AENSI Journals
Australian Journal of Basic and Applied Sciences
ISSN:1991-8178

Journal home page: www.ajbasweb.com

Task Scheduling for Online Real Time Services with Multiple Resources Using RM Algorithm for Cloud Computing

[1]R. Santhosh and [2]T. Ravichandran

[1]*Research Scholar, Department of Computer Science and Engineering, Karpagam University Coimbatore, India.*
[2]*Principal, Hindusthan Institute of Technology, Coimbatore, India.*

ARTICLE INFO
Article history:
Received 8 August 2014
Received in revised form
12 September 2014
Accepted 25 September 2014
Available online 2 November 2014

Keywords:
Scheduling, Resource, Priority, Deadline, Waiting Time

ABSTRACT
Cloud Computing provides various services through virtualized resources over the internet. In the view of the fact that the cost rendered for the services used by the consumers are very high. So the cloud consumers are in the position to exploit these services in a proficient manner. Task scheduling plays an imperative role to finish the task within deadline in accordance with effective utilization of resources. Earlier, two approaches have been proposed for scheduling of real time services. Non Pre-emptive real time scheduling using checkpointing algorithm concentrates on reducing the execution time of the migrated tasks. A priority constrained pre-emptive scheduling algorithm gives more importance for high priority tasks. Both the above approaches provide the services using solitary resource. Therefore only limited numbers of tasks are executed in certain duration of time interval. In this regard, task scheduling for online real time services with multiple resources using resource matching (RM) algorithm is proposed. This algorithm gives preference to higher priority tasks. By executing various tasks with multiple resources, the waiting time of the tasks in the queue is minimized. Our simulation shows better results than the existing models.

To Cite This Article: R. Santhosh and T. Ravichandran., Task Scheduling for Online Real Time Services with Multiple Resources Using RM Algorithm for Cloud Computing. ***Aust. J. Basic & Appl. Sci.,*** *8(16): 296-301, 2014*

INTRODUCTION

Cloud Computing sets a unique mark in the field of computing. There is no boundary limit in this cloud environment for providing the services. The services can be varied from one cloud provider to another. The providers charges for the services used by the consumers. Scheduling of a task is the major factor to get maximum profit in this computing environment. The main goal of scheduling algorithm is to schedule the task properly among the resources and achieves high performance throughput.

II. Related Work:

Cloud providers offer data, application, hardware, software, server, storage and networks as resources, Weiss (2007). Cloud consumers can avail these resources and properly schedule their tasks in order to complete within the deadline. Successful completion of tasks achieves better throughput. The first author had already proposed two algorithms namely task migration algorithm and fixed checkpointing algorithm. In task migration algorithm, Santhosh and Ravichandran (2012) stated that, whenever the task misses its deadline during its execution, then the task will be migrated to another virtual machine and starts its execution. This reduces the potential loss. Therefore the remaining task in the queue will be executed as per scheduled otherwise the currently executing task continues its execution even though it misses its deadline. The migration of a task from one machine to another machine includes the following steps.

A. Task Migration Algorithm:

Step 1: Migration request is made to a remote node.
Step 2: Detach the task from the source node and place it in a migrating state.
Step 3: Communications are temporarily redirected.
Step 4: Processing states are pulled out from the source node.
Step 5: Processing states are moved to the remote node.
Step 6: Enable the communication channels once the migration completes at the remote node.

Corresponding Author: R. Santhosh, Research Scholar, Department of Computer Science and Engineering, Karpagam University Coimbatore, India.
E-mail: santhoshrd@gmail.com

Even though the task has been migrated, its starts its execution from the beginning. This increases the execution time of the migrated task. To overcome this problem, checkpoint intervals are allocated to each task using fixed checkpointing algorithm, Santhosh and Ravichandran (2013). Incompletion of the tasks and successful completion of the tasks are taken into account while calculating checkpointing interval. This allows the migrated task to start its execution from the last checkpoint interval saved. The fixed checkpointing algorithm is described in the subsequent steps.

B. Fixed Checkpointing Algorithm:

Step 1: Checkpoint intervals are allocated according to the size of the task.
Step 2: Save the execution of a task in the derived disk space for each interval.
Step 3: Compute the Fixed checkpoint intervals of a task.
Step 4: The migrated task and the pre-empted task will restarts its execution from the last saved checkpoint interval.

In this paper, we present a task scheduling for online real time services with multiple resources using resource matching algorithm. This algorithm executes the higher priority tasks preferably and minimizes the waiting time of the tasks in the normal queue.

III. Task Scheduling For Online Real Time Services With Multiple Resources Using Rm Algorithm For Cloud Computing:

In this section, we proposed a real time task scheduling algorithm. Scheduler plays an important role in allocating the right resources to the various tasks present in the queue. Here, the queue has been separated as normal queue and migration queue to execute the normal task and the migrated task. More number of machines will be allocated to process the normal queue relatively to process the migration queue. Scheduler executes the tasks in the queue in a pre-emptive manner.

