ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (884-891), Month: April - June 2015, Available at: www.researchpublish.com

Efficient Scheduling Of On-line Services in Cloud Computing Based on Task Migration

[1]Harish H G, [2]Dr. R Girisha

[1]PG Student, [2]Professor, Department of CSE, PESCE Mandya (An Autonomous Institution under VTU Belgaum), Karnataka India

***Abstract*: This paper presents a new scheduling approach to focus on providing a solution for online scheduling problem of real-time tasks using "Infrastructure as a Service" model offered by cloud computing. The real time tasks are scheduled pre-emptively with the intent of maximizing the total utility and efficiency. In traditional approach, the task is scheduled non- pre-emptively with two different types of Time Utility Functions (TUFs) – a profit time utility function and a penalty time utility function. The task with highest expected gain is executed. When a new task arrives with highest priority then it cannot be taken for execution until it completes the currently running task. Therefore the higher priority task is waiting for a longer time. This scheduling method sensibly aborts the task when it misses its deadline. Note that, before a task is aborted, it consumes system resources including network bandwidth, storage space and processing power. This leads to affect the overall system performance and response time of a task. In our approach, a preemptive online scheduling with task migration algorithm for cloud computing environment is proposed in order to minimize the response time and to improve the efficiency of the tasks. Whenever a task misses its deadline, it will be migrated the task to another virtual machine. This improves the overall system performance and maximizes the total utility. Our simulation results outperform the traditional scheduling algorithms such as the Earliest Deadline First (EDF) and an earlier scheduling approach based on the similar model.**

***Keywords*: non-preemptive, preemptive, priority, deadline, migrate, virtual machine, Time Utility Function.**

I. INTRODUCTION

Cloud computing is a prototype in which resources are permanently stored in servers on the internet and cached provisionally on clients. Large pool of systems is connected in private or public network. Cloud has the extended sized vision of computing as utility, making the software even more attractive as a service and they consume resources as a service in which pay only for the resources that they use [9]. It includes any subscription based or pay-per-use, where it can obtain network storage space and computer resources. Pay-as-you-go is the means of payment of cloud computing, only paying for the actual consumption of resource. Traditionally, users have to equip with all software and hardware infrastructure before computing starts, and maintain them during computing process. Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way in which hardware is designed and purchased [8]. A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over Internet.

Utility computing considers the computing and storage resources as a metered service. The customers can use the utility services immediately whenever and wherever they need without paying the initial cost of the devices. Infrastructure as a Service (IaaS) provides the processing, storage, networks, and other fundamental computing resources to users [1] [8]. IaaS users can deploy arbitrary application, software, operating systems on the infrastructure, which is capable of scaling up and down dynamically. IaaS user sends programs and related data, while the vendor's computer does the computation processing and returns the result. The infrastructure is virtualized, flexible, scalable and manageable to meet user requirements. Cloud service provider gives many resources to the consumer through private cloud or public cloud. A number of users access the same resource, but they exchanged. Each task happen in real time system has timing properties and these properties are measured when scheduling the task on real time system. Cloud Computing refers to the use of computing, platform, software, as a service. It's a form of utility computing where the customer need not own the necessary infrastructure and pay for only what they use.

