International Journal of Computer Applications Technology and Research
Volume 2– Issue 5, 595 - 600, 2013, ISSN: 2319–8656

QoS Driven Task Scheduling in Cloud Computing

Sonal Dubey
NITTTR,
Bhopal, India

Sanjay Agrawal
NITTTR,
Bhopal, India

Abstract: Cloud computing systems promise to offer pay per use, on demand computing services to users worldwide. Recently, there has been a dramatic increase in the demand for delivering services to a large number of users, so they need to offer differentiated services to users and meet their expected quality requirements. Most of scheduling schemes proceeding nowadays have no QoS (Quality of Service) differentiation, which is necessary for Cloud Computing service operation. As a cloud must provide services to many users at the same time and different users have different QoS requirements, the scheduling schemes should be developed having different QoS requirements. So, this paper explores various methods of task scheduling done in cloud computing. Real-time applications play a significant role in cloud environment. We have examine the particular scheduling algorithms for real-time tasks, that is, priority-based strategies.The purpose of this paper is to discuss the fixed priority preemptive task scheduling algorithms in cloud computing for improving the QoS parameters.

Keywords: Cloud Computing, QoS, RMS, DMS, UB Test.

1. INTRODUCTION

Cloud computing is the rising technology that delivers many forms of resources as services, mainly over the internet. It permits customers to use applications without deployment and access the required files at any computer using internet [3]. Cloud Computing allows on-demand resource provisioning. It is the convergence of several concepts such as grid, distributed application design, virtualization and enterprise IT management. It enables a more flexible approach for deploying and scaling applications [2].

The Task management is the key role in cloud computing. Task scheduling problems are primary which relate to the efficiency of the whole cloud computing facilities. Because of different QoS parameters such as CPU speed, CPU utilization, turnaround time, throughput, waiting time etc., task scheduling in cloud computing is different from conventional distributed computing environment. The demand for resources fluctuates dynamically so scheduling of resources is a difficult task. Task scheduling based on QoS parameters is necessary for efficient resource utilization and for satisfying user requirement.

Scheduling in cloud computing can be categorized into three stages namely–

- Resource discovering and filtering – Data centre Broker discovers the resources present in the network system and collects status information related to them.
- Resource selection – Target resource is selected based on certain parameters of task and resource. This is key stage.
- Task submission -Task is submitted to resource selected.

The goal of scheduling algorithms in distributed systems is to schedule jobs to the flexible resources in accordance with flexible time, which includes finding out a proper sequence in which jobs can be executed under transaction logic constraints. The main advantage of task scheduling algorithm is to achieve a high performance computing and the best system throughput.

Here, we consider the following terms for our understanding:

- Task: ti
- Virtual machine: mj
- Time when task ti arrives: ci
- Time when machine mj is available: aj
- Execution time for ti on mj: eij
- Time when the execution of ti is finished on mj, cij=aj + eij: cij
- Maximum value of cij: makespan

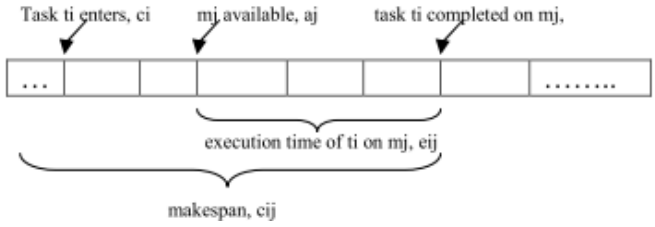

Figure 1. Task Scheduling

Cloud computing has been defined by NIST "as a model for supporting convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction". This cloud model is composed of five essential characteristics, three service models, and four deployment models. The five essential characteristics are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The three service models are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (laaS). The four deployment models are Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud [1].

A scheduling is a set of rules that determine the jobs to be executed at a particular time. This paper is concerned only with fixed-priority pre-emptive scheduling, which works as

follows. A distinct and fixed priority is assigned to each task. When a job is initiated with a priority higher than the one currently being executed, the current job is immediately interrupted and the new job is started. The remaining paper is ordered as follows. Section II describes RMS and DMS algorithms. Section III presents the related work in this area. Section IV concludes the paper.

