World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:3, 2015

Accrual Based Scheduling for Cloud in Single and Multi Resource System: Study of Three Techniques

R. Santhosh, T. Ravichandran

Abstract**—**This paper evaluates the accrual based scheduling for cloud in single and multi-resource system. Numerous organizations benefit from Cloud computing by hosting their applications. The cloud model provides needed access to computing with potentially unlimited resources. Scheduling is tasks and resources mapping to a certain optimal goal principle. Scheduling, schedules tasks to virtual machines in accordance with adaptable time, in sequence under transaction logic constraints. A good scheduling algorithm improves CPU use, turnaround time, and throughput. In this paper, three real-time cloud services scheduling algorithm for single resources and multiple resources are investigated. Experimental results show Resource matching algorithm performance to be superior for both single and multi-resource scheduling when compared to benefit first scheduling, Migration, Checkpoint algorithms.

Keywords**—**Cloud computing, Scheduling, Migration, Checkpoint, Resource Matching.

I. Introduction

CLOUD computing is an upcoming technology. To increase working in cloud computing environments efficiently, job scheduling is performed to gain maximum profit [1]. Cloud computing is based on some attributes, massive scalability, multi-tenancy (shared resources), elasticity, pay as you go, resources self-provisioning, security assessment, policies, and physical security. It makes advances in processors, disk storage, virtualization technology, broadband Internet connection, and fast servers to make the cloud a compelling solution.

The job scheduling algorithm's advantage is achieving high performance computing and best system throughput [2]. Conventional job scheduling algorithms cannot ensure scheduling in cloud environments. Task scheduling is classified into 2 categories: preemptive scheduling and non-preemptive scheduling. Under non-preemptive scheduling, a higher priority task is scheduled after completion of current task. A scheduling discipline is non-preemptive when once a process is given to a CPU it cannot be taken away.

Short jobs wait for longer jobs in non-preemptive system, but overall treatment of processes is fair. Non-preemptive scheduling algorithms are not difficult to implement and exhibit lower overhead at run-time. Non-preemptive scheduling on a uni-processor ensures exclusive access to shared resources and data, eliminating need for synchronization and its overhead [3]. Scheduling tasks without preemption is a theoretical basis for general tasking models including shared resources.

R. Santhosh is Research Scholar, Karpagam University, Tamilnadu, India (e-mail: cse.r.santhosh@gmail.com).

T. Ravichandran is Principal, Hindusthan Institute of Technology, Tamilnadu India.

Response times are predictable as incoming high priority jobs cannot displace waiting jobs in non-preemptive systems. In such scheduling, a scheduler executes jobs in 2 situations: When a process switches from running state to waiting state and when a process is terminated [4], [5]. Non-preemptive scheduling has advantages of accurate response time analysis, implementation ease, no synchronization overhead, and reduced stack memory needs.

Non-preemptive scheduling is used in lightweight multi-tasking kernels and has been shown to be beneficial in multimedia applications. For scheduling of real time tasks, Rate Monotonic and Earliest Deadline First (EDF) are two well-known scheduling algorithms under which execution of tasks is based on its period of arrival or deadline as well [15]. Rate Monotonic algorithm works with Static Priority Scheduling (offline tasks), and EDF algorithm is with Dynamic Priority Scheduling (online tasks).

The arrival of tasks in a particular system can be periodic, aperiodic or sporadic. Mostly systems set aside the arrival of tasks periodically because the period of task arrival is fixed, and these tasks are able to meet their respective deadline. A checkpoint is a local state of a job saved on stable storage [16]. By periodically executing the checkpoint, status of a process at consistent intervals can be saved. If there is a failure, computation can be resumed from the earlier checkpoints, thus, avoiding restarting execution from the beginning.

