Real Time Application Scheduling on A Private Cloud Environment

Hsu Mon Kyi, Thinn Thu Naing
University of Computer Studies, Yangon
hsumonkyi.ucsy@gmail.com, ucsy21@most.gov.mm

Abstract

Cloud computing is a scalable distributed computing environment in which a large set of virtualized computing resources, different infrastructures, various development platforms and useful software are delivered as a service to customers as a pay-as-you-go manner usually over the Internet. In cloud computing, virtual machines (VMs) are used as a computing resource. Parallel jobs of user request in cloud need to allocate these resources. Therefore, parallel jobs require a mechanism to scheduling the executions order as well as resource allocation. In this paper, the proposed algorithm schedules two phases. First phase is priority based job scheduling and second is resource allocation phase. The proposed algorithm aims to real time jobs to meet their deadline and effective and efficient resource allocation. As a cloud Testbed, this system implements an Eucalyptus cloud infrastructure along with Xen virtualization technology as VM Monitor backend.

Keywords*: Cloud Computing; Virtual Machine; Scheduling; Virtualization*

1. Introduction

Clouds aim to resource management and scheduling to user applications for the high performance computing community. So, Virtual machine technology is adopted for high end computing to achieve efficient computing resource usage in cloud clusters. Some technical work has reported that virtual machine could be used for scientific applications with tolerable performance punishment [4]. A virtual machine is a software based machine emulation technique to provide a desirable, on demand computing environments for users. As documented in [7], [10], virtual machine provisioning is a popular part of cluster deployments.

In such an environment, it is necessary to schedule the jobs to the appropriate resources that exactly match the requirements of the job. And also it is essential to decide the job that should be given a higher priority so that it can be scheduled first. This priority is given based on deadline. The deadline factor is used to determine the emergent jobs that have to be scheduled first.

To allocate this priority based jobs in virtual machine resource, an effective scheduling strategy is important to meet the desired quality of service parameters from both user and system perspectives. Resource allocation in virtual machines contain more properties which are used for scheduling, for example, memory size, software packages, and access to specific devices.

In this paper, the proposed algorithm schedules both the resources and jobs. Scheduling of jobs to the resources with the parameters like processing time and deadline is the job scheduling. Selection of resources for allocating the jobs fitting the requirements of an application with the parameters like, CPU, memory and storage and software requirements is resource scheduling. The proposed scheduling algorithm is designed and implemented in a private cloud environment. The cloud cluster provides multiple virtual machine templates for incoming jobs.

The two major contributions of this system are to:

- Propose priority based real time scheduling algorithm for parallel jobs, which intend for scheduling real time jobs to meet their deadline.
- Develop a resource allocation algorithm in private cloud system, which considers additional resource properties (CPU, memory size, software requirements) in order to effective and efficient allocation.

The paper is organized as follows: Section 2 discusses related work to this topic. This paper formally defines the research of scheduling virtual machines on cloud computing environment: the target system model and job model in section 3, followed by scheduling algorithms: dynamic priority based real time scheduling and resource allocation are presented in section 4. Section 5 shows implementation of Eucalyptus private cloud environment with scheduling algorithms. Finally, Section 6 concludes the paper.

2. Related Work

Systems for management of VMs across a cluster of physical machines (PMs) have been proposed in the past. Eucalyptus [1] and Usher [6] are open source systems, which include support for managing VM creation and allocation across a PM cluster. However, they do not have VM scheduling policies to dynamically consolidate or redistribute VMs.VM scheduling policies for this purpose have also been investigated in the past. In [8], the authors proposed a dynamic and adaptive real-time virtual machine scheduling technique for HPC workloads on the Grid. The primary objective is to increase overall job throughput in the system. The authors in [2] presented vGreen design to manage VM scheduling across different PMs with the objective of managing the over performance and system level energy savings.

The authors of [7] presented a dynamic approach to create virtual clusters to deal with the conflict between parallel and serial jobs. The authors in [5] presented a Multi-Dimensional Scheduling Algorithm (MDSA) for task scheduling in virtual machine based SOA environments. Each virtual machine is pre-installed with some application level software packages. This algorithm is static based scheduling algorithm.

Similarity, the authors in [3] proposed a preemptable scheduling mechanism. This scheduling considers both resources and jobs. These jobs base on execution order of tasks. In [9], the authors presented deadline aware virtual machine scheduling approach to optimize job deadlines when run in virtual machines.