2. FIXED PRIORITY SCHEDULING IN CLOUD COMPUTING

In real time cloud applications the cloud users and providers must have a strong service level agreement to ensure the timing of applications, and deadlines of applications. A real-time scheduler must ensure that processes meet deadlines, regardless of system load or makespan. Priority is applied to the scheduling of these periodic tasks with deadlines. Every task in priority scheduling is given a priority through some policy, so that scheduler assigns tasks to resources according to priorities.

In fixed priority scheduling the dispatcher will make sure that at any time the highest priority runnable task is essentially running. So, if we have a task with a low priority running and a high priority task arrives. The low priority task will be suspended and the high priority task will start running. If while the high priority task is running a medium priority task arrives the dispatcher will leave it unprocessed and the high priority task will carry on running and at some later time finish its computation. So the task with medium priority starts executing to finish at some later time. Only when both tasks have completed can the low priority task resume its execution. The low priority task can then carry on executing until either higher priority tasks arrive or it has finished its work. The fixed priority scheduling algorithms help in improving the QoS parameters in cloud computing environment as they have less runtime overhead, robust, optimal and easy to implement.

2.1 Related Terms

The deadline of a request for a task is defined to be the time of the next request for the same task. For a set of tasks scheduled as per some scheduling algorithm, an overflow occurs at time 't' if 't' is the deadline of an unfulfilled request. For a set of tasks, a scheduling algorithm is feasible if the tasks are scheduled so that no overflow ever occurs. We define the response time of a request for a certain task to be the time span between the request and the end of the response for the request. A critical instant is defined as an instant at which a request for that task will have the largest response time. A schedulability test is a mechanism that proofs that all deadlines are met, when scheduling with a particular algorithm. The schedulable utilization of a scheduling algorithm is defined as follows: A scheduling algorithm can feasibly schedule *any set* of periodic tasks on a processor if the total utilization of the tasks is equal to or less than the schedulable utilization of that algorithm.

2.1.1 Periodic Task Model

- A task = (C, T)

  C: worst case execution time/computing time (C<=T!)

  T: period (D=T)
- A task set: (Ci,Ti)

  All tasks are independent

The periods of tasks start simultaneously at time 0

- C/T is the CPU utilization of a task
- U = Σ (Ci/Ti) is the CPU utilization of a task set

2.2 Rate Monotonic Scheduling algorithm (RMS)

It is a dynamic pre-emptive algorithm for scheduling set of independent hard real time tasks. This was published in 1973 by Liu and Layland [5]. The algorithm was based on static task priorities. The assumptions made about the task set are mentioned below [3, 4].

1. The request for all the task sets is periodic.

2. All tasks are independent of each other. No precedence constraints or mutual exclusion constraints exist between any pair of tasks.

3. The deadline interval of every task is equal to its period.

4. The required maximum computation time is known beforehand and is constant.

5. Time required for context switching can be ignored.

6. Sum of utilization factors of n tasks with period p is given by $U=\Sigma (ci/pi) \leq n (21/n - 1)$. As n approaches infinity, term $n (21/n - 1)$ reaches ln 2 (about 0.7).

The task priorities are assigned on the basis of their periods. The task with shortest period gets the highest priority and the task with longest period gets lowest priority. If all the assumptions stated above are satisfied then this algorithms guarantees that all the tasks will meet their deadlines. The algorithm is optimal for single processor systems.

2.2.1 Basic properties of rate monotonic scheduling

For each task that is to be scheduled we must know the value of its period T and the worst case performance time C, so that the value of the processor load coefficient could be calculated as C/T.

If the set of the tasks being scheduled is given and the characteristics of the tasks are known, a significant question is whether the time constraints of all the tasks will always be met. This is answered by the Liu and Layland theorem which is given by the following formula:-

$$\sum_{i=1} (C_i / T_i) \leq n (2^{1/n} - 1) \qquad (1)$$

In the inequality (1), n is the number of the tasks scheduled. The inequality (1) delivers only a sufficient condition for the set of schedulable tasks. However, the condition (1) is not a necessary condition for the set of schedulable tasks. Furthermore, if the condition (1) is not fulfilled, it does not automatically follow that the set of tasks is not schedulable. In this case, one must check whether the necessary condition is fulfilled. The necessary condition is given by the following formula

$$\sum_{i=1}^{n} (C_i / T_i) \leq \quad (2)$$

or

n

$$_i \min \sum_{i=1} (C_i / lT_k) \lceil lT_k / T_i \rceil \leq 1 \qquad (i= 1,2\ldots..,n) \quad (2)$$

where, min is calculated over $(k,l) \in W_i$

and $W_i = \{ (k,l) , 1 \leq k \leq i, l = 1,\ldots.., \llcorner T_i / T_k \lrcorner$

