Article can be accessed online at http://www.publishingindia.com

A Priority Constrained Pre-emptive Scheduling of Online Real Time Services with Fixed Checkpoint Intervals for Cloud Computing

Santhosh R.*, Ravichandran T**

Abstract

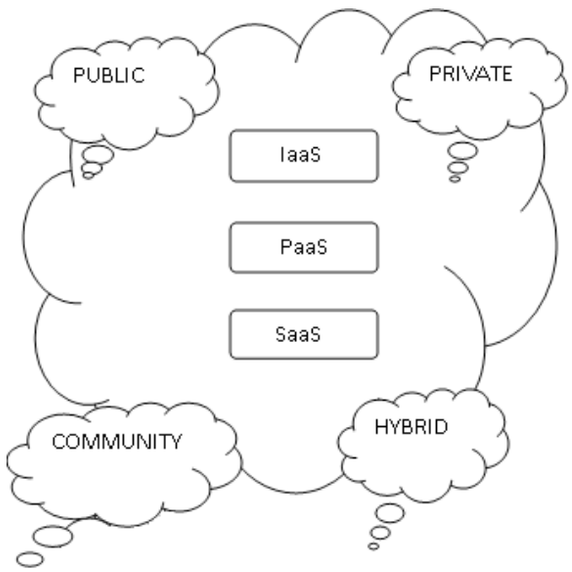
In cloud computing, various services are accessed by the different types of clients as pay per use over the internet. This paper presents a new scheduling technique for real time tasks in order to minimize the execution time and focuses on the priority constrained tasks. In older approaches, a non-preemptive scheduling with task migration algorithm is used to schedule the task with highest expected gain and executes the tasks in the queue in a non-preemptive manner. Therefore it increases the execution time of the task and response time of the priority constrained tasks. In order to overcome this problem, a priority constrained pre-emptive scheduling of online real time services with fixed checkpoint intervals is proposed to minimize the execution time of the tasks and improves the overall system performance by giving importance for higher priority tasks. Our simulation results outperform the traditional scheduling algorithms based on the similar model.

Keywords: Execution Time, Priority, Deadline, Pre-emptive, Checkpoint Intervals.

1. Introduction

Cloud is one of the computing models and it refers to cyberspace based computing to provide the resources as a service to cloud clients. The services are incorporated into infrastructure as a service, software as a service and platform as a service. Cloud service provider has resources such as data, application, hardware, server, storage, and networks (Weiss, 2007). Infrastructure as a service (IaaS) offer infrastructure components to the consumer on a self service basis. Platform as a service (PaaS) provides operating system, database and middleware to the consumers to address the complications in software development environment. Software as a service (SaaS) is to consume the software whenever needed for cloud clients (Weiss, 2007). These services are deployed through different types of cloud environments such as public cloud, private cloud, community cloud and hybrid cloud.

Figure 1: Cloud Architecture



Cloud deployment models make their services available and ready to use.

* Research Scholar, Department of Computer Science and Engineering, Karpagam University, Coimbatore, Tamil Nadu, India. E-mail: santhoshrd@gmail.com

**Principal, Hindusthan Institute of Technology, Coimbatore, Tamil Nadu, India. E-mail: dr.t.ravichandran@gmail.com

In cloud computing, scheduling plays a vital role to decide how to allocate the resources for different tasks. The main aim of this scheduling is to completion of a task within its deadline. This incurs profit rather than loss and performance of the system will be improved. Hence an effective scheduling algorithm is needed for successful completion of various tasks in the cloud environment.

In non-preemptive scheduling with task migration algorithm (Santhosh, 2012), a task with highest expected gain is taken for its execution. When a new task enters with highest priority, it cannot be taken for its execution until an executing task successfully completed. Whenever a task misses its deadline it will be migrated to another virtual machine and restarts its execution from the beginning. Therefore it increases the execution time of a task as well as the higher priority tasks remains waiting for a longer duration of time.

In this paper, we present a priority constrained pre-emptive scheduling of online real time services with fixed checkpointing algorithm. Our algorithm gives importance for higher priority tasks and decreases the execution time of the tasks.

