Efficient Real-Time Tasks Scheduling in Virtualized Cloud Environment

D. Nithya[1], T. Karthik[2], K. S. Krishnapriya[3], K. Menakashree[4]

[1]Assistant Professor, Computer Science and Engineering, VCET, Tamilnadu, India. Email: d.nithyacse1990@gmail.com

[2]Student,Department Of Computer Science and Engineering, VCET, Tamilnadu, India. Email: karthikmoorthy719@gmail.com

[3]Student, Department of Computer Science and Engineering, VCET, Tamilnadu, India. Email: krishnapriya14031999@gmail.com

[4]Student, Department of Computer Science and Engineering, VCET, Tamilnadu, India. Email: menakashree.k@gmail.com

Abstract - The architecture integrated soft real-time task scheduling algorithms, namely master node and virtual machine node schedules. In addition Adaptive Job Scoring algorithm is also applied. When the technology is in advance we must have more computing power to handle the complicated problems. In contrast to tradition, grid computing is proposed instead of using supercomputers. Distributed computing supports resource sharing. Parallel computing supports computing power. The power of both distributed computing and parallel computing is achieved through grid computing .To aggregate idle resources on the Internet such as Central Processing Unit (CPU) cycles and storage spaces to facilitate utilization is the main goal of grid computing. In this project, the energy efficiency of virtual machines in cloud can be calculated using the computation power and transmission power. Also the task splitting strategy is used in order to split the large task into multiple sub tasks. Task replication is used along with this to schedule the task to virtual machines in cloud efficiently. Python 3.6 is used as the front end language to develop the application.

Keywords - Real-time, Cloud computing, Virtual Machine, Deadline, Laxity.

I. INTRODUCTION

With the exponential growth of big data, there has been an increase in the demand for cloud computing over the Past few years. Cloud computing is a type of distributed parallel computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements between the service provider and consumers. Cloud computing facilitates _flexible and dynamic outsourcing of applications while improving cost-effectiveness. Cloud computing offers three different types of service models: Software-as-a-Service (SaaS), Platform as-a-Service (PaaS) and Infrastructure- as-a-Service (IaaS). In SaaS model, the consumer is offered to use provider's applications hosted in a cloud infrastructure. In PaaS, the consumer is provided with required software and hardware tools to develop cloud applications that are hosted in the provider’s cloud infrastructure.

In IaaS, the consumer is provisioned the use of virtualized computing, storage and network resources that are delivered on demand basis. In IaaS model, the consumer will not have control on the cloud infrastructure, however, he can control the operation system, the storage and deployed applications beside the possibility of controlling limited network components such as firewalls .Cloud computing is enabling the development of the next generation of computing services, which would be heavily geared toward massively distributed on-line computing .It also affords a new model of globally accessible on-demand high-performance computing (HPC) services. However, such benefits are presently not available to real-time safety-critical applications. This could be due to the inability of current cloud infrastructures to support timing constraints. Compared with the case of existing real-time scheduling platforms such as multiprocessors and multicores, the handling of real-time constraints on cloud platforms is more complex due to the difficulty of predicting system performance in virtualized and heterogeneous environments .However, real-time applications are gradually progressing unto cloud computing platforms, driven by the tremendous possibilities afforded by such platforms. Examples of hard and soft real-time system applications of cloud computing are military distributed control systems for remote surveillance, early warning and response systems, sensor-driven unmanned vehicles with augmented intelligence, and cloud gaming. Cloud computing technology is not actually geared toward hard real-time applications in closed environments, but soft real-time applications that do not require direct exposure to the hardware bypassing system software. Incidentally, soft real-time scheduling policies can be integrated in virtualization platforms, enabling the system to deliver hierarchical real-time performance. In this paper, we propose and analyse the possibility of applying real-time task scheduling, or deadline- constrained task scheduling, on IaaS cloud computing service model, for the support of soft and near-hard real-time applications. Examples of such applications are cloud-based gaming, online video streaming, and telecommunication management. Such applications could benefit significantly from cloud computing despite the limitations associated with the start-up time and communication of virtual machines. This is because of the ability of cloud computing to support dynamic workloads, thereby enabling the elastic allocation of resources.

II. SCOPE ANDOBJECTIVES

- To avoid the missing deadlines
- To utilize the power of grid completely
- To fit the task with required operating system
- To decrease the completion time of submitted job.

III. EXISTINGSYSTEM

In existing system, a deadline look-ahead module was incorporated into each of the algorithms to fire deadline exceptions and avoid the missing deadlines and to maintain the system criticality. Adaptive Scoring Job Scheduling algorithm (ASJS) for Job execution is being carried out. Cluster Score calculation includes storage capacity requirement. During task assignment to VMs, to fit the tasks with required operating system in the VM with corresponding operating systems (either windows or Linux) is also carried out. Also, time factor is considered for task execution success or failure.

IV. DRAWBACKS OF EXISTING SYSTEM

- Each task is considered as separate unit.
- A single task is given to a selected high score virtual machine cluster only.
- Additional Virtual machine creation is required if the current cloud node is not capable of executing task.
- Splitting of tasks into various Virtual machines is not considered.
- Energy efficiency and task replication features are not considered.

V. PROPOSED SYSTEM

Along with existing system implementation, energy efficiency of virtual machine in cloud is calculated based on the computation power and transmission power. In addition, jobs can be divided into sub tasks and given to one or more clusters. This will be more useful to avoid task failure scenario when larger task is requested. The task replication strategy is used herein order to complete task more efficiently. Also, time factor is considered for task execution success or failure.

VI. ADVANTAGES OFPROPOSED SYSTEM

- Each job is considered as sub tasks.
- A single job is given to a selected multiple clusters since jobs are split into tasks.
- Cluster score values are recalculated even during the job is partially completed. This is achieved when a particular sub task is finished.
- Job Split method avoids task failure scenario and task replication helps in faster job completion.

VII. MODULES

A. Addcluster

In this module, the cluster id is given with ATP (Average Transmission Power) and ACP (Average Computing Power) and CS (Cluster Score) value set to zero. The details are saved in ‘Cluster’ table.

B. Addresource

In this module, the resource id, Resource Name, IP Address, CPU MHz, CPU MHz available, Load Percent, CP (available computing power) and Storage Capacity details are keyed in. The details are saved in ‘Resources’ table.

C. Assign Cluster to Resource

In this module, the cluster id is fetched from ‘Clusters’ table and resource id is fetched from ‘Resource’ table. The ids are selected from combo boxes and are saved in ‘Clusters_Resources’ table.

D. Addjob

In this module, the job id, name, required RAM in MHz, required hard disk storage in MB, CPU MHz and network bandwidth is keyed in and saved into ‘Jobs’table.

E. Cluster Score Calculation For Data Intensive And Computation Intensive Strategy

In this module, the cluster score is calculated based on the formula: CSi = a.ATPi + b.ACPi

where cluster score is denoted as CSifor cluster i, a and b are the weight value of ATPi and ACPi respectively, the sum of a and b is 1, the average transmission power and average computing power are ATPi and ACPi of cluster i respectively. ATPi means the average available bandwidth in the cluster i that can supply to the job and is defined as: wherei, j is the available bandwidth between cluster 'i' and cluster 'j', m is the number of clusters in the grid system entirely. Similarly, ACPi means the average available CPU power of cluster i that can supply to the job and is defined as: where CPU_speed k is the CPU speed of resource k in cluster i, load k is the current load of the resource k in cluster i, n denotes number of resources in cluster i. Also let the available computing power of resource k is denoted as CPk. The performance of job execution is affected by both transmission power and computing power. These two factors are used for job scheduling. Since in the same cluster the bandwidth between resources is normally very large, we only consider the bandwidth between different clusters. global update and local update are used to adjust the cluster score. After a job submission to a resource, the status of the resource will vary and local update will be applied to adjust the score of the cluster containing the resource. After the completion of job by a resource, global update will get in all resources information in the grid system entirely and recalculate the ATP, ACP and CS of all clusters.

F. Cluster Score Calculation with Storage Capacity

In addition with existing formula, the storage capacity is also calculated like Average Transmission Power and so sum of α β and ϒ. All other calculations are used in same scenario as above module.

G. Task Split and Task Replication Strategy

The job is split into tasks with a, b and g values for each sub task. So one cluster is assigned for one single task and others cluster are used for other tasks. Likewise jobs are considered as replica units and so more clusters are assigned for each job. Through this task failure scenario is avoided completely.

VIII. **IMPLEMENTATION OF PROJECT**

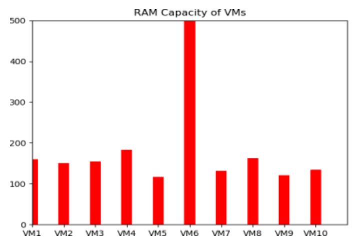

Fig. 1 Task Details

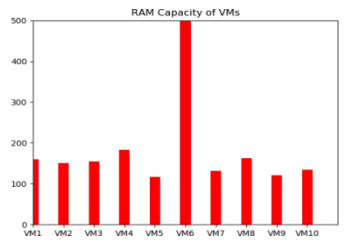

Fig. 2 RAM Capacity

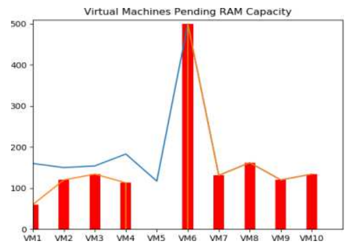

Fig. 3 VM Pending RAM Capacity

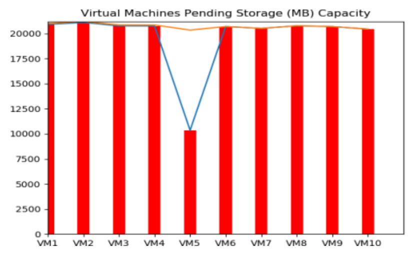

Fig. 4 VM Pending Storage Capacity

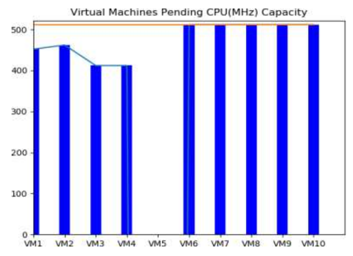

Fig. 5 VM Pending CPU Capacity

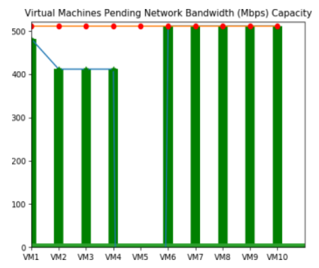

Fig. 6 VM Pending Bandwidth Capacity

IX. **CONCLUSION**

The Mater Node and Virtual machines attempt to assign the tasks to an empty processor. In the absence of such a processor, a new VM node is dynamically created and assigned the task. The results of the demonstrative implementation of the proposed architecture and algorithms in the present study were expressed in terms of the number of deadline exceptions fired by each algorithm, the number of extra resources provisioned to each algorithm to handle the deadline exceptions, and the average response time of the tasks. This project proposes an adaptive job scoring scheduling method to schedule jobs in grid environment. ASJS selects the fittest resource to execute a job based on the status of resources. Local and global update conditions are applied to get the newest status of each resource. Local update rule updates the status of the resource and cluster which are selected to execute the job after assigning the job and the Job Scheduler uses the newest information to assign the next job.

REFERENCES

[1] https://www.sciencedirect.com/science/article/pii/S0020025512002836

[2] https://ieeexplore.ieee.org/document/8645642

[3] https://www.researchgate.net/publication/331238073_Investigating_the_Schedulability_of_Periodic_Real-Time_Tasks_in_Virtualized_Cloud_Environment

[4] Burns and A. J. Wellings, Real-Time Systems and Programming Languages, 4th ed. Toronto, ON, Canada: Pearson Education, 2009.

[5] G. C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, vol. 24, 3rd ed. New York, NY, USA:Springer, 2013.

[6] J. W. S. Liu, Real-Time Systems, 1st ed. Upper Saddle River, NJ, USA:Prentice-Hall, 2000.

[7] R. Mall, Real-Time Systems: Theory and Practice. London, U.K.: Pearson,2009.

[8] H. Kopetz, Real-Time Systems, 2nd ed. New York, NY, USA: Springer,2013.

[9] J. A. Stankovic and K. Ramamritham, ``What is predictability for real-time systems?'' Real-Time Syst., vol. 2, no. 4, pp.247_254, 1990.

[10] I. Lee, J. Y. Leung, and S. H. Son, Handbook of Real-time and Embedded Systems. Boca Raton, FL, USA: CRC Press,2007.

[11] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, ``Multiprocessor scheduling by reduction to uniprocessor: An original optimal approach,''Real-Time Syst., vol. 49, no. 4, pp. 436_474, 2013.

[12] G. Nelissen, V. Berten, V. Nélis, J. Goossens, and D. Milojevic, ``U-EDF:An unfair but optimal multiprocessor scheduling algorithm for sporadic tasks,'' in Proc. 24th Euromicro Conf. Real-Time Syst. (ECRTS), Pisa, Italy,Jul. 2012, pp. 13_23.

[13] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt, ``DP-FAIR:A simple model for understanding optimal multiprocessor scheduling,''inProc. 22nd Euromicro Conf. Real-Time Syst. (ECRTS), Jul. 2010,pp. 3_13.