Moreover, for each task of the scheduled set of tasks it needs to be checked, whether their time constraints are met in the worst case situation, i.e. under the conditions when all the tasks enter into the ready state at the same moment. If under the worst-case conditions the performance of all the scheduled tasks is ended before the elapse of their time constraints, it means that the given set of tasks is schedulable under any circumstances. In order to prove this, one has to calculate for each task the time when its execution end. If the execution end time of each task is shorter than its time deadline, it means that the set of tasks is schedulable [5].

To calculate the time of the execution end of a periodic task the recurrent formula can be used. If we consider the lowest priority task, then the first approximation of its time of the execution end is assumed as the sum of its execution time and the times of execution of all the other tasks. This results from the fact that before the execution of the lowest priority task can be started, all the other tasks must be performed at least once. Thus, the first estimation of the time of the execution end of a task is given by the following formula

$$t_{0=} \sum_{i=1}^{n} C_i \qquad (3)$$

Then, we must systematically repeat the recurrent procedure, which is given by the following formula

$$t_{m+1 =} \sum_{i=1}^{n} C_i . \lceil t_m / T_i \rceil \qquad (4)$$

The recurrent procedure is repeated until the following condition is fulfilled

$$t_{m+1 =} \; t_m \qquad (5)$$

In such a case we consider time tm as the time of the execution end of the lowest priority task. If this time is shorter than the deadline of the lowest priority task, we can consider this task schedulable under any circumstances, because it proved to be schedulable in the worst-case scenario.

The recurrent procedure, which is discussed above, must be repeated for all the tasks and all the tasks in the worst-case scenario must be proved to be able to end their executions before the elapse of their deadlines. Only if this condition is met, the given set of periodic tasks may be considered schedulable.

2.2.2 Sufficient Schedulability Test: Utilization Bound Test (UB Test)

Assume a set of n independent tasks: S= {(C1,T1) (C2,T2)...(Cn,Tn)}and Let U= Σ Ci/Ti and B(n) =n*($2^{1/n}$-1)

Three possible outcomes:

- 0<= U<= B(n): schedulable
- B(n)<U<=1: no conclusion
- 1< U : overload

2.2.3 Sufficient and Necessary Schedulability Test

1. Calculate the worst case response time R for each task with deadline D. If R<=D, the task is schedulable/feasible. Repeat the same check for all tasks
   - Ri= Ci + ∑j ∈ HP(i) ⌈ Ri/Tj ⌉*Cj

⌈ Ri/Tj ⌉ is the number of instances of task j during Rj

⌈Ri/Tj ⌉ *Cj is the time needed to execute all instances of task j released within Rj

∑j ∈ HP(i) ⌈ Ri/Tj ⌉*Cj is the time needed to execute instances of tasks with higher priorities than ith task, released during Rj

Rj is the summation of the time required for executing task instances with higher priorities than task j and its own computing time

- We need to solve the equation:

  Ri= Ci + ∑j ∈ HP(i) ⌈ Ri/Tj ⌉*Cj

- This can be performed by numerical methods to calculate the fixed point of the equation by iteration: let

Ri0 = Ci + ∑j ∈ HP(i) Cj = C1 + C2 +…..+ Ci (the first guess)

Ri k + 1 = Ci + ∑j ∈ HP(i) ⌈ Ri/Tj ⌉*Cj (the (k+1)th guess)

- The iteration stops when either

  Rim + 1 > Ti or non-schedulable

  Rim + 1 < Ti and Rim + 1 = Rim schedulable

2. If all tasks pass the test, the task set is schedulable.
3. If some tasks pass the test, they will meet their deadlines even the other don't (stable and predictable).

The following rule of thumb can be given to simplify the schedulability check by RMS:

Step1. Apply Equation (1) and stop if all individual conditions are met. If not, apply Equation (2) for all doubtful cases, as in the next Steps (Steps 2a – 2c).

Step2a. Determine all schedulability Points by marking on a time axis all successive periods for all tasks in question, from time 0 up to the end of the first period of the lowest-frequency task.

Step2b. For each time instant marked in Step 2a - that is, for all schedulability Points - construct an inequality that has, on its left-hand side, a sum of all possible execution times of all tasks that can be activated (possibly multiple times) before this schedulability Point and, on its right-hand side, only the value of time corresponding to this schedulability Point.

Step2c. Check if values on the left-hand sides are smaller than or equal to their corresponding right-hand-side values. If at

least one of these inequalities holds, then the set of tasks is schedulable according to RMS Equation (2). If not, then the set of tasks is not schedulable according to RMS.

2.3 Deadline Monotonic Scheduling algorithm (DMS)

This technique is an extension of Rate Monotonic scheduling algorithm. This is first proposed in 1982 by Leung and Whiteland. This is also fully pre-emptive technique used for scheduling tasks with static priorities [3]. The third assumption mentioned in rate monotonic technique that says the deadline interval of every task is same and equal to its period has been relaxed. The tasks have deadlines that relative deadlines (Di) can be less than or equal to its period. Each task is allotted a fixed priority inversely proportional to its relative deadline. So, at any time task with the shortest deadline is executed. Deadline monotonic is a static priority scheduling method, as relative deadlines are constant.

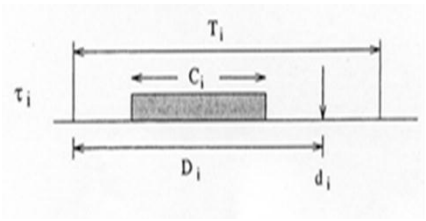

Figure2. Deadline Monotonic Scheduling

There are four parameters for each periodic task:

- A phase ϕi;
- A worst- case computation time Ci (constant for each instance);
- A relative deadline Di (constant for each instance);
- A period Ti;

which have the following relationships:

- $Ci \leq Di \leq Ti$
- $ri,k = \phi i + (k-1)Ti$
- $di,k = ri,k + Di$

2.3.1 Sufficient schedulability test: Utilization Bound test (UB test)

$\Sigma (Ci/Di) <= n*(21/n-1)$ implies schedulable by DMS

2.3.2 Precise test

- Calculate the worst case response time R as described above, for each task with deadline D. If R<=D, the task is schedulable/feasible. Repeat the same check for all tasks.
- If the test is passed by all the tasks, the task set is schedulable.
- If the test is passed by some tasks only, they will meet their deadlines even the other doesn't (stable and predictable).

3. RELATED WORK

Tradition ways of task scheduling is not fit to Cloud Computing [1][7][8]. At present, there are lots of task scheduling schemes implemented in different cloud framework. Hadoop [9] implements the FIFO (First In First Out) [10] scheme by default. The benefit of FIFO is simple and low overhead. All the jobs from different users are submitted to a queue. After that they will be examined according to the order of submission time and priority. The first job with highest priority will be selected for processing. The disadvantage of FIFO is poor fairness. The jobs with lower priority have little chance to process with lots of higher priority jobs.

In order to improve the fairness, Facebook presented Fair Scheduling Algorithm [6]. The goal of fair scheduling is that all tasks can achieve their resource as the time passes. This algorithm allows short tasks finish in reasonable time while not starving long tasks. Task occupies the whole resource with no other tasks in the system. And the system will allocate the idle time slot to those new tasks and make each of them could get equal CUP time. Fair Scheduling defines insufficiency of tasks. Tasks with more shortfalls mean they got more unfair treatments, so they have more probability to obtain resource. Apart this, fair scheduling algorithm assured the minimized shared resource. It means task with lowest priority might have its turn even if there are many tasks with higher priority.

Yahoo! presents Capacity Scheduling for Hadoop as well [11]. It allows for multiple-tenants to securely share a large cluster such that their applications are allocated resources in a timely manner under constraints of allocated capacities. This scheme allows sharing a large cluster while giving each organization a minimum capacity guarantee. Clusters will be partitioned among multiple organizations and each organization can access any excess capacity no being used by others.