In our system, we propose a deadline first algorithm for parallel jobs and resource allocation algorithm to produce efficient schedule for the jobs in the job queue. Therefore, the proposed algorithms produce efficient result for scheduling.

3. Scheduling Virtual Machine on a Cloud Computing Environment: System Model

A Cloud is a type of parallel and distributed 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 (SLA) established through negotiation between the service providers and consumers.

Some of the classical cloud-based applications include Social Networking, Web Hosting, Content Delivery, and Real-Time Instrumented data processing. It is very difficult to quantify the performance of scheduling and allocation policy on cloud infrastructures for different applications under varying workload and system size. In our proposed system contains a number of computing serves (or host resources), each of which supplies several virtual machines. Parallel tasks for different user applications will be allocated to some virtual machine by using proposed scheduling algorithm. This algorithm assigns priorities to jobs and then scheduled based on the priorities.

3.1.Target System Model

The target system, where resources are managed in a cloud environment, can be described as: $G = \{NC,VM,N\}$ is the set of resources on cloud infrastructure, where:

- Node controller: $NC = \{nc_i\}, 1 \le i \le H$

NC is the set of H node controllers in the target cloud system. H is the total amount of the node controllers. Each node controller nc_i has 2 properties:
-nc_i. *CPUPerformance* is the value of CPU bandwidth of the node controller,
-nc_i. *MemorySize* is the value of memory size of the node controller.

- Virtual machines: $VM = \{vm_i\}, 1 \le i \le M$

VM is the set of M virtual machines. M is maximum available VMs in 1 NC. Each virtual machine vm_i has 1property:
-vm_i.*Software* is the software installed on the virtual machine, this can be a set of software.

- N is a $H \times M$ matrix $\sum_{i=1}^{H}\sum_{k=1}^{M}$. N_{ik} represents maximum number of VMs in whole system, $1 \le i \le H$ *and* $1 \le k \le M$.

$$N_{ik} = \begin{cases} 0 \text{ if } nc_i \text{ has not } vm_k \\ 1 \text{ if } nc_i \text{ has the } vm_k \end{cases}$$

3.2. Job Model

Parallel real time jobs of the system as $T = \{J\}$, where:

- $J=\{ j_i\}, 1 \le i \le n$

J is the set of n jobs to be scheduled. n is the total number of jobs in J. Each job j_i has 5 properties:
-j_i.*Req_Software*: software required, this can be a set of software,
- j_i.*Req_CPU*: CPU bandwidth required,
- j_i.*Req__Mem:* memory size required, and
- j_i.*Req_Time*: execution time required.
- j_i.*Req_Deadline*: deadline required.

The parallel jobs of users constraints are expressed as follows:

$J \le N_{ik}$ $1 \le i \le H$ *and* $1 \le k \le M$.

4. Scheduling Algorithms

The primary goal of is to schedule a job as early as possible when all the resources required by the job are available. The papers contain two steps:

- Priority based real time scheduling
  Initially all the jobs of the parallel real time job should be scheduled according to their deadline between jobs.

- Resource allocation
  Then the sorted jobs are scheduled onto the target system. The proper host resources and virtual machines will be allocated to each job.

4.1. Priority Based Real Time Scheduling

The parallel jobs $T=\{J\}$to be scheduled based on execution time and deadline and fed into multiple queues. Average of the deadline of the jobs at that instant is taken as a threshold. Priority is given in the order as in the table [1]

Table.1 Priority Order

Deadline	Priority Queue
<=	1
>	2

Job scheduling is repeated for every set of jobs taken at regular intervals. The newly arrived jobs are placed at the appropriate queue and the jobs in the interval of changed threshold are alone rescheduled. In this way, the jobs are prioritized dynamically at regular intervals which prevent the starvation as the priority of the

job varies with the varying threshold at regular intervals. Within the queue, the jobs are put in the descending order of ratio is the execution time divided by deadline. The algorithm for allocation of priority is shown in below.

Algorithm1: Priority Based Real Time Scheduling

BEGIN
Assumes that Get Execution time and Deadline for the job from the system
Compute the threshold value for these inputs
FOR each job
IF job_deadline <= threshold_deadline
THEN insert that job into queue1
ELSEIF job_deadline>threshold_deadline
THEN insert the job into queue2
ENDIF
ENDIF
Compute the execution time/deadline ratio for each job
THEN arrange each queue with decreasing order of ratio and put to priority queue list
END
END

4.2. Resource Allocation Algorithm

In algorithm 2, priority queue list is the sorted job list which is input of algorithm 2. NC is the allocated host for all jobs. VM is the allocated virtual machine for a job. J is the parallel job in priority queue. For a certain task selected from priority queue list, the algorithm searches all the hosts for the suitable a virtual machine resource. In this algorithm, the system defines the suitable resource *nc* and *vm* for task j as follows:

$$j.Req\_CPU \le nc.CPUPerformance$$
$$j.Req\_Mem \le nc.MemorySize$$
$$j.Req\_Software \subseteq vm.Software$$

Moreover, the defined parameter, RLS_t – Resource Load Status, to represent the load status of the target system at certain time, t.

$$RLS(t) = \frac{\sum_j (T_j.\mathrm{Re}q\_CPU \times T_j \mathrm{Re}q\_Mem)}{\sum_j (nc_j.CPUPerform\ ance \times nc_j.MemorySize)} \quad (1)$$

Where:
$nc_i.CPUPerformance$ is the value of CPU bandwidth of the node controller nc_i,
$nc_i.MemoerySize$ is the value of memory size of the node controller nc_i,
T_j is the task which occupies one node controller resource at time t,
T_j *.Req_CPU* is the value of required CPU of task T_j , and
T_j *.Req_Mem* is the value of required memory of task T_j .
RLS(t) shows, at certain time t, how much resources are occupied. In other words, it reflects, how busy the system is in certain time transaction.

5. Implementation of Eucalyptus Private Cloud Environment with Scheduling Algorithms

5.1 System Overview

This paper considers a private cloud system. This cloud system builds on Eucalyptus cloud Testbed. The architecture of Eucalyptus, which is the main component of Ubuntu Enterprise Cloud, has been designed as modular set of 5 simple elements that can be easily scaled: Cloud Controller (CLC), Walrus Storage Controller (WS3), Elastic Block Storage Controller (EBS), Cluster Controller (CC) and Node Controller (NC). **CLC** is the entry-point into the cloud for users and administrators. It queries information about resources, makes high level scheduling decisions, and implements. **WS3** is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data. **EBS** provides functionality similar to the Amazon Elastic Block Storage. It is a block device that can be attached to an instance file system. **CC** gathers information about and schedules VM execution on specific node controllers, as well as manages virtual instance network. **NC** controls the execution, inspection, and terminating of VM instances on the host where it runs.

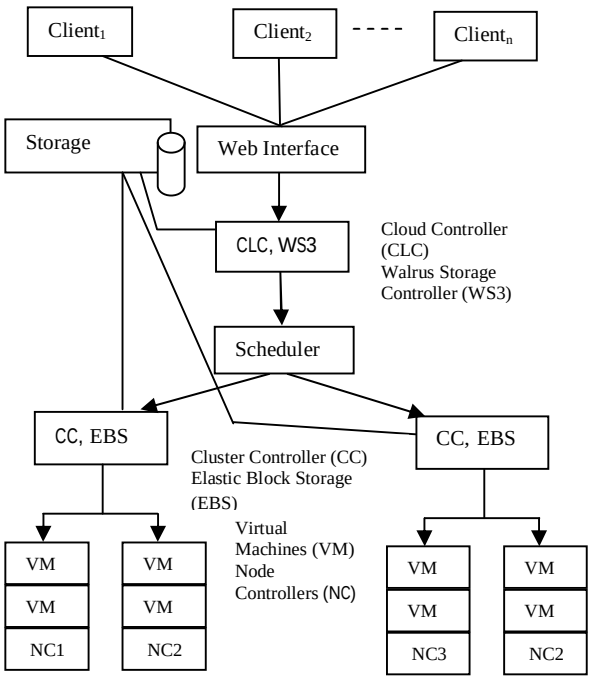

Figure 1: System Architecture

Eucalyptus existing virtualization technology is KVM. In this system, Xen virtualization is used instead of default package KVM. Xen hypervisor allows several guest operation systems to be executed on the same computer hardware concurrently. Xen partitions a single physical machine into multiple virtual machines, to provide server consolidation and utility computing. The private cloud system architecture is shown in Figure 1.