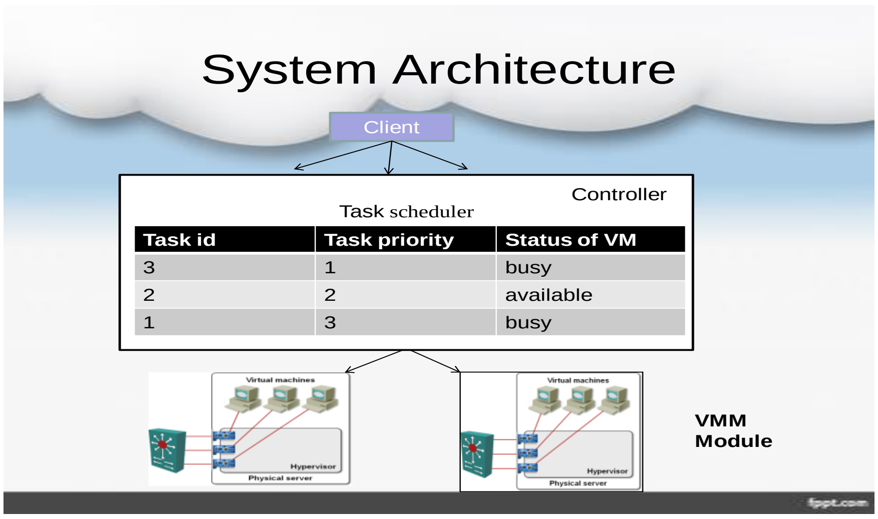
Computing resources are delivered as virtual machines [1][8]. While there exist different interpretation and views on cloud computing, it is less disputable that being able to effectively exploit the computing resources in the clouds to provide computing service at different quality levels is essential to the success of cloud computing. For real-time applications and services, the timeliness is a major criterion in judging the quality of service. Due to the nature of real-time applications over the Internet, the timeliness here refers to more than the deadline guarantee as that for hard real-time systems. In this regard, an important performance metric for cloud computing can thus be the sum of certain value or utility that is accrued by processing all real-time service requests. An independent task models that are subject to deadline constraints specified using Time Utility Functions (TUFs). It is inefficient for independent task model in the sense that it simply aborts any task with lower utility without accounting that those tasks just need to be preempted instead of being aborted. A time constraint using a TUF that it accrues optimal utility during under loads and attempts to maximize the accrued utility during overloads [10].

To improve the performance of cloud computing, that not only measures the profit when completing a job in time, but also account for the penalty when a job is aborted or discarded [11]. Before a task is aborted or discarded it consumes system sources including network bandwidth, storage space and processing power and thus can be affect the system performance. Virtualization in clouds refers to multi-layer hardware platforms, operating systems, storage devices, network resources, etc [8]. The first prominent feature of virtualization is the ability to hide the technical complexity from users, so it can improve independence of cloud services. Secondly, physical resource can be efficiently configured and utilized, considering that multiple applications are run on the same machine. Thirdly, quick recovery and fault tolerance are permitted. Virtual environment can be easily backed up and migrated with no interruption in service. A TUF is the generality of deadline constraints which specifies the value of the system ensuring from the completion of an activity as a function of its completion time. Successful completion of a task must not go beyond a specific bound and maximize the system utility.

All these variations in UA scheduling algorithm determines the maximal utility to be accrued only when a task is successfully completed and the task should be deferred or rejected in a controlled fashion. A task model considers both profit and penalty while executing a task. The optimization goal of Profit Penalty (PP) aware scheduling for online service system is to maximize profit obtained through timely service.

II. SYSTEM ARCHITECTURE

System architecture is the conceptual design that defines the structure and behavior of a system shown in fig1. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system.



Task id	Task priority	Status of VM
3	1	busy
2	2	available
1	3	busy

Fig1. System Architecture

The fig1 shows the task is associated with a pair of unimodel time functions represents the accrued profit of a system. Therefore, one utility function cannot accurately represent both the profit and penalty information when executing a task. Scheduling problems belong to a broad class of combinational optimization problems aiming at finding an optimal matching from a finite set of objects, so the set of feasible solutions is usually discrete rather than continuous. The scheduling points of a scheduler are of points on time line at which the scheduler makes decision regarding which task is to be run next. A task scheduler does not need to be run continuously, it is activated only at the scheduling points to make decision to which the task to be run next [8]. Real time task scheduling determines the order in which the various task are taken for execution [1] [8]. No task is taken for schedule before its arrival time and the precedence and the timing constraints of all tasks are satisfied. In such a scenario, task scheduling algorithms play an important role where the aim is to schedule the tasks effectively so as to reduce the turnaround time and improve resource utilization.

A real-time scheduler must ensure that processes meet deadlines, regardless of system load or makes span. 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. A scheduler is called dynamic if it makes scheduling decisions at run time, selecting one out of the current set of ready tasks. A scheduler is called static if it makes scheduling decisions at compile time. A non preemptive online scheduling of real time services with task migration algorithm to provide the solution for the task abortion when its misses the deadline. The execution of a task may get potential profit or potential loss. The penalty will degrade the overall computing performance [10].

In the context of real time systems, the scheduling algorithm is priority driven. The tasks are assigned priorities according to their constraints and generally the highest priority is assigned to the most urgent task [8]. In existing approaches, a non-preemptive scheduling algorithm is used. Initially tasks are accepted in the ready queue and then each task is scheduled non-preemptively. If the task is currently being executed, it cannot be removed until it complete from its execution. Suppose a new task arrives with highest priority, the high priority task cannot be taken to execute until the existing task is successfully completed. Therefore, high priority tasks are waiting for a longer duration of time. As a result, it affects the overall system performance and response time of a task. In this paper, we present a preemptive online scheduling algorithm of real time services with task migration. Our algorithm give priority for the tasks and sensibly migrate the tasks when it misses its deadline in order to minimize the response time and consequently achieve better performance.

III. PRELIMINARY

In this paper, the arrived real time tasks αt = {T1, T2, T3,…….., Tn} are arranged in a priority order. A ready queue consists of n real time tasks which are scheduled non- preemptively. In this scheduling criterion, the currently executing task cannot be removed until it completes its execution. The highest priority tasks are waiting for a longer duration of time. The task will be aborted from its execution when it misses its deadline.

Therefore the response time of the task will be greater and problem; we developed an online pre-emptive scheduling with task migration algorithm in order to maximize the total utility and efficiency.

IV. PREMPTIVE ON-LINE SCHEDULING OF ON-LINE SERVICES WITH TASK MIGRATION

In this section, we present a preemptive online scheduling of real time services with task migration algorithm to provide the solution for the task abortion when its misses its deadline and most preferably task with the higher priority is taken for its execution. Our pre-emptive scheduling algorithm works at scheduling points that include the arrival of a higher priority task and the critical point of the current task. Each task has its own priority. The priority of the task will be calculated by its deadline and cost. The critical point is the point in which a task misses its deadline.

The detailed algorithm is described in algorithm1

Algorithm 1: PRE-EMPTIVE ON-LINE TASK SCHEDULING ALGORITHM

Step 1: Let the tasks {T1,T2…….Tn} be the accepted in the ready queue.

Step 2: Arrange the tasks in the ready queue based on the priority

Step 3: Task with the highest priority in the ready queue taken for its execution.

Step 4: If a new task arrives, put the task at the head of the ready queue and sort the task based on the priority.

Step 5: A new task arrives with the highest priority is ready to run and pre-empts the currently executing task comparatively.

Step 6: The pre-empted task re-enters into the ready queue and sorted according to the priority.

Step 7: Whenever an executing task misses its deadline, it will be migrated to another virtual machine.

The real time tasks are accepted in the ready queue. For each task, the priority is calculated based on deadline and cost. Sort all the tasks in the ready queue based on the priority and the task with highest priority is selected for execution. Upon the completion of the current task, choose the task from the ready queue which has the highest priority and starts its execution. If a new task arrives, it is first inserted at the head of the ready queue and calculates the priority. Based on the priority of the task, it is compared with all the tasks and inserted accordingly in the queue. This procedure continues until the entire ready queue becomes a list ordered according to their priority. The currently executing task is preempted whenever a new task arrives with the higher priority. The preempted task is put into the ready queue and sorts it according to the priority. Whenever the currently executing task reaches its deadline, the task is instantly migrated to another virtual machine.

Algorithm 2: TASK MIGRATION ALGORITHM

Step 1: When a task misses its deadline, the decision is taken to migrate the process is made to destination node.