All the algorithms introduced above focus on tasks of computing oriented, and not fit for service oriented tasks. In addition, Lee et al. presented a dynamic priority-scheduling algorithm on service request scheduling [12]. It adjusts the priority of task units on scheduling to increase the performance of scheduling. Yoshitomo et al. presented a history-based job scheduling mechanism for a queue system [13]. This mechanism estimates the time to start the job execution according the history of job-execution and the jobs scheduling mechanism automatically allocates the job to a suitable resource. Luqun Li offered an optimistic differentiated service task scheduling system. This paper developed a non-pre-emptive priority M/G/1 queuing model for the tasks and the system cost function for this model. Subsequent to that, the author gave the corresponding strategy and algorithm to get the approximate optimistic value of service [14]. QuXilong and Hao Zhongxiao et al. researched the distributed software resource sharing in Cloud Manufacturing system and implemented the sharing scheme in a cloud platform [15].

However, the scheduling schemes introduced above are centralized algorithms and will become bottleneck in large scale Cloud Computing environment. Moreover, they are designed for a precise computing concept, which is performance oriented and not suitable for other Cloud Computing Services, which are service oriented. The earlier one executed with short period and high utility and the later one executed with long term and lower utility [16].

- RSDC (Reliable Scheduling Distributed in Cloud computing)

Arash Ghorbannia Delavar,Mahdi Javanmard , Mehrdad Barzegar Shabestari and Marjan Khosravi Talebi [1] proposed a reliable scheduling algorithm in cloud computing environment. In this algorithm main job is divided into sub jobs. To balance the jobs, the request and acknowledge time are calculated independently. Scheduling of each job is done by calculating the request and acknowledges time in the form of a shared job, so that the efficiency of the system is increased.

- An Optimal Model for Priority based Service Scheduling Policy for Cloud Computing Environment

Dr. M. Dakshayini, Dr. H. S. Guruprasad [3] proposed a new scheduling algorithm based on admission and priority control method. In this algorithm, priority is assigned to each admitted queue. Entrance of each queue is decided by calculating tolerable delay and service cost. The advantage of this algorithm is that with the proposed cloud architecture this scheme has achieved very high (99%) service completion rate with definite QoS. As this scheduling provides the highest preference for highly paid user requests for service, total servicing cost for the cloud also increases.

- A Priority based Job Scheduling Algorithm in Cloud Computing

Shamsollah Ghanbari, Mohamed Othman [17] proposed a new scheduling algorithm based on multi – criteria and multi - decision priority driven scheduling algorithm. This scheduling algorithm have three levels of scheduling: object level, attribute level and alternate level. This algorithm set the priority by job resource ratio. Next priority vector can be compared with each queue. This algorithm has high throughput and less finish time.

- Extended Max-Min Scheduling Using Petri Net and Load Balancing

El-Sayed T. El-kenawy, Ali Ibraheem El-Desoky, Mohamed F. Al-rahamawy [5] has proposed a new algorithm based on impact of RASA algorithm. Extended Max-min algorithm is based on the expected execution time rather on complete time as a selection basis. To model the concurrent behavior of distributed systems Petri nets are used. Max-min algorithm shows achieving schedules with comparable lower makespan rather than RASA and original Max-min.

- An Optimistic Differentiated Job Scheduling System for Cloud Computing

Shalmali Ambike, Dipti Bhansali, Jaee Kshirsagar, Juhi Bansiwal [6] has proposed a differentiated scheduling algorithm with non-preemptive priority queuing model for activities performed by cloud user. In this method, a web application is created to do some activity like one of the file uploading and downloading then there is need of efficient job scheduling algorithm. This algorithm helps in achieving the QoS requirements of the cloud computing user and the maximum profits of the cloud computing service provider.

- Improved Cost-Based Algorithm for Task Scheduling

G.Mrs.S.Selvarani, Dr.G.Sudha Sadhasivam [7] proposed an improved cost-based scheduling algorithm for making efficient mapping of tasks to available computing resources in cloud environment. The managing of traditional activity based costing is proposed by new task scheduling strategy for cloud environment where there may be no relation between the overhead application base and the way that different tasks cause overhead cost of resources in cloud. The proposed algorithm divides all user tasks depending on priority of each task into three different lists. The proposed algorithm calculates both resource cost and computation performance. It also improves the computation/communication ratio.

- Performance and Cost evaluation of Gang Scheduling .in a Cloud Computing System with Job Migrations and Starvation Handling

Ioannis A. Moschakis and Helen D. Karatza proposed a gang scheduling algorithm with job migration and starvation handling. The number of Virtual Machines (VMs) available at any moment is dynamic and scales according to the demands of the jobs being serviced. The above mentioned model is studied through simulation in order to analyze the performance and overall cost of Gang Scheduling with migrations and starvation handling. Results show up that this scheduling strategy can be effectively deployed on cloud environment, and that cloud platforms can be feasible for HPC or high performance enterprise applications.