To test Eucalyptus, a simplified two-node cluster was used. CLC, WS3, EBS and CC are used as front-end node. One or more nodes are used as backend node. A front-end node with two network interfaces was connected to both the campus network and a private test network, and a back-end node was connected only to the private network. In this system, cloud providers deliver basic on demand storage and computing capacities over the internet. The provision of these computational resources is in the form of virtual machines (VMs) deployed in a provider's data center. A virtual machine is an abstract unit of storage and computing capacities provided in a cloud. This system assumes that VMs are offered in different types, each of which has different characteristics.

When a user submits a parallel task from the web interface to the cloud controller, it queries resource information to accept the user request. Then, it communicates the scheduler for make scheduling decision. Scheduler gathers the information of VMs on the specific node controllers from the cluster controllers. Then, scheduler makes the scheduling decision to the cluster controller. In this way, each job allocates the specific VMs on the node controller.

5.2 Resource Scheduling Process

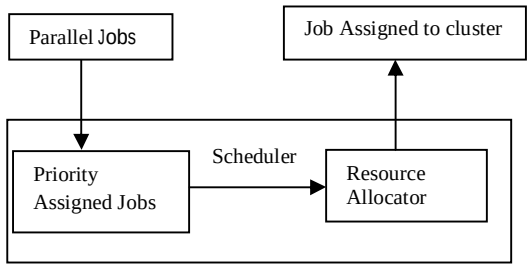

Figure 2: Resource Scheduling Process

Figure 2 shows process of resource scheduling. When parallel jobs are submitted to a cloud, each job assigns priority based on deadline. The resource allocator checks whether its resource availabilities can meet the requirement of this job. Then for a job, the resource allocator decides a node used to execute this job. If the resource allocator does not assign a job to relevant node controller of VM, it will store the job in a queue. When the resources and the data are ready, this task's execution begins.

6. Conclusion

It has been widely accepted that virtual machines can be employed as computing resources for high performance computing. So, virtual machine scheduling and resource allocation is essential in cloud computing environment for increase resource utilization and efficient allocation in virtual machine. This paper proposes priority based real time scheduling on

private cloud environment. This algorithm may efficiently allocate the resource by meeting the deadline of jobs.

Algorithm2 Resource Allocation Algorithm

Priority_Queue_List P: the list of sorted jobs
$NC=\{nc_1,nc_2,\ldots,nc_H\}$
$VM=\{ vm_1,vm_2,\ldots,vm_M\}$
$J=\{ j_1,j_2,\ldots,j_n \}$
j_i.properties={ j_i.Req_Software, j_i.Req_CPU, j_i.Req_Mem, j_i.Req_Deadline , j_i.Req_Time }
t:certain time of RLS
BEGIN
FOR i: job $j_i \in$ P **DO**
  FOR j: node $nc_j \in$ NC **DO**
    Send j_i.properties to the nc_j
    Scan the RLS(t) of nc_j
  ENDFOR
IF(find RLS_{min}(nc) && satisfied j_i.properties) **THEN**
    Assign j_i to node nc
    Update RLS of nc
  ENDIF
ENDFOR
END

References

[1] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. "The Eucalyptus open-source cloud-computing system", *In Proceedings of Cloud Computing and Its Applications*,2008.

[2] Gaurav Dhiman, Giacomo Marchetti, Tajana Rosing. "vGreen: A System for Energy Efficient Computing in Virtualized Environments", ACM,*In Proc.ISLPED*,2009.

[3] Jiayin Li, Meikang Qiu, Jianwei Niu, Wenzhong Gao, Ziliang Zong and Xiao Qin "Feedback Dynamic Algorithms for Preemptable Job Scheduling in Cloud Systems" , *IEEE International Conference on Web Intelligence and Intelligent Agent Technology*, pp.561-564,2010.

[4] L. Wang, M. Kunze, and J. Tao, "Performance evaluation of virtual machine based Grid workflow system," Concurrency and Computation: Practice and Experience, vol. 20, no. 15, pp. 1759–1771, 2008.

[5] Lizhe Wangy, Gregor von Laszewski y, Marcel Kunzez and Jie Taoz "Schedule Distributed Virtual Machines in a Service Orientedent Environment",2009.

[6] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. Usher: an extensible framework for managing custers of virtual machines. *In Proc. LISA*,2007.

[7] N. Fallenbeck, H. Picht, M. Smith, and B. Freisleben, "Xen and the art of cluster scheduling," in First International Workshop on Virtualization Technology in Distributed computing, pp. 4–4, 2006,

[8] Omer Khalid, