Non-Preemptive Real Time Scheduling using Checkpointing Algorithm for Cloud Computing

R. Santhosh
Research Scholar
Department of Computer Science and Engineering
Faculty of Engineering, Karpagam University

T. Ravichandran
Principal
Hindusthan Institute of Technology

ABSTRACT
This paper focuses on providing a solution for online real time services using non-preemptive scheduling algorithm in order to minimize the execution time of the migrated tasks. Earlier, a non-preemptive scheduling with task migration algorithm is used to minimize the penalty. Whenever a task misses its deadline, it will migrate the task to another virtual machine and starts its execution from the beginning. Therefore it increases the execution time of the migrated task. In order to overcome this problem, a non-preemptive real time scheduling using checkpointing algorithm is proposed to minimize the execution time of the migrated tasks and minimizes the penalty even better by earlier completion of migrated tasks. This improves the overall system performance. Our simulation results outperform the older approaches based on the similar model.

Keywords
Non-preemptive, Migration, checkpoint, Potential profit, Potential loss, Deadline, Virtual machine, Vital point.

1. INTRODUCTION
Cloud computing is a network based computing, where more number of systems is connected in private, public or hybrid network (M. Armbrust et al. 2009).Cloud Service Provider makes the software even more widen as a service and they consume virtualized resources such as memory, I/O, storage and computational capacity as a service in which pay only for the resources that they use (F. Casati and M. Shan 2001).

Scheduling is the method for accessing the quality of service of the system in real time applications or services. The time factor plays a major role in analyzing the quality of service. With this regard, a performance metric for cloud computing can be the sum of utility that is ensued by handling all real time service requests.

(Liu et al., 2010) measures the profit when completing a job in time and the penalty when a job is aborted or discarded. The performance of the system is affected when a task is aborted or discarded because it consumes system sources including network bandwidth, storage space and processing power.

In this paper, we present a non-preemptive real time scheduling
using checkpointing algorithm. Our algorithm sensibly migrate the aborted tasks and start its execution from where the last checkpoint interval saved. This minimizes the penalty and consequently achieves better performance.

2. PRELIMINARY
In this paper, the arrived real time tasks αt = {T1, T2, T3,……..,Tn} are arranged in a sequence and defined using the following parameters.

Let Ai –Arrival Time of the tasks, t- Current Time of the tasks, τo –Current Task being executed, Pt –Predicted Starting Time, Bt –Best Case Execution Time, Wt –Worst Case Execution Time, Dt-Deadline of tasks, Pp-Potential Profit, Pl-Potential Loss, Et-Expected Gain, C-Checkpoint interval, N-Number of Checkpoints, T-Time required for saving each checkpoint interval, I-Incompletion of tasks, S-Completed tasks

The real time tasks which are arranged leads to profit or penalty function, therefore the task which attains the penalty acquire potential loss. The main cause for the potential loss of a task is due to missing its deadline. Whenever the task misses its deadline, it will migrate the task to another virtual machine and starts its execution from the beginning. Therefore it increases the execution time of the migrated task. With this problem, we developed an online non-preemptive real time scheduling using checkpointing algorithm in order to maximize the total utility and decrease the execution time of the migrated tasks.

3. NON- PREEMPTIVE REAL TIME SCHEDULING USING CHECKPOINTING ALGORITHM
In this section, we present a non-preemptive real time scheduling using checkpointing algorithm to provide the solution for minimizing the execution time of the migrated tasks. The execution of a task may get potential profit or potential loss. The penalty will degrade the overall computing performance. Checkpoint intervals will be allocated for the task with highest expected gain in the queue and ready for its execution. Whenever the task misses its deadline, the execution of the task will be aborted, it will be migrated to another virtual machine and start its execution from where the last checkpoint interval saved. Then the task which has the highest expected gain in the queue is allocated with checkpoint intervals and starts its execution.

Our non-pre-emptive scheduling algorithm works at scheduling points that include the arrival of a task, the completion of the current task and the vital point of the current task. The detailed algorithm is described in algorithm1.