Step 2: The execution of the process on the source node is suspended and put into the migrating state.

Step 3: Extract the processing states from the source node.

Step 4: The subset of the process’s state is transferred transparently from the source node to destination node.

Step 5: The process state is reconstructed on the destination node from the state information.

Step 6: Communication channels are enabled after migration at the destination node.

Step 7: The execution of the process is begun on the destination node.

When a task reaches its deadline, a decision is made to send a migration request is issued to destination node. The execution of a task on the source node is suspended and affirms it to be in a migrating state. The task is extracted and is typically retained on the source node until the end of the migration. Once the processing state resumes its execution on the destination host, the transfer of task from the source node to destination node should be transparent to the execution of the process. When the sufficient state has been transferred and imported, then the task migration completes. Once all of the states have been transferred from the queue, it may be deleted on the source node.

V. SIMULATION RESULTS

The proposed algorithm is simulated to explore the system performance to achieve less response time, total utility and efficiency. We compared our proposed algorithm with the traditional Earliest Deadline First (EDF) and non pre-emptive scheduling algorithm. EDF scheduling policy is the job closest to its deadline is to be served. The key EDF is a dynamic scheduling algorithm, the priority of a task can change during its execution. Non pre-emptive scheduling algorithm uses a heuristically defined parameter and currently executing task cannot be removed until it completes from its execution. We compared the three algorithms with a set of twenty Five tasks based on five constraints. The constraint includes absolute profit, absolute penalty, total utility, efficiency and throughput.

Absolute profit is defined as when the task is successfully completed in the ready queue. The profit value by preemptive is compared with EDF and Non preemptive online scheduling approaches. While comparing, EDF can execute the task with highest priority dynamically and Nonpreemptive measures the profit value when the job is completed in time. In preemptive scheduling algorithm, the high priority tasks are scheduled first in the ready queue. As a result the response time will be minimum .Fig. 2 shows that the profit value is maximum for the preemptive algorithm when comparing other two different approaches.

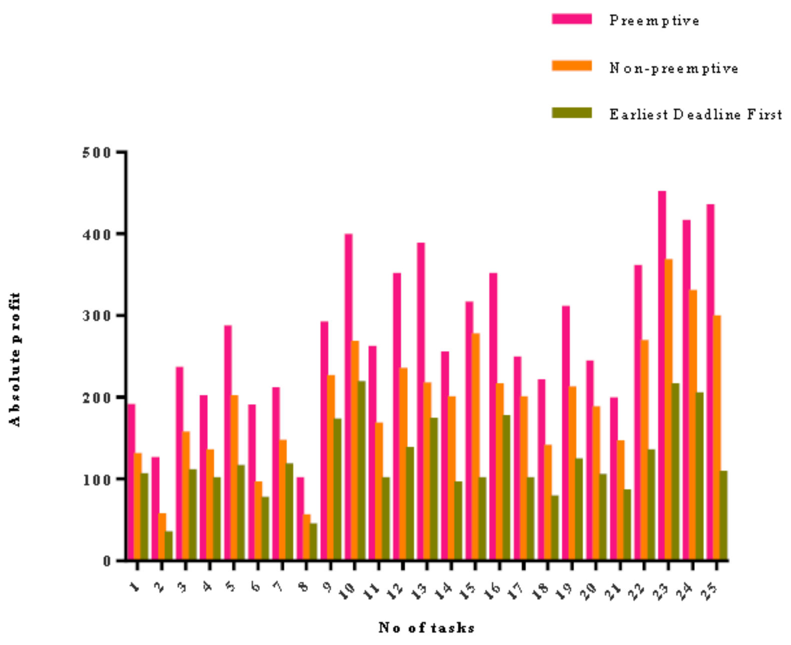


Fig2. Comparison of absolute profit by three distinct methods

To depict the absolute penalty, Fig. 3 shows the variation between the penalty and the number of tasks. Penalty is defined as the task when misses its deadline. Our algorithm is compared with other two different approaches. In preemptive scheduling algorithm, if a task misses its deadline then the task will be migrated to another virtual machine. The results show that our proposed algorithm attains lesser penalties. Because in EDF, the task closest to deadlines is to be served and the task will be aborted when it misses its deadline in Non-preemptive online scheduling approach.

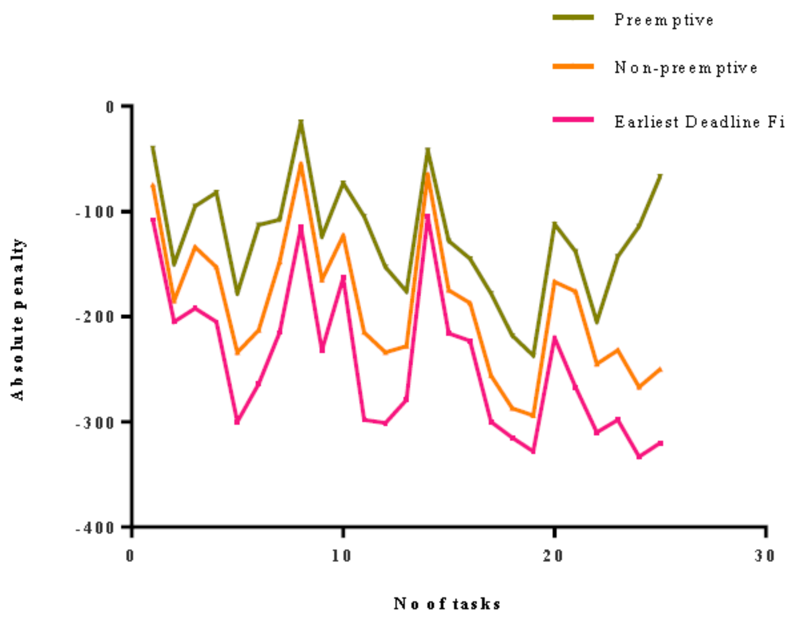


Fig3. Comparison of absolute penalty by three distinct methods

Scheduling a task is to show the performance of total utility in limited resource. The total utility is the difference between the absolute profit and absolute penalty. In our proposed algorithm, the tasks are scheduled pre emptively in order to reduce the waiting time for high priority tasks. Hence, it will improve the total utility. Fig. 4 shows that the preemptive algorithm can achieve better performance than the other two different approaches. Our algorithm significantly reduces the total penalty and as a result, total utility is increased.