2. Preliminary

In this paper, a sequence of erratically arrived tasks is $\eta = \{P_1, P_2, P_3 \ldots Pn\}$ are arranged and defined using the following parameters.

T_P = Priority of the tasks. P_E = Predicted execution time of a task.

P_s = Predicted starting time of a task.

H= Time required for executing a task.

R= Rate of a task for its execution.

A_T = Arrival time of a task.

I_n = Checkpoint interval.

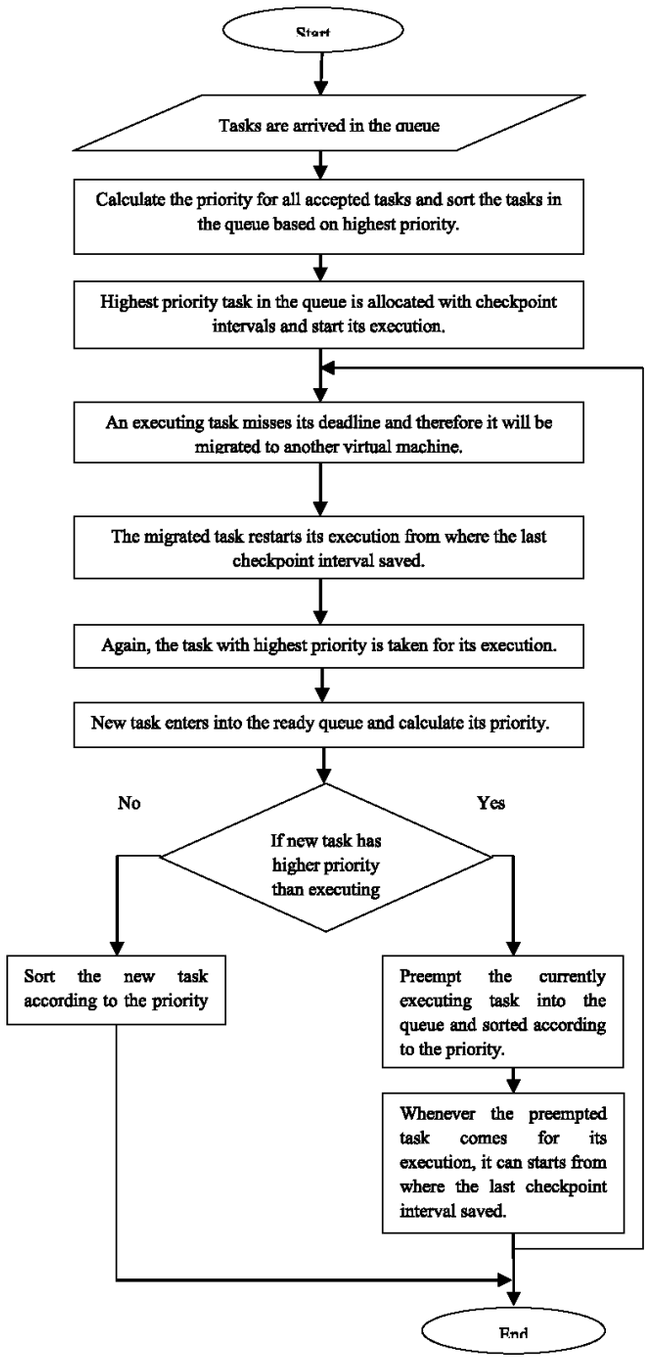
C_s= Time required for saving each checkpoint interval.

R_f = Incompletion of the task due to resource failure.

N_t = Successfully completed tasks.

With this, we developed a priority constrained pre-emptive scheduling of real time services with fixed checkpointing algorithm in order to improve the overall system performance.

Figure 2: Flow Chart for Pre-Emptive Scheduling with Fixed Checkpoint Intervals



3. A Priority Constrained Pre-Emptive Scheduling of Real-Time Services with Fixed Checkpoint Intervals

In this section, we present a priority constrained pre-emptive scheduling of real time services with checkpointing

algorithm to provide a solution for decreasing the execution time of the task using checkpointing intervals and also it gives importance for higher priority tasks in the queue.

