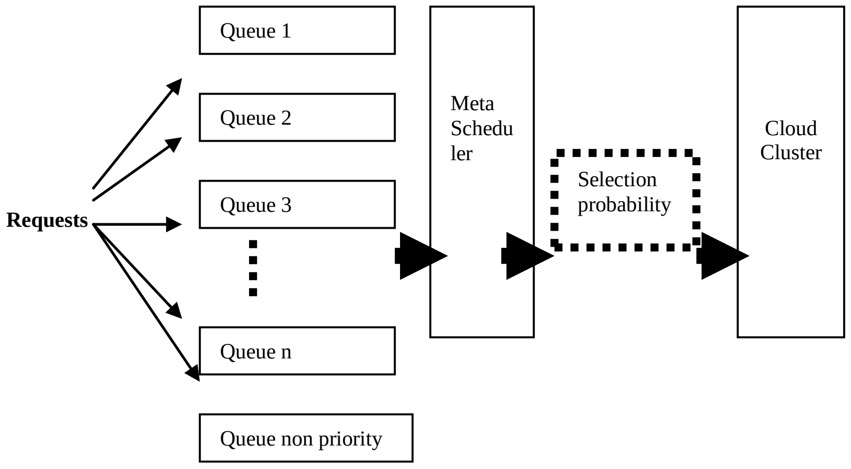

Figure 2: Selection probability mechanism for the Cloud Computing server

In this model user can demand QoS for his request. When a user comes to the website, he opts for a particular request/task (see Figure 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.

3. 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

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

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

where P_0 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

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

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 + $1/\lambda$ seconds. Thus, the average request-generation rate of the web-server is K/(R + $1/\lambda$). In the steady state, the request generation and completion (execution) rates must be equal. Thus we have,

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

or $$p = \frac{\lambda K D}{U(\lambda R + 1)} \qquad \ldots(5)$$

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
                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
```

4. NUMERICAL ILLUSTRATIONS

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
$C_1=M_1=30, C_2=M_2=25, C_3=M_3=20, C_4=M_4=15,$

$C_5=M_5=10, C_6=M_6=50$.

Sensitivity analysis for various parameters is carried out to validate the analytical model. Figure 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. The figure 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.

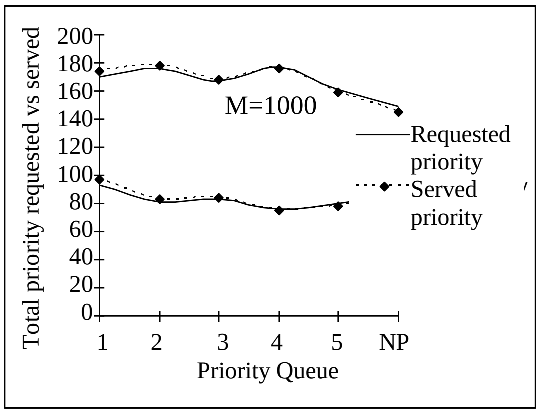

Figure 3: Priorities requested vs. priorities served

Figure 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.

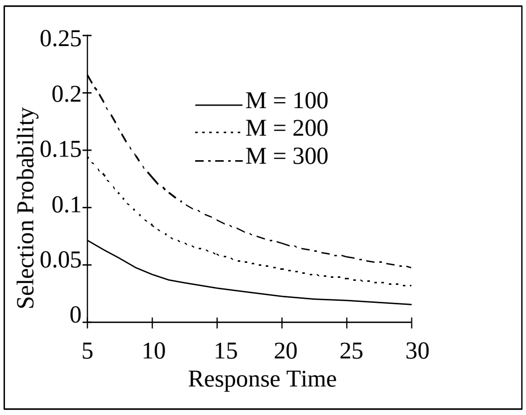

Figure 4: Effect of response time (R) on selection probability (p)

Figure 5 reveals 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.

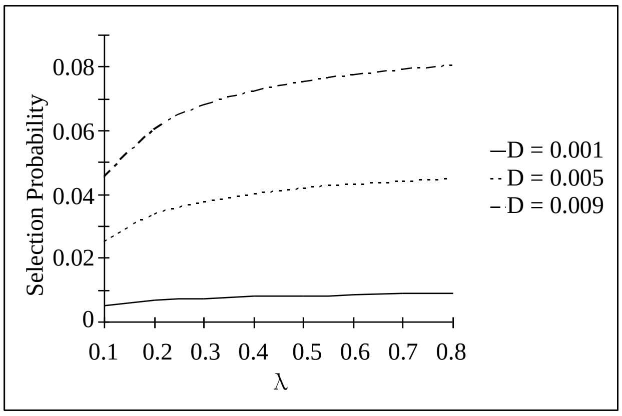

Figure 5: Effect of arrival rate (λ) on selection probability (p)

5. 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.

REFERENCES

[1] Wolski, R. , Grzegorczyk, C., Obertelli, G. , Soman, S. ,Youseff, L., Zagorodnov, D.(2009). The eucalyptus open-source cloud computing system, In Proceedings of *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai*, 124-131.

[2] Sotomayor, B., Keahey, K., Foster, I. (2008). Combining batch execution and leasing using virtual machines, *HPDC 2008, Boston, MA,* 1-9.

[3] Keahey, K. and Freeman, T. (2008). Science Clouds: early experiences in cloud computing for scientific applications, In Proceedings of *First workshop on Cloud Computing and its Applications*, Chicago, IL.

[4] Aurrecoechea, C., Campbell, A. and Hauw, L. (1998). A survey of QoS architectures, *ACM/Springer-Verlag Multimedia Systems J.*, special issue on QoS architecture, 6(3),138-151.

[5] Barford, P. and Crowell, M. (1998). Generating representative web workloads for network and server performance evaluation, In Proceedings of *ACM SIGMETRICS '98*, 151-160.

[6] Bhatti, N. and Friedrich, R. (1999). Web server support for tiered services, *IEEE Network J.*, 13 (5), 1186-1200.

[7] Menansce, D., Almeida, V., Fonseca, R. and Mendes, M. (1999). Resource management policies for Cloud Computing servers, In Proceedings of *Second Workshop Internet Server Performance*, 295-296.

[8] Cherkasova, L. and Phaal P. (2002). Session-based admission control: A mechanism for peak load management of commercial web site, *IEEE Trans. On Comput*ers, 51 (6), 669-685.

[9] Avritzer, A. and Weyuker, E. J.(2004). The role of modeling in the performance testing of cloud computing applications, *IEEE Trans. on Soft. Engg.* , 30(12), 1072-1083.

[10] Lin, M., Liu, Z., Xia, C.H. and Zhang, L. (2005). Optimal capacity allocation for web systems with end-to-end delay guarantees, *Performance Evaluation*, 62 (1-4), 400-416.

[11] Awan, I. and Singh, S. (2006). Performance evaluation of cloud computing requests in wireless cellular networks, *Information and Software Technology,* 48 (6), 393-401.

[12] Provost, F., Sundarajan, A. (2007). Modeling complex networks for electronic commerce, In Proceedings of *8th ACM Conference on Electronic Commerce,* 368-368.

[13] Kun Li, Gaochao, X. Guangyu Z. Yushuang, D., Wang, D. (2011). Cloud Task scheduling based on Load Balancing Ant Colony Optimization, In Proceedings of *Sixth Annual China Grid Conference*, 3-9.

[14] Suresh, A. Vijayakarthick, P. (2011). Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler, In Proceedings of *International Conference on Recent Trends in Information Technology, IEEE Explore*, 624-627.