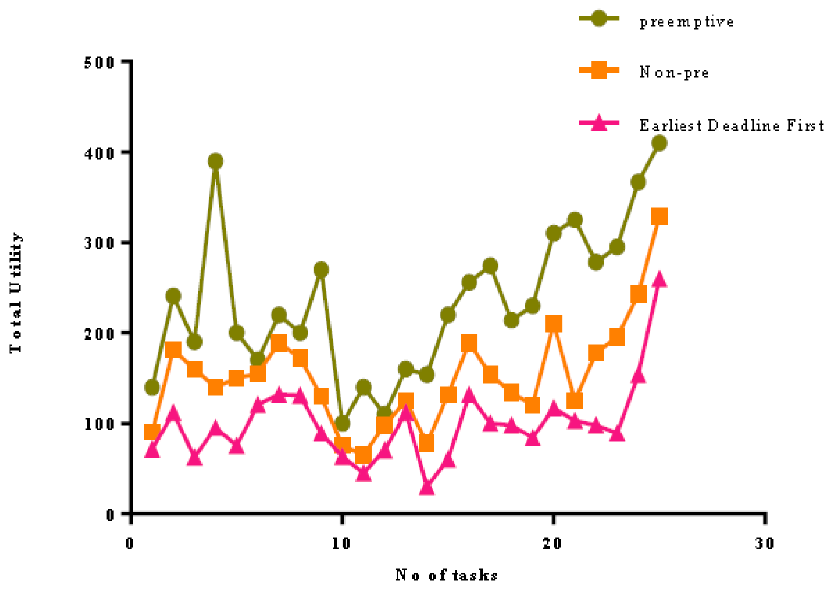


Fig4. Comparison of total utility by three distinct methods

Efficiency is the ability to achieve a task with least overheads of time. Efficiency of preemptive scheduling algorithm is compared with Earliest Deadline First and Non-preemptive algorithm. In our approach, the high priority task could not wait for indefinite amount of time. If the task misses its deadline, it will be migrated to another virtual machine. Fig. 5 shows that our proposed algorithm attains better efficiency than Earliest Deadline First and Non-preemptive algorithm. In Earliest Deadline First, the high priority task delay for longer time and Non-preemptive algorithm, the highest priority task waits until the executing task is completed.

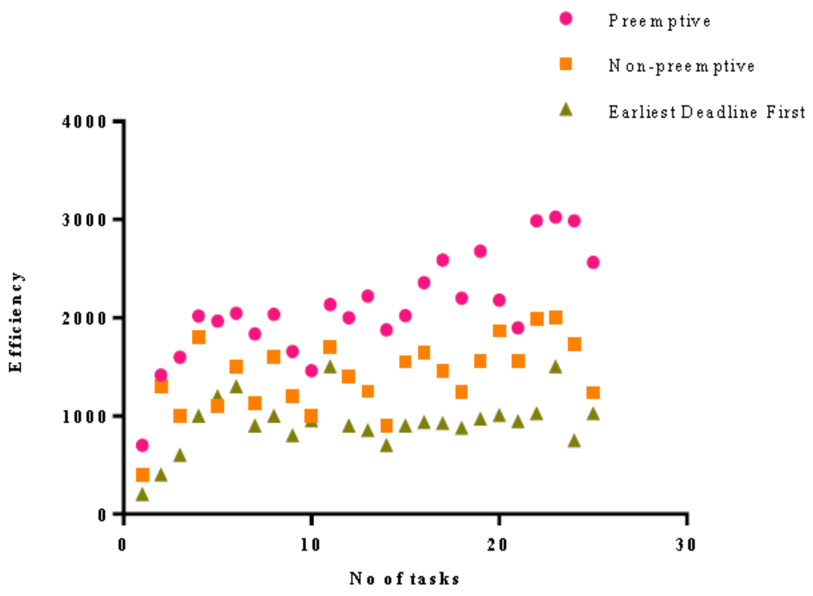


Fig5. Comparison of efficiency by three distinct methods

Throughput calculates the number of tasks completed from the arrived set of tasks in the ready queue. Fig. 6 describes that our algorithm achieves better throughput when compared to Non-preemptive and EDF. In novel approach, the preference will be given to the higher priority task to perform its execution. Whenever a penalty occurs, it will be migrated a task to another virtual machine. So that many number of task can be completed successfully. In EDF, the completion of the task remains delayed, since it is running for a longer time and Non preemptive, the task will be aborted or discarded when a penalty is occurred.