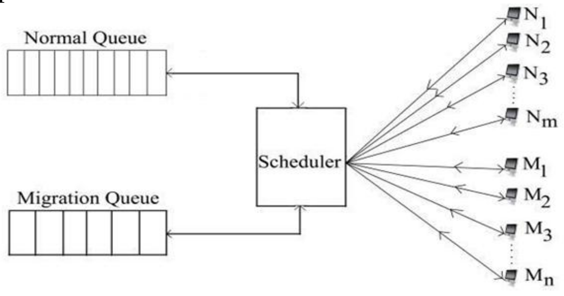

Fig. 1: Task Scheduling Architecture.

A. Task Scheduling Algorithm:

Step 1: Let N be the arrived tasks in the normal queue.
Step 2: Sort the tasks in the normal queue according to the priority.
Step 3: Checkpoint intervals will be allocated for the various tasks in the normal queue using fixed checkpointing algorithm.
Step 4: Scheduler generates the resource table using resource table generation algorithm.
Step 5: Each task in the normal queue is allocated with the resources by the scheduler using resource matching algorithm.
Step 6: The task will be processed by the following cases
Case1: (Empty Migration Queue, Empty Normal Queue)
Scheduler is ready to process the new task from the normal queue only.
Case2: (Empty Migration Queue, Load Normal Queue)
Scheduler allows using all the available resources to process the task from the normal queue.
Case3: (Load Migration Queue, Empty Normal Queue)
Scheduler allows using all the available resources to process the task from the migration queue.
Case4: (Load Migration Queue, Load Normal Queue)
Scheduler allocates more systems to process the normal queue and few systems to process the migration queue.
Step 7: Whenever a new task arrives with highest priority than the executing tasks in the normal queue, the scheduler pre-empts the executing task which has less priority.
Step 8: The pre-empted task again enters into the normal queue and sorted according to the priority.
Step 9: When the pre-empted task comes for its execution, it can start its execution from where the last check point interval saved.
Step 10: Whenever an executing task misses its deadline, the scheduler migrate the task to the migration queue and starts its execution from where the last check point interval saved. If the task does not have the resources

allocated to the migration queue, the scheduler checks for the available resources in the systems allocated to the normal queue. If the task needed resources matched with the systems, then the scheduler places that task in the normal queue.
Step 11: Whenever scheduler places the migrated task again to the normal queue, the task will be taken for its execution only when the resource needed for the task in the normal queue is free.
Step 12: If a task does not match with the systems allocated to the normal queue, the scheduler checks for the available resources allocated to migration queue. If the task needed resources matched with the systems, then the scheduler places that task in the migration queue.
Step 13: Whenever scheduler places the task directly in the migration queue due to insufficient resources allocated to the normal queue, the task gets allocated to the system immediately if it is free otherwise the executing task will be pre-empted. It again re-enters into the migration queue and processed once the normal task gets executed. The pre-empted task again starts its execution from where the last check point interval saved.
Step 14: When a task is waiting for the resources over a period of time in both normal queue and migration queue, the scheduler allocates the resources for the subsequent tasks only if the required resources are available to process the task.

The tasks are sorted according to the priority in the queue. Checkpoint intervals are allocated to the tasks, Santhosh and Ravichandran (2013). With the connected resources, resource table has been generated. Successful completion of the task will depend on matching the resources. Because every task in the queue is allocated with the proper resources otherwise the task does not get enough resources for its execution and leads to potential loss to the consumers. In this algorithm, scheduler takes the responsibility of matching the resources for the tasks using resource matching algorithm. The tasks in both of the queues are executed in a pre-emptive manner. Therefore higher priority tasks are given more preference than the other tasks present in the queue. Whenever a task misses its deadline, the scheduler migrate the task to the migration queue, Santhosh and Ravichandran (2012). Then the migrated task is allocated with the resources by the scheduler and starts its execution from the last check point interval saved.

B. Resource Table Generation Algorithm:

Step 1: Let the connected systems for the normal queue will be N_1, N_2, N_3....... N_m and for the migration queue will be M_1, M_2, M_3....... M_n
Step 2: Gather the software and hardware details for the connected systems
Step 3: Generate the resource table with the collected information using unique resource software id and hardware id
Step 4: Scheduler can add new resources to the table whenever needed.

The scheduler collects the information about the software and hardware for the connected systems. With that information resource table is generated and unique id is created for the resources. Scheduler can add and modify the resources through the unique id at any point of time.

C. Resource Matching Algorithm:

Step 1: Select the task from the normal queue
Step 2: Gather the software and hardware requirements for the task to be processed.
Step 3: Match the needed requirements with the available resources using resource table generation algorithm
Step 4: The task is allocated with the right system and gets executed.