4. CONCLUSION

Scheduling is one of the most important accept in cloud computing environment. So, in this paper, we focused on task scheduling in cloud computing environment with certain QoS constraint. Rate Monotonic algorithm is simpler to implement and exhibits a predictable behaviour resulted from its associated analysis. In this paper, fixed priority scheduling algorithms i.e. Rate Monotonic and Deadline Monotonic scheduling algorithms are explained, when using them in cloud computing environment for improving the QoS parameters

REFERENCES

[1] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, NIST (National Institute of Standards and Technlogy) Special Publication 800-145.

[2] Jun Huang, Yanbing Liu, Qiang Duan, “Service Provisioning in Virtualization-based Cloud Computing: Modeling and Optimization”, Globecom, 2012.

[3] Q. Duan, “Modeling and Performance Analysis on Network Virtualization for Composite Network-Cloud Service Provisioning,” in Proc. of SERVICES, 2011, pp. 548–555, July 2011.

[4] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” IEEE J. Sel. Areas. Commun., vol. 14, no. 7, pp. 1228–1234, Sept. 1996.

[5] G. Xue, A. Sen, W. Zhang, J. Tang and K. Thulasiraman, “Finding a path subject to many additive QoS constraints,” IEEE/ACM Trans. Netw., vol. 15 , no. 1, pp. 201-211, Feb. 2007.

[6] G. Xue, W. Zhang, J. Tang and K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing," IEEE/ACM Trans. Netw., vol. 16, no. 3, pp. 656-669, Jun. 2008.

[7] J. Huang, X. Huang, and Y. Ma, "An Effective Approximation Scheme for Multi constrained Quality-of-Service Routing," in Proc. IEEE GLOBECOM 2010, Miami, Florida, pp. 1–6, Dec. 2010.

[8] F. A. Kuipers, P.V. Mieghem, T. Korkmaz and M. Krunz, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing," IEEE Com. Mag., vol. 40, no. 12, pp. 50–55, Dec. 2002.

[9] Z. Tarapata, "Selected multi criteria shortest path problems: an analysis of complexity, models and adaptation of standard algorithms," Int. J. Applied Math. and Comp. Sci., vol. 17, no. 2, pp. 269–287, July 2007.

[10] R. G. Garroppo, S. Giordano and L. Tavanti, "A survey on multi constrained optimal path computation: Exact and approximate algorithms," Compt. Netw., vol. 54, no. 17, pp. 3081–3107, Dec. 2010.

[11] X. Yuan, "Heuristic Algorithms for Multi constrained Quality-of-Service Routing," IEEE/ACM Trans. Netw., vol. 10, no. 2, pp. 244–256, Apr. 2002.

[12] P. V. Mieghem and F.A. Kuipers, "Concepts of Exact QoS Routing Algorithms," IEEE/ACM Trans. Netw., vol. 12, no. 5, pp. 851–864, Oct. 2004.

[13] Jun Huang, et al., "Novel End-to-End Quality of Service Provisioning Algorithms for Multimedia Services in Virtualization-based Future Internet", IEEE Transactions On Broadcasting.

[14] Yee Ming Chen, Yi Jen Peng, "A QoS aware services mashup model for cloud computing applications" Journal of Industrial Engineering and Management, JIEM, 2012 – 5(2): 457-47.

[15] Pinal Salot Purnima Gandhi , "Job Resource Ratio Based Priority Driven Scheduling in Cloud Computing", International Journal for Scientific Research & Development, Vol. 1, Issue 2, 2013 , ISSN (online): 2321-0613.

[16] Bing Li, A Meina Song, Junde Song, "A Distributed QoS-Constraint Task Scheduling Scheme in Cloud Computing Environment: Model and Algorithm", Advances in information Sciences and Service Sciences(AISS), Volume4, Number5, March 2012.

[17] Ghanbari Shamsollah, and Othman Mohamed, " A Priority based Job Scheduling Algorithm in Cloud Computing", Procedia Engineering, Vol. 50, pp. 778-785, 2012.