Rollback recovery is the process of restarting computation by going back to a consistent state. In cloud computing environment, as the nodes in data centers do not share memory, it is required to transfer load of failed node to different nodes in case of any sort of failure. In this work, the task migration based scheduling and checkpoint based scheduling are investigated for scheduling on multiple resources and compared with Earliest Deadline First Algorithm. In this paper, Section II reviews related work; Section III explains methods used for scheduling. Section IV discusses experimental results for scheduling with a single processor and for 4 machines. Section V concludes the work.

II. Literature Survey

The problem of scheduling checkpoints of sequential jobs in a Desktop Grids context including volunteered distributed resources was studied by [6]. A checkpoint scheduling algorithm provably optimal for discrete time when failures obey any general probability distribution was crafted.

Simulations with parameters based on real-world systems showed the optimal strategy scales and outperformed other strategies regarding check pointing costs and batch completion times.

An Event Based Check-pointing tool (EBC), where user's checkpoint and restart running programs in cloud systems introduced by [7] is based on event-driven architecture and integrated into cloud infrastructures like OpenNebula easily. EBC supports sequential programs and MPI programs. Also, any C&R included in MPI libraries was reused in EBC and is independent of MPI library versions. EBC is a tool which supports migration and scheduling in cloud systems.

A new scheduling approach to provide a solution for online scheduling problem using IaaS model offered by cloud computing was presented by [8]. The task in traditional approaches is scheduled non- preemptively with 2 types of Time Utility Functions (TUFs) - profit time utility function and penalty time utility function. A task with highest expected gain is executed. A preemptive online scheduling algorithm for cloud computing environment was proposed to minimize response time and to improve task efficiency in the new approach. When a task misses its deadline, it is sent to another virtual machine thereby improving overall system performance and increasing total utility. Simulation results outperformed conventional scheduling algorithms like EDF and a similar model based earlier scheduling approach.

A Migration-based Elastic Consolidation Scheduling (MECS) mechanism to automate elastic resource scaling for cloud systems was proposed by [9]. Different from earlier researches, dynamic workload fluctuation and Virtual Machine (VM) migration overhead were considered. An online resource demand predictor was developed - an ARIMA-based VM resource demand state predictor - achieved adaptive resource allocation for cloud applications. A migration-based elastic consolidation scheduling heuristic dynamically consolidated VMs with adaptive resource allocation to reduce number of physical machines. Experiments showed that the new scheduling realized elastic resource allocation with acceptable effect on SLAs.

Two exact algorithms for energy efficient VMs scheduling in cloud data centers was presented by [10]. Energy aware allocation and consolidation modeling to reduce overall energy consumption resulted in combining optimal allocation algorithm and a consolidation algorithm relying on VMs migration at service departures. The new migration goes beyond current state of the art reducing the number of migrations for consolidation and energy consumption in one algorithm with a set of valid inequalities and conditions. Results showed benefits of combining allocation and migration algorithms demonstrating their ability to achieve huge energy savings while maintaining feasible convergence times compared to a best fit heuristic.

A comprehensive availability model for 2 different types of rejuvenation scheduling based on live migration mechanism, one with test before migration, and other without was proposed by [11]. Five scenarios were evaluated with distinct time intervals for triggering rejuvenation. The goal is to explain benefits from using this rejuvenation technique, and understanding the differences between both approaches. Results showed that use of a schedule with a checking mechanism before rejuvenation ensures great system availability improvement.

III. Methodology

In this section, task migration algorithm, checkpoint algorithm, and the task scheduling algorithm is briefly discussed. All the techniques are accrual based techniques, and hence the utility function becomes the most important Quality of Service parameter.

A. Task Migration

Migration from source processors to destination processors increases system efficiency. Migration algorithms impact system performance greatly [12]. Task Migration time from source sub network to destination sub network and task delay time are important criteria for efficiency assessment. Task Migration Algorithms are [13]:

Step1. Migration request is made to remote node.
Step2. Detach task from source node and place it in migrating state.
Step3. Communications are redirected temporarily.
Step4. Processing states are pulled out from source node.
Step5. Processing states are moved to remote node.
Step6. Enable communication channels when migration is completed at remote node.