Algorithm 1: Pre-Emptive Online Scheduling Algorithm

Step 1: Let $\eta = \{P_1, P_2 \ldots Pn\}$ be the arrived task in the ready queue.

Step 2: Priority of a task can be calculated by

$$T_P = P_E - (A_T - P_S) + (H \times R) \quad (1)$$

Step 3: Sort the tasks in the ready queue according to the priority.

Step 4: Checkpoint intervals will be allocated for the task with highest priority and starts its execution.

Step 5: Whenever a new task arrives with highest priority than the executing task, then the currently executing task will be preempted.

Step 6: The preempted task again enters into the ready queue and sorted according to the priority.

Step 7: When the preempted task comes for its execution, it can starts its execution from where the last checkpoint interval saved.

Step 8: Whenever an executing task misses its deadline, it will be migrated to another virtual machine.

Step 9: A migrated task is removed from its source machine and the communication channels are forwarded temporarily.

Step 10: The processing methods are transferred from the source machine to the migrated virtual machine and the communication channels are enabled.

Step 11: The migrated task restarts its execution from where the last checkpoint interval saved.

Calculate the priority for all the tasks in the queue and sort according to the priority. The task with the highest priority is allocated with checkpoint intervals and starts its execution. When a new task arrives and it has higher priority value than executing task then it will be preempted and again it reenters into the ready queue. When a preempted task comes for its execution it will start its execution from where the last checkpoint interval saved.

Whenever an executing task misses its deadline, it will be migrated to another virtual machine. The migrated task is detached from its source machine and transfers the task into another virtual machine. The migrated task restarts its execution from where the last checkpoint interval saved.

The detailed algorithm for fixed checkpoint interval is described in algorithm 2.

Algorithm 2: Fixed Checkpointing Algorithm

Step 1: Size of the task will be allocated with fixed checkpoint intervals.

Step 2: For each interval, save the execution of a task in the secondary disk space.

Step 3: Calculate the fixed checkpoint intervals of a task.

$$I_n = n\sqrt{\left(\frac{2C_s R_f}{N_\tau + R_f}\right)} + (n-1)C_s \quad (2)$$

Where n= 1, 2, 3… m

Step 4: Find the last saved fixed checkpoint interval for the migrated and preempted task and restarts its execution.

Whenever a task comes for its execution, it will be allocated with fixed checkpoint intervals according to its size. For both migrated and preempted task, find the last saved fixed checkpoint interval and restarts its execution.

4. Simulation Results

Our algorithm is proposed to improve the profit, throughput and minimize the execution time of the task. Our algorithm is compared with non-preemptive scheduling with task migration algorithm. This scheduling algorithm executes the tasks in the queue in non-preemptive manner but whenever a task with the higher priority reaches the ready queue it has to wait for longer duration of time. After successful completion of an executing task, the next task with higher priority is taken for its execution. We compared this algorithm with our proposed algorithm with twenty five set of tasks based on profit, loss, throughput and execution time.

Figure 3: Comparison of profit between pre-emptive and non-preemptive scheduling

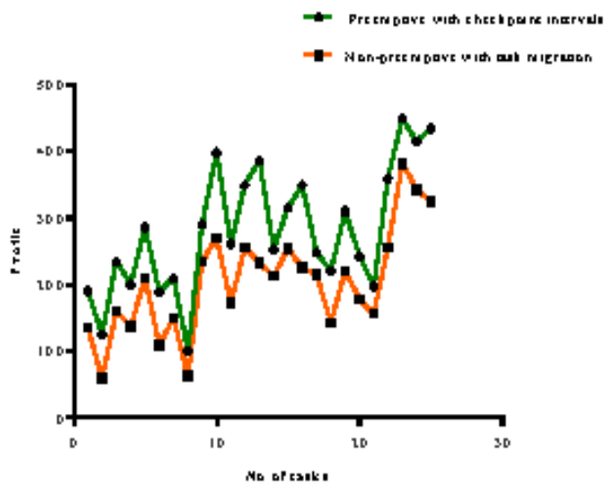


Figure 4: Comparison of loss between pre-emptive and non-preemptive scheduling

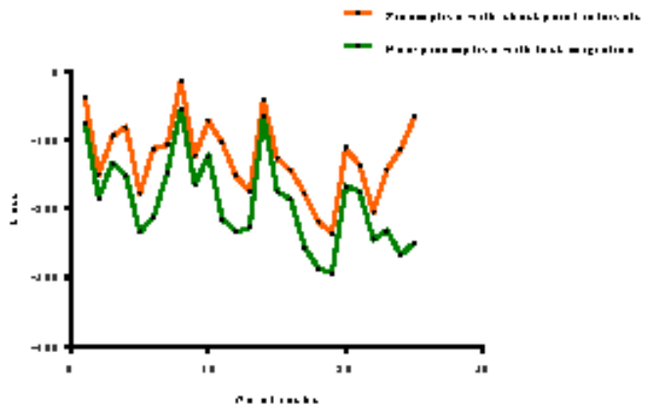


Figure 5: Comparison of throughput between pre-emptive and non-preemptive scheduling

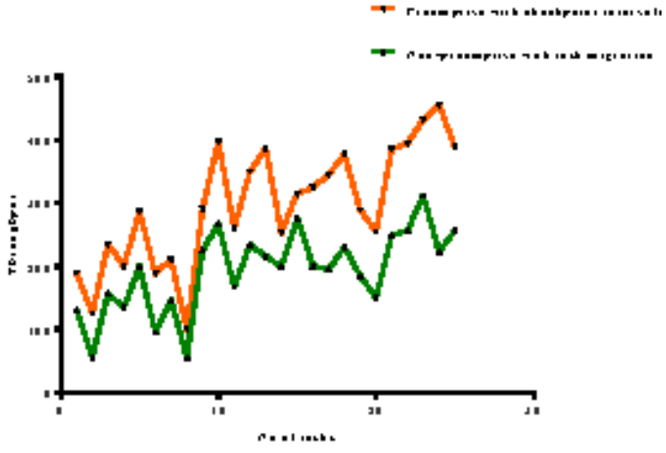


Figure 6: Comparison of execution time of a task between pre-emptive and non-preemptive scheduling

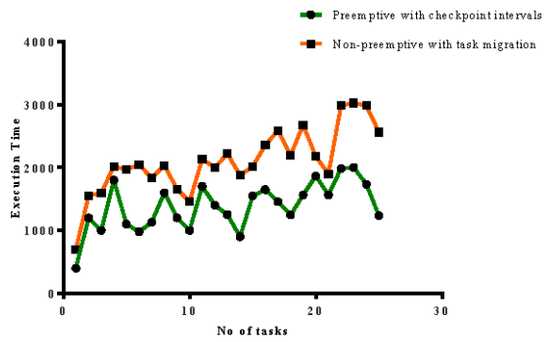


Figure 3 shows the graph a set of tasks which attains a profit value. It can be assessed only when the task successfully completes its execution within its deadline. In non-preemptive scheduling with task migration algorithm, higher priority tasks remains waiting for a longer duration of time. While comparing these two scheduling algorithms, a proposed algorithm attains more profit than the non-preemptive scheduling with task migration algorithm.

The graph for potential loss is shown in figure 4. The loss value is defined as the task did not complete its execution within its deadline. In non-preemptive scheduling with task migration algorithm, a task which misses its deadline, it will be migrated to another virtual machine and the task restarts its execution from the beginning. Therefore our proposed algorithm gets lesser penalties when compared to non-preemptive scheduling with task migration algorithm because the migrated task will be check pointed and restarts the execution of a task from where the last checkpoint interval saved.

Throughput measures the number of tasks successfully completed from the set of tasks in the ready queue. Figure 5 show that our proposed algorithm has better throughput when compared with non-preemptive scheduling with task migration algorithm. Since higher priority task waiting for a longer duration of time it may misses its deadline. But in our proposed algorithm, higher priority tasks are taken for its execution whenever it reaches the ready queue.

Figure 6 depicts the graph for execution time of the task between two scheduling algorithms. In non-preemptive scheduling with task migration algorithm, the migrated task restarts its execution from the beginning and in our proposed algorithm, the task restarts it execution from

where the last checkpoint interval saved. This reduces the execution time of the task when compared to non-preemptive scheduling with task migration algorithm.

5. Conclusion

In cloud operating environment, various clients are plugged into the cloud to assess the resources which are priced and on-demand services. The resources are stored in the cloud server and multiuser can access these resources simultaneously. Hence scheduling plays a vital role in cloud computing. In traditional approach, a non-pre-emptive with task migration algorithm is to increases the execution time of the task because the migrated task executed from the beginning and the new task with highest priority cannot be taken for its execution until an executing task successfully completed so it will be waited for indefinite amount of time. In order to overcome this problem, a pre-emptive scheduling with checkpoint intervals algorithm is proposed. Whenever a new task arrives with the higher priority than the currently running task, then it will be preempted and again enter into the ready queue. The migrated task restarts its execution from where the last checkpoint interval saved. Therefore it minimizes the execution time of the tasks and it improves the overall system performance. Our simulation results show a better performance in the cloud environment while compared with the traditional algorithm.

References

1. Weiss, A. (2007). Computing in the clouds. *Networker*, 11(4), 16–25.
2. Singh, D., Singh, J. & Chhabra, A. (2012). Evaluating Overheads of Integrated Multilevel Checkpointing Algorithms in Cloud Computing Environment. *International Journal of Computer Network and Information Security*, 5(1), 29-38. Published Online June 2012 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijcnis (2012.05.04).
3. Burford, D. (2010). *Cloud Computing- A Brief Introduction*. LAD Enterprizes, Inc.
4. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. & Zaharia, M. (2005). *Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley Technical Report UCB/EECS-2009-28*, February 2009.
5. Shastry, P. M. M. & Venkatesh, K. (2010). Selection of a Checkpoint Interval in Coordinated Check-pointing Protocol for Fault Tolerant Open MPI. *International Journal on Computer Science and Engineering*, 2(6), 2064-2070.
6. Chtepen, M., Claeys, F. H. A., Dhoedt, B., Turck, F. D. & Demeester, P. (2009). *Adaptive Task Checkpointing And Replication: Toward Efficient Fault-Tolerant Grids.* IEEE Transactions on Parallel and Distributed Systems, February, 20(2), 180-190.
7. Paul, M. & Sanyal, G. (2011). Task-Scheduling in Cloud Computing using Credit Based Assignment Problem. *International Journal on Computer Science and Engineering*, October, 3(10), 3427-3431.
8. Paul, M., Samanta, D. & Sanyal, G. (2011). Dynamic job scheduling in cloud computing based on horizontal load balancing. *International Journal of Computer Technology and Application*, 2(5), 1552-1556.
9. Santhosh, R. & Ravichandran, T. (2012). Non-pre-emptive on-line scheduling of real-time services with task migration for cloud momputing. *European Journal of Scientific Research*, October, 89(1), 163-169.
10. Santhosh, R. & Ravichandran, T. (2013). *Pre-emptive Scheduling of On-line Real Time Services With Task Migration for Cloud Computing*. International Conference on Pattern Recognition, Informatics and Mobile Engineering.
11. Tayal, S. (2011). Tasks scheduling optimization for the cloud computing system. *International Journal of Advanced Engineering Sciences and Technologies*, 5(2), 111-115.
12. Rao, S., Rao, N. & Kumari, K. (2005-2009). Cloud computing: An overview. *Journal of Theoretical and Applied Information Technology*, 71-76.
13. Suen, T. T. Y. & Wong, J. S. K. (1992). Efficient task migration algorithm for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, July, 3(4), 488-499.
14. Paindaveine, Y. & Malojicic, D. S. (1996). *Process vs Task Migration*. Twenty Ninth Hawaii International Conference, (Vol. 1, pp. 636-645).