Algorithm 1: Non-Preemptive Online Scheduling Algorithm

Step1: Letαt= {T1, T2, T3,……,Tn} be the accepted tasks in the ready queue. Let the current time be t and $\tau 0$ be the current task being executed**.**

Step2: The difference between the expected finishing time of the current task being executed and the current time gives the predicted starting time.

Step3: Calculate the tasks expected utility at the predicted starting time through by adding the predicted starting time with the expected finish time of all the tasks in the queue.

Step4: The deadline execution time of the tasks can be calculated by
Dt = (Bt) – (Ai - Pt)

Step5: The potential profit of the task can be calculated by
Pp = (Dt + (Ai - Pt)) – (Bt + (Ai - Pt))

Step6: The potential loss of the task can be calculated by
Pl = (Wt) – (Dt)

Step7: The difference between the potential profit and the potential loss gives the expected gain.
Et = Pp- Pl

Step8: The list of tasks in the ready queue is sorted based on their expected gain.

Step9: The task which has the highest expected gain in the queue is allocated with checkpoint intervals and starts its execution.

Step10: When the current executing task reaches its vital point, it will be migrated to another virtual machine.

Step11: A request of migration is made to a remote node.

Step12: The task is detached from its source node and put it to be in a migrating state.

Step13: Redirect the communication temporarily.

Step14: Extract the processing states from the source node.

Step15: Transfer the processing states to the remote node.

Step16: Communication channels are enabled after migration at the remote node.

Step17: The task's migration completes, once all of the states has been transferred from the source node to remote node.

Step18: The migrated task restarts its execution from where the last checkpoint interval saved.

Step19: Choose the task which has the highest expected gain from the ready queue is allocated with checkpoint intervals and starts its execution.

Step20: If a new task arrives, put the task at the head of the ready queue, then sort the tasks in the ready queue based on the recalculated expected gain.

Whenever the current task reaches the vital point, the task is instantly migrated to another virtual machine. A migration request is issued to a remote node during migration. A task is detached from its source node by suspending its execution and affirms it to be in a migrating state. Temporarily redirecting communication channels by queuing up arriving messages directed to the migrated task and by delivering them to the task after migration. The task is extracted and is typically retained on the source node until the end of migration. A destination node instance is created into which the transferred state will be imported. After that, destination node will be promoted into regular 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. The migrated task starts its execution from where the last checkpoint interval saved.

Then the next task with the highest expected gain is allocated with checkpoint intervals and then it is selected for execution. Upon the completion of the current task, choose the task from the ready queue which has the highest expected gain, allocate checkpoint intervals and starts its execution. After the selection of new tasks, recalculate the expected gain for rest of the tasks in both cases.

If a new task arrives, it is first inserted at the head of the ready queue and calculates the expected gain. Based on the expected gain 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 expected gain.

Algorithm 2: 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 checkpoint intervals of a task.

$$C = \sqrt{Z},\ 2\sqrt{Z}+T,\ 3\sqrt{Z}+T \ldots\ldots\ldots\ N\sqrt{Z}+(N-1)T$$

Where Z= (2*T*I) / (S+I)

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

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

4. SIMULATION RESULTS

Our proposed algorithm is simulated to improve the overall system performance in order to achieve the total utility. We compared our proposed algorithm with the non-preemptive scheduling with task migration algorithm based on the constraint which includes total profit, total penalty, execution time and throughput. Non-preemptive scheduling with task migration algorithm is evaluated by its utility gain after the task has been migrated to another virtual machine when it misses its deadline.

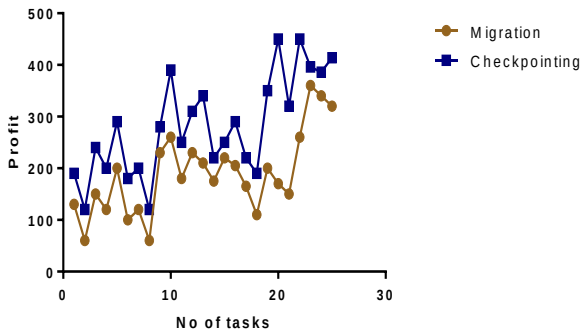

Figure 1: Comparison of Profit between Checkpointing and Migration

Successful completion of the tasks in the ready queue gives the total profit. The profit value for checkpointing is compared with migration algorithm. While comparing, migration algorithm measures the profit value when the job is completed in time. As shown in figure 1, the profit value can be maximum for the checkpointing algorithm when comparing with migration algorithm.

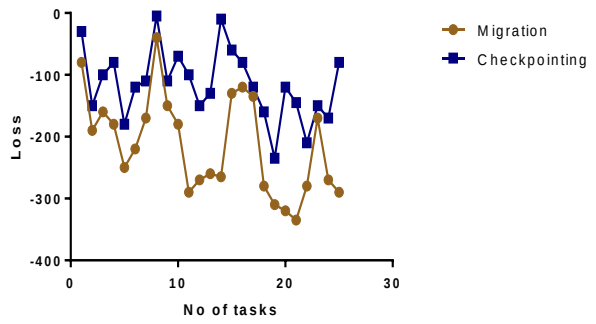

Figure 2: Comparison of Loss between Checkpointing and Migration

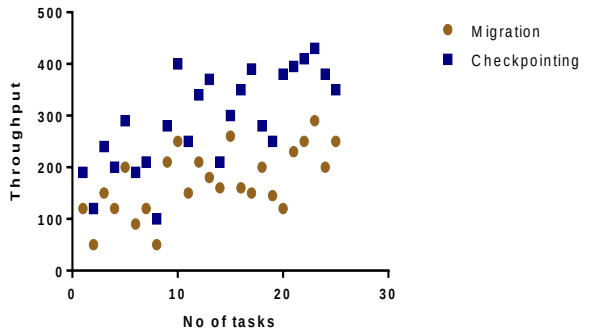

Figure 3: Comparison of Throughput between Checkpointing and Migration

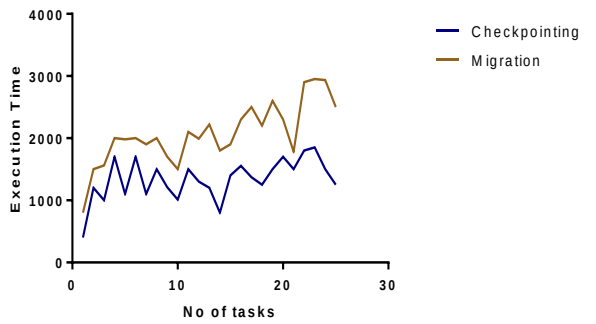

Figure 4: Comparison of Execution Time between Checkpointing and Migration

Figure 2 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 migration algorithm. The results show that our proposed algorithm attains fewer penalties. Because in migration, the task will be aborted when it misses its deadline and migrated to another virtual machine. It starts its execution from the beginning.

Throughput measures the number of tasks successfully completed from the set of tasks in the queue. Figure 3 shows that our proposed algorithm has better throughput when compared with non-preemptive scheduling with task migration algorithm.

Figure 4 shows 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.

5. CONCLUSION

With the rising trends of cloud computing, Scheduling facilitates the infrastructure to merge on very large scale of computing resources for cloud computing, it is essential that the profit is not only the main constraint but also the cost of task execution should be taken into account during the resource management. In existing approach, the system takes a profit, if the tasks successfully completes its deadline and suffers a penalty, whenever the task starts its execution from the beginning in the migrated virtual machine. Therefore, the system performance can be degraded. In order to address these problems, this paper is proposed to minimize the execution time of the migrated tasks and minimizes the penalty even better. Whenever an executing task reaches its vital point, a task will be migrated to new virtual machine and start its execution from where the last checkpoint interval saved. It improves the total utility. Our simulation result shows that our proposed algorithm can significantly outperforms the Non Preemptive scheduling with task migration algorithm. There a quite a few interesting research problem for our future work is to decrease the execution time of the task and incorporating the preemption into our scheduling method.

6. REFERENCES

[1] A. Weiss (2007). Computing in the clouds, Networker, 11(4):16–25.

[2] C. D. Locke (1986). Best-effort decision making for real-time scheduling, PhD thesis, Carnegie Mellon University.

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

[4] 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.

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

[6] F. Casati and M. Shan (2001)., Definition, execution, analysis and optimization of composite e-service, IEEE Data Engineering.

[7] H. Wu (2005)., Energy-Efficient utility Accrual Real-Time Scheduling, PhD thesis, Virginia Polytechnic Institute and State University.

[8] H.Wu, U. Balli, B. Ravindran, and E. Jensen (2005)., Utility accrual real-time scheduling under variable cost functions, Pages 213–219.

[9] H. Wu, B. Ravindran, and E. Jensen (2004)., On the joint utility accrual model, pages 124.

[10] H. Wu, B. Ravindran, and E. D. Jensen (2004)., Utility accrual scheduling under joint utility and resource constraints, Pages 307.

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

[12] H. Wu, 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.

[13] 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.

[14] MallikarjunaShastry P.M. ,K. Venkatesh," Selection of a Checkpoint Interval in Coordinated Checkpointing Protocol for Fault Tolerant Open MPI," (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 06, 2010, 2064-2070.

[15] Maria Chtepen, Filip H.A. Claeys, Bart Dhoedt, Filip De Turck, Piet Demeester," Adaptive Task Checkpointing And Replication: Toward Efficient Fault-Tolerant Grids", IEEE Transactions On Parallel And Distributed Systems, Vol. 20, No. 2, February 2009.

[16] Mehdi Kargahi, Ali Movaghar (2006).,A method for performance analysis of Earliest Deadline First Scheduling policy, The Journal of Supercomputing, 37, 197–222, 2006.

[17] M. Armbrust, 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.

[18] P. Li (2004)., Utility Accrual Real-Time Scheduling: Models and Algorithms, PhD thesis, Virginia Polytechnic Institute and State University.

[19] 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.

[20] R. K. Clark (1990)., Scheduling dependent real-time activities PhD thesis, Carnegie Mellon University.

[21] R.Santhosh, T.Ravichandran, "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, 2012, pp.163-169.

[22] R.Santhosh, T.Ravichandran," 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 2013.

[23] ShouLiu , Gang Quan, ShangpingRen (2010).,On-line Scheduling of real time services for cloud computing, In IEEE World congress on services.

[24] Y.Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. S. gall, and L. Stougie (1996)., Multiprocessor scheduling with rejection, In Proceedings of SODA, pages 95 – 103.

[25] Y. Yu, 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'2010).

7. AUTHOR'S PROFILE

R.Santhosh – He received his B.Tech degree in information technology from K.S.R College of Technology in 2006, M.E degree in Software Engineering from Sri Ramakrishna Engineering College in 2009, M.B.A in Education Management from Alagappa University in 2011 and pursing Ph.D in Computer Science and Engineering at Karpagam University. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering at Karpagam University.

Professor Dr.T.Ravichandran – He received the B.E degree from Bharathiar University, Tamilnadu, India and M.E degree from Madurai Kamaraj University, Tamilnadu, India in 1994 and 1997, respectively and Ph.D degree from the Periyar University, Salem, India, in 2007. He is currently the Principal of Hindustan Institute of Technology, Coimbatore, TamilNadu, India. Before joining Hindustan Institute of Technology, Professor Ravichandran has been a Professor and Vice Principal in Vellalar College of Engineering & Technology, Erode, Tamilnadu, India. His research interests include theory and practical issues of building distributed systems, Internet computing and security, mobile computing, performance evaluation, and fault tolerant computing. Professor Ravichandran is a member of the IEEE, CSI and ISTE. Professor Ravichandran has published more than 30 papers in refereed international journals and refereed international conferences proceedings.