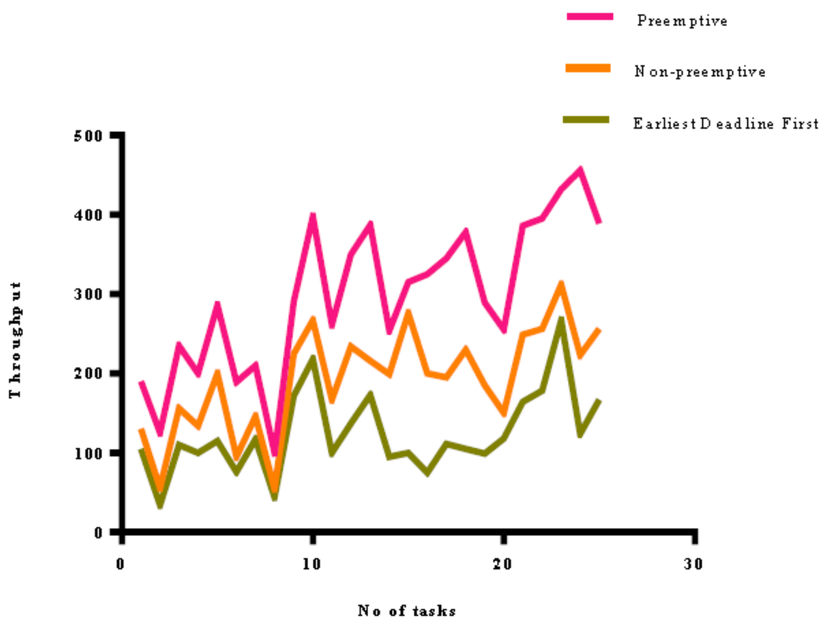


Fig6. Comparison of throughput by three distinct methods

VI. CONCLUSION AND FUTURE ENHANCEMENT

With the trends in cloud computing, cloud scheduler are designed to assist the infrastructure as a service to reflect on the large scale of computing resource for cloud computing. In existing approach, the tasks are scheduled non-preemptively. Whenever a task enters into the ready queue, it is not rubbed out until it completes its execution. The highest priority task is waiting for vague amount of time. When the time reaches the deadline of the current task, the current active task is immediately discarded and the task with the highest expected gain is selected for execution.

Therefore, it affects the overall system performance and response time. In order to address these challenges, this paper is proposed to minimize the response time and to improve the efficiency. The task with higher priority in the ready queue is accepted for its execution. The currently executing task is pre-empted as soon as the task with the higher priority enters into the ready queue. The pre-empted task re-enters into the ready queue and sorted according to their priority. A task will be migrated to new virtual machine, when it misses its deadline. Migration algorithm executes the tasks completely and improves the total utility and efficiency. Our simulation result shows that our proposed algorithm can significantly outperform the EDF and Non Preemptive scheduling algorithm. There are quite interesting research problem for our future work is to decrease the execution time of pre-empted task.

REFERENCES

[1] E. Knorr and G. Gruman (2010)., What cloud computing really means, http://www.infoworld.com.

[2] "Amazon EC2 Pricing" (http://aws.amazon.com/ec2/pricing/).

[3] Utility Accrual, Real-Time Scheduling Under the Unimodal Arbitrary Arrival Model with Energy Bounds, 2008---IEEE Transactions.

[4] Scheduling Fixed-Priority Tasks with Preemption Threshold, 2009---IEEE Transactions.

[5] A new process migration algorithm, 2010---IEEE Transactions.

[6] A Method for Performance Analysis of Earliest-Deadline-First Scheduling Policy, 2011---IEEE Transactions.

[7] On-Line Scheduling of Real-Time Services for Cloud Computing, 2012---IEEE Transactions.

[8] Xiaomin Zhu, Member, IEEE, Laurence T. Yang, Member, IEEE, Huangke Chen, Ji Wang, Shu Yin, Member, IEEE, and Xiaocheng Liu "Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds" IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 2, APRIL-JUNE 2014.

[9] E. D. Jensen, C. D. Locke, and H. Toluca (1985)., A time-driven scheduling model for real-time systems, In IEEE Real-Time Systems Symposium.

[10] R. Santhosh and T. Ravichandran," Non-Preemptive on-line scheduling of real-time services with task migration for cloud computing", European Journal of Scientific Research ISSN 1450- 216X Vol. 89 No 1 October, 2012, pp.163-169© Euro Journals Publishing, Inc. 2012

[11] P. Li, H. Wu, B. Ravindran, and E. Jensen (2006)., A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints, Computers, IEEE Transactions on, 55(4):454–469.

[12] Venkatesa Kumar, V. and S. Palaniswami," A dynamic resource allocation method for parallel data processing in cloud computing", Journal of Computer Science 8 (5): 780-788, 2012 ISSN 1549-3636© 2012 Science Publications.