Using the generated resource table, the scheduler can match the right resources for the tasks present in the queue. The scheduler verifies the requirements of a task and gets allocated with the resources. Then the task starts its execution with the allocated machine.

IV. Simulation Results:

Our proposed algorithm is compared with non pre-emptive checkpointing algorithm and priority constrained pre-emptive checkpointing algorithm. Non pre-emptive checkpointing algorithm executes the task in a non pre-emptive manner and does not give preference to the higher priority tasks whereas priority constrained checkpointing algorithm executes the task in a pre-emptive fashion and gives inclination to higher priority tasks. Both the algorithms execute the tasks with a single resource. This increases the waiting time of the task. Our proposed algorithm significantly reduces the waiting time of the task as well as gives preference to execute the higher priority tasks. We simulated and compared our algorithm with the existing algorithms with thirty set of tasks based on impending gain, throughput and waiting time.

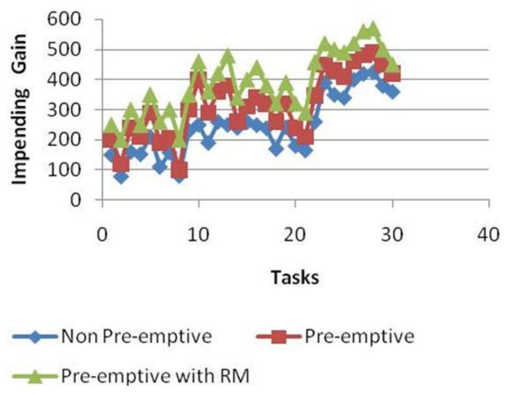

Fig. 2: Comparison of Impending Gain between Non Pre-emptive, Pre-emptive, Pre-emptive with RM.

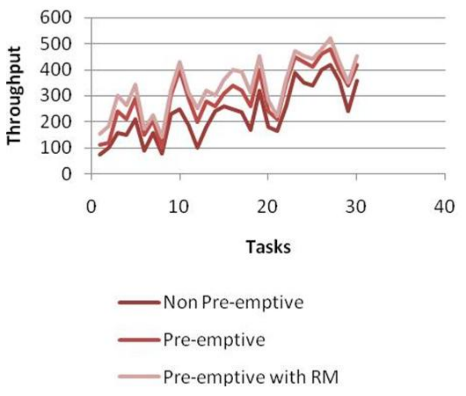

Fig. 3: Comparison of Throughput between Non Pre-emptive, Pre-emptive, Pre-emptive with RM.

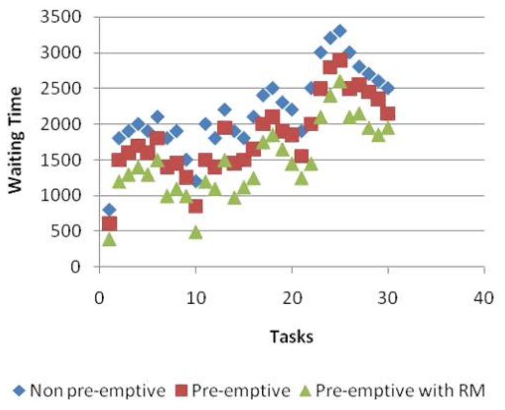

Fig. 4: Comparison of Waiting Time between Non Pre-emptive, Pre-emptive, Pre-emptive with RM.

The graph for impending gain is shown in the Fig.2. The gain can be achieved by finishing the task within the deadline. In non pre-emptive checkpointing algorithm, higher priority tasks are waiting for a longer duration of time. Even though priority constrained pre-emptive checkpointing algorithm executes the higher priority tasks first, the task should be waiting for longer time in the queue due to the execution of a task with a single resource. Therefore the task cannot be finished within the deadline and leads to potential loss. Our algorithm radically attains more gain by overcoming the problems faced by the two approaches.

Successful completion of tasks can be calculated from the arrived set of tasks in the queue. This provides the throughput value for the various tasks present in the queue. Fig.3. shows that our proposed algorithm achieves better throughput while comparing with other two algorithms.

Fig.4. shows the graph for waiting time of a task. Both the non pre-emptive checkpointing algorithm and priority constrained pre-emptive checkpointing algorithm uses a single resource for its task execution. This increases the waiting time of the task. Our proposed algorithm executes the task with multiple resources and minimizes the waiting time of the task.

V. Conclusion:

Nowadays, the cloud computing technology proving its efficiency rapidly by providing various services to the cloud consumers. This provides a healthy environment for the consumers who are not able to purchase the software and hardware resources. The consumers have to pay for accessing the resources over the internet. The services can be accessed through private, public or hybrid cloud. For effective use of services in the cloud, scheduling is the major criterion for allocating the correct resources for the various tasks. In this developing environment, day by day the cloud consumers are extensively growing large to avail the services. In older approaches the services have been given through a single resource. This increases the waiting time of the task to get the resources for its execution. To overcome this problem we proposed a task scheduling for online real time services with multiple resources using resource matching algorithm. This algorithm provides the services through multiple resources and also gives leaning to higher priority tasks. Our simulation results show better performance than the existing approaches.