B. Checkpoint

Checkpoints are taken using fixed checkpoint interval method or variable checkpoint interval method. If using fixed checkpoint interval method, checkpoint interval size is same between two successive checkpoints and checkpoint interval size does not need to be uniform between two successive checkpoints in the incremental or variable checkpoint interval method [14]. A Fixed Checkpoint Algorithm is given below:

Step1. Checkpoint intervals are allotted according to task size.
Step2. Save task execution in derived disk space for every interval.
Step3. Compute a task's Fixed checkpoint intervals.
Step4. Migrated task and pre-empted task restart execution from last saved checkpoint interval.

C. Resource Matching Scheduling Algorithm

Resource Matching (RM) Scheduling Algorithm

The steps of RM algorithm are:

Step1. Let N be arrived tasks in a normal queue.
Step2. Sort tasks in normal queue according to priority.
Step3. Checkpoint intervals are allocated for various tasks in normal queue using fixed checkpoint algorithm.
Step4. Scheduler generates resource table using resource table generation algorithm.
Step5. Every task in a normal queue is allocated with resources by scheduler using RM algorithm.
Step6. Task is processed by the subsequent cases

Case 1. (Empty Migration Queue, Empty Normal Queue) Scheduler ready to process new task from normal queue only.
Case 2. (Empty Migration Queue, Load Normal Queue) Scheduler allows using all available resources to process task from normal queue.
Case 3. (Load Migration Queue, Empty Normal Queue) Scheduler allows using all available resources to process task from migration queue.
Case 4. (Load Migration Queue, Load Normal Queue) Scheduler allocates more systems to process normal queue and few systems to process migration queue.
Step7. When a new task arrives with highest priority than executing tasks in normal queue, scheduler pre-empts executing task with less priority.
Step8. Pre-empted task enters into normal queue again and is sorted according to priority.
Step9. When pre-empted task comes for execution, it starts its execution from last check point interval saved.
Step10. When an executing task misses deadline, scheduler migrates it to migration queue and starts execution from last check point interval saved. If task lacks resources allocated to migration queue, scheduler checks for available resources in systems allocated to normal queue. If task needs resources matched with systems, then scheduler places that task in normal queue.
Step11. When scheduler places migrated task again in normal queue, the task is taken for execution only when resource needed for task in the normal queue are free.
Step12. If task does not match with systems allocated to normal queue, scheduler checks for available resources allocated to migration queue. If task needs resources matched with systems, then scheduler places that task in migration queue.
Step13. When scheduler places a task directly in migration queue due to insufficient resources allocated to normal queue, task gets allocated to a system immediately if free, otherwise executing task is pre-empted. It re-enters into migration queue and is processed when normal task is executed. The pre-empted task starts execution from last check point interval saved.
Step14. When a task awaits resources in normal and migration queues, scheduler allocates resources for subsequent tasks only if required resources are available to process it.

IV. Experimental Results

The main goal of this work is to investigate the existing Task Migration and Checkpoint algorithm for multiple resources. Experiments were conducted in two scenarios, the first with single processor and the second scenario with two datacenter consisting of four virtual machines. Twenty-five runs were conducted each for Benefit first scheduling, Migration, Checkpoint, and RM methods. Figs. 1-3 show the results of single processor scheduling and Figs. 4-6 represent the four resource results. The parameters measured are the utility gain, the utility loss, and the total utility. Fig. 1 shows the utility gain function for all the four techniques.

The RM method increased the utility gain by 10. 0928% when compared with Migration. The checkpoint method increased utility gain by 8.3255% than migration. Fig. 2 shows the utility loss function.

The Checkpoint method decreased utility loss by 19.741% than migration method. The migration method decreased utility loss by 27.2472% than benefit first search method. Fig. 3 shows the total utility function given by

$$Total\_utility=Utility\_gain+Utility\_loss$$

Fig. 4 shows the utility gain when four resources are used.

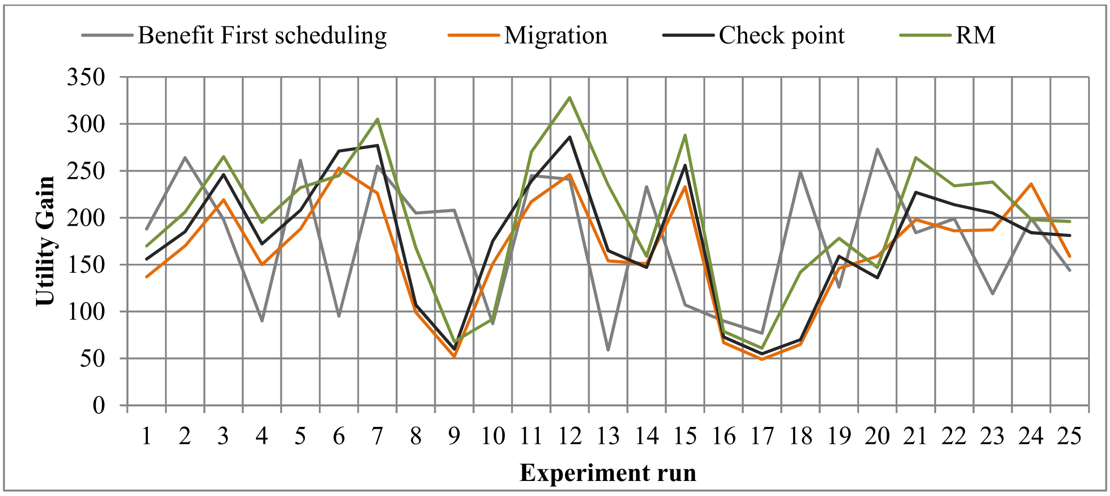

Fig. 1 Utility gain in single resource system

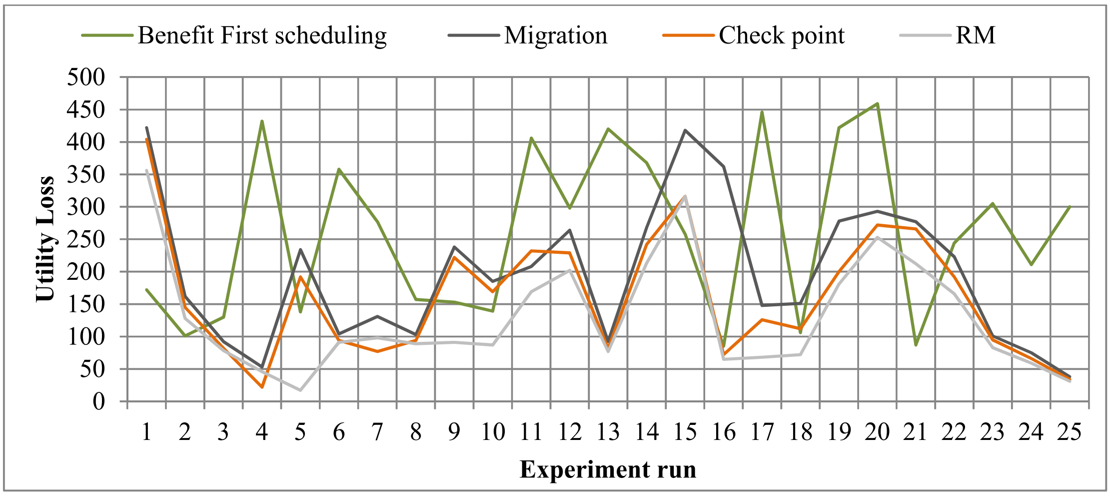

Fig. 2 Utility Loss in single resource system

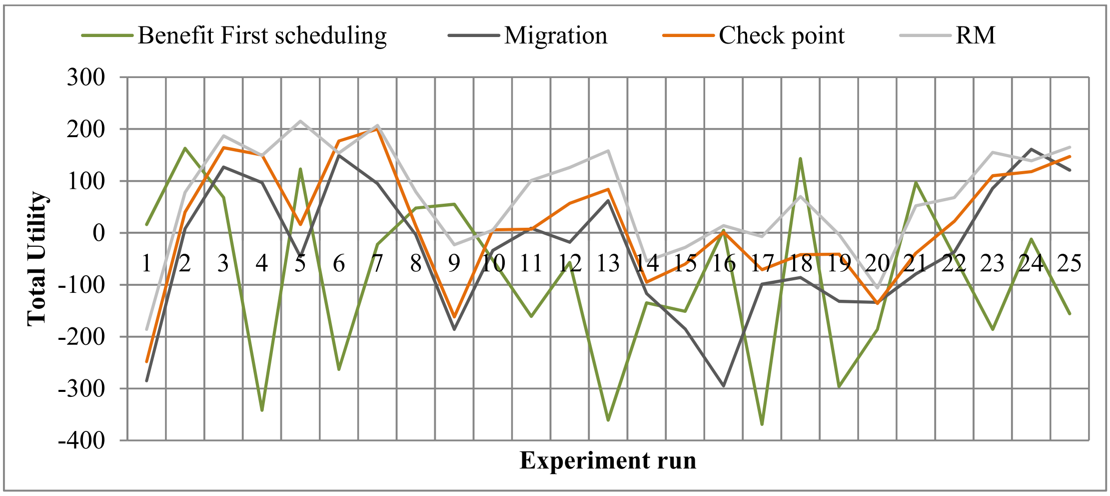

Fig. 3 Total Utility in single resource system

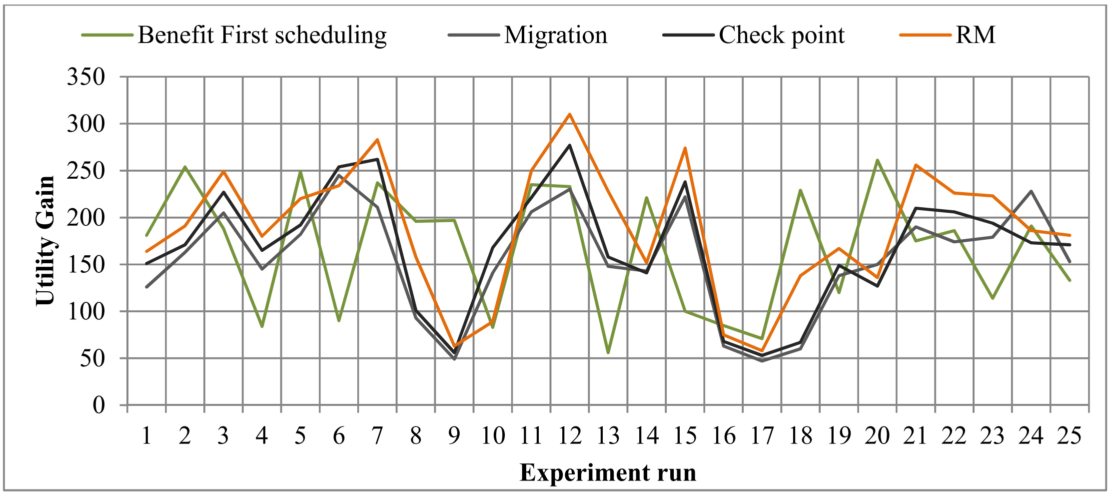

Fig. 4 Utility Gain in multi resource system

The benefit first search method has lower utility gain of 6.8983% when compared with migration method. RM method increased utility gain by 18.6437% when compared with migration method. Fig. 5 shows the utility loss.

The migration method has higher utility loss of 19.9082% when compared with checkpointing method. RM method has lower utility loss by 41.0478% when compared with migration method. Fig. 6 shows the total utility.

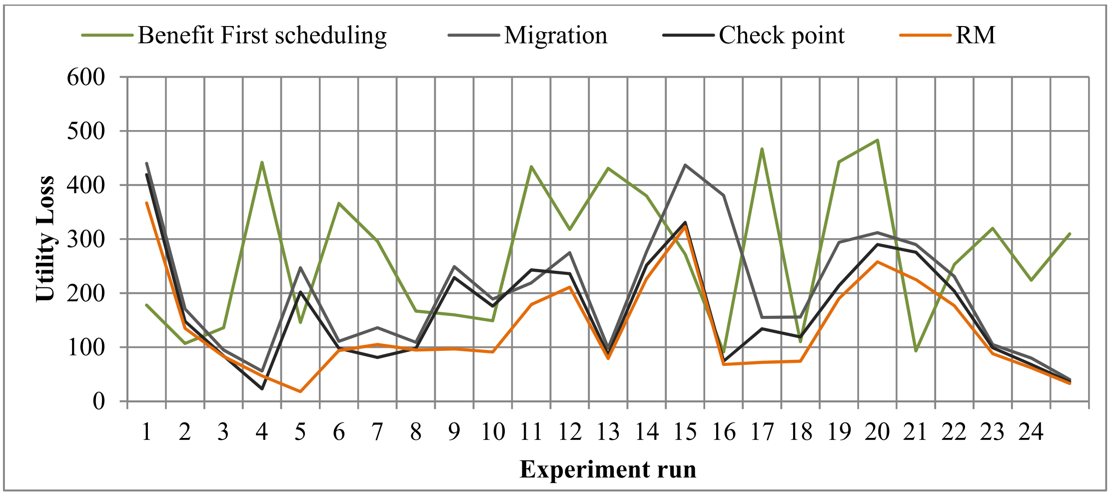

Fig. 5 Utility Loss in multi resource system

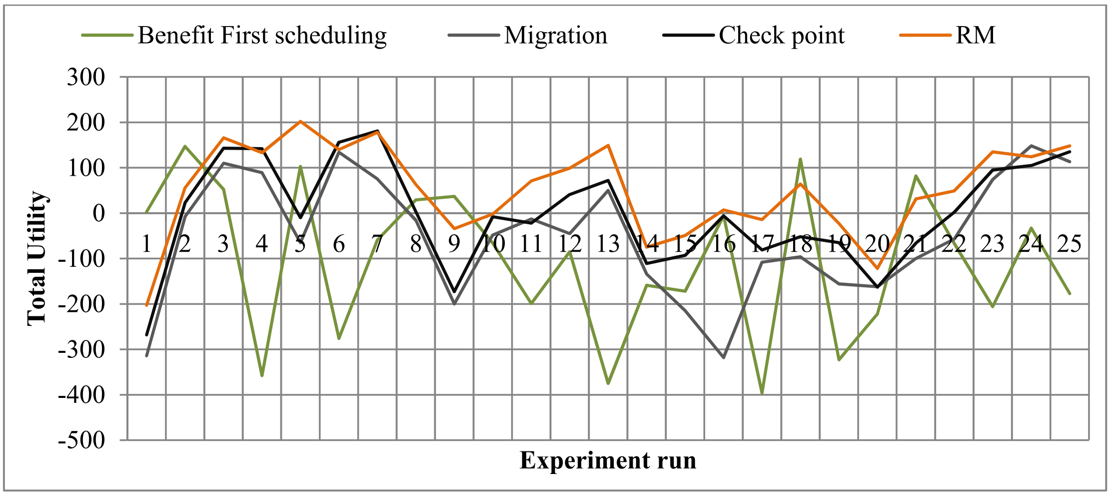

Fig. 6 Total Utility in multi resource system

V. Conclusion

Cloud computing is a paradigm, having potential to reduce costs through optimization and increase operating/economic efficiencies. Cloud computing tasks are different from each other needing optimal execution time. Cloud scheduler assigns multiple users tasks to multiple virtual machines. A good scheduling algorithm assigns virtual machines optimally. In this work, four popular accrual based scheduling mechanisms were studied for multi resource cloud system. Experiments were conducted for four scheduling methods with single processor and with four VM. The utility gain, utility loss, and total utility are calculated, and the results compared with one another algorithm. From the results, RM method outperformed other methods.

References

[1] Salot, P. (2013). A survey of various scheduling algorithm in cloud computing environment. International Journal of research and engineering Technology (IJRET), ISSN, 2319-1163.

[2] Nikshape, S. S., & Deepak, S. S. Delivering Of Real Time Services through Internet Protocol Using Cloud Computing.

[3] Jeffay, K., Stanat, D. F., & Martel, C. U. (1991, December). On non-preemptive scheduling of period and sporadic tasks. In Real-Time Systems Symposium, 1991. Proceedings, Twelfth (pp. 129-139). IEEE.

[4] Jejurikar, R. (2005, July). Energy aware non-preemptive scheduling for hard real-time systems. In Real-Time Systems, 2005.(ECRTS 2005). Proceedings. 17th Euromicro Conference on (pp. 21-30). IEEE.

[5] http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/cpuScheduling.htm

[6] Bouguerra, M. S., Kondo, D., & Trystram, D. (2011, May). On the scheduling of checkpoints in desktop grids. In Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on (pp. 305-313). IEEE.

[7] Nguyen, D., & Thoai, N. (2012, May). Ebc: Application-level migration on multi-site cloud. In Systems and Informatics (ICSAI), 2012 International Conference on (pp. 876-880). IEEE.

[8] Santhosh, R., & Ravichandran, T. (2013, February). Pre-emptive scheduling of on-line real time services with task migration for cloud

computing. In Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on (pp. 271-276). IEEE.

[9] Huang, Q., Su, S., Xu, S., Li, J., Xu, P., & Shuang, K. (2013, July). Migration-based elastic consolidation scheduling in cloud data center. In Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on (pp. 93-97). IEEE.
Ghribi, C., Hadji, M., & Zeghlache, D. (2013, May). Energy efficient VM scheduling for cloud data centers: exact allocation and migration algorithms. InCluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on (pp. 671-678). IEEE.

[10] Melo, M., Araujo, J., Matos, R., Menezes, J., & Maciel, P. (2013, October). Comparative Analysis of Migration-Based Rejuvenation Schedules on Cloud Availability. In Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on (pp. 4110-4115). IEEE.

[11] Shamsinezhad, E., Shahbahrami, A., Hedayati, A., Zadeh, A. K., & Banirostam, H. (2013). Presentation Methods for Task Migration in Cloud Computing by Combination of Yu Router and Post-Copy.

[12] Santhosh, R., & Ravichandran, T. (2014). Task Scheduling for Online Real Time Services with Multiple Resources Using RM Algorithm for Cloud Computing. Australian Journal of Basic & Applied Sciences.

[13] PM, M. S., & Venkatesh, K. A New Optimal Checkpoint Restart Model.

[14] Sharma, R. (2012, August). Task Migration with EDF-RM Scheduling Algorithms in Distributed System. In Advances in Computing and Communications (ICACC), 2012 International Conference on (pp. 182-185). IEEE.

[15] Singh, D., Singh, J., & Chhabra, A. (2012). Evaluating overheads of integrated multilevel checkpointing algorithms in cloud computing environment". IJ Computer Network and Information Security, 5, 29-38.

R. Santhosh is Research Scholar, Karpagam University. He is currently pursuing his doctorate in India.

T. Ravichandran is Principal, Hindusthan Institute of Technology, India.