REFERENCES

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, 2009. *Above the clouds: A Berkeley view of cloud computing,*UC Berkeley.

Bartal, Y., S. Leonardi, A. Marchetti-Spaccamela, J.S. Gall and L. Stougie, 1996. *Multiprocessor scheduling with rejection,* In Proceedings of SODA, pp: 95-103.

Casati, F. and M. Shan, 2001. *Definition, execution, analysis and optimization of composite e-service,* IEEE Data Engineering.

Clark, R.K., 1990. *Scheduling dependent real-time activities* PhD thesis, Carnegie Mellon University.

Dilbag Singh, Jaswinder Singh, AmitChhabra, 2012. *Evaluating Overheads of Integrated Multilevel Checkpointing Algorithms in Cloud Computing Environment*‖, I. J. Computer Network and Information Security, 5, 29-38 Published Online June in MECS (http://www.mecs-press.org/)DOI:10.5815 /ijcnis.2012.05.04.

Idawaty Ahmad, S. Shamala, M. Othman and Muhammad Fauzan Othman, 2008. *A Preemptive Utility Accrual Scheduling Algorithm for Adaptive Real Time System,* IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.5.

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

Knorr, E. and G. Gruman, 2010. *What cloud computing really means,* http://www.infoworld.com.

Kuno, H., 2000. *Surveying the e-services technical landscape, In 2nd International Workshop on Advanced Issues of Ecommerce and Web-based Information Systems.*

Li, P., 2004. *Utility Accrual Real-Time Scheduling: Models and Algorithms,* PhD thesis, Virginia Polytechnic Institute and State University.

Li, P., 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.

Locke, C.D., 1986. *Best-effort decision making for real-time scheduling,* PhD thesis, Carnegie Mellon University.

Mallikarjuna Shastry, P.M., K. Venkatesh, 2010. *Selection of a Checkpoint Interval in Coordinated Checkpointing Protocol for Fault Tolerant Open MPI,*‖ (IJCSE) International Journal on Computer Science and Engineering, 02(06): 2064-2070.

Maria Chtepen, Filip H.A. Claeys, Bart Dhoedt, Filip De Turck, Piet Demeester, 2009.*" Adaptive Task Checkpointing And Replication: Toward Efficient Fault-Tolerant Grids*‖, IEEE Transactions On Parallel And Distributed Systems, 20(2).

Mehdi Kargahi, Ali Movaghar, 2006. *A method for performance analysis of Earliest Deadline First Scheduling policy, The Journal of Supercomputing,* 37: 197-222.

Santhosh, R., T. Ravichandran, 2012. *Non-Pre-emptive 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, pp: 163-169.

Santhosh, R., T. Ravichandran, 2013. *A Priority constrained Pre-Emptive scheduling of online Real time services with Fixed Checkpoint Intervals for Cloud Computing*‖, International Journal of Distributed and Cloud Computing, 1(1).

Santhosh, R., T. Ravichandran, 2013. *Non-Preemptive Real Time scheduling using checkpointing algorithm for cloud computing*‖, International journal of Computer Applications (0975 - 8887), 80(9).

Santhosh, R., T. Ravichandran, 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 (PRIME), 978-1-4673-5845-3/13,February 21-22.

Shou Liu, Gang Quan, Shangping Ren, 2010. *On-line Scheduling of real time services for cloud computing,* In IEEE World congress on services.

Weiss, A., 2007. *Computing in the clouds*, Networker, 11(4): 16-25.

Wu, H., 2005. *Energy-Efficient utility Accrual Real-Time Scheduling*, PhD thesis, Virginia Polytechnic Institute and State University.

Wu, H., B. Ravindran and E. Jensen, 2004. *On the joint utility accrual model*, pages 124.

Wu, H., B. Ravindran and E.D. Jensen, 2004. *Utility accrual scheduling under joint utility and resource constraints*, Pages 307.

Wu, H., B. Ravindran and E.D. Jensen, 2010. *Energy-efficient, utility accrual real-time scheduling under the unimodal arbitrary arrival model*, In ACM Design, Automation, and Test in Europe.

Wu, H., U. Balli, B. Ravindran and E. Jensen, 2005. *Utility accrual real-time scheduling under variable cost functions*, Pages 213-219.

Yu, Y., S. Ren, N. Chen and X. Wang, 2010. *Profit and penalty aware (pp-aware) scheduling for tasks with variable task execution time*., In SAC2010 - Track on Real-Time System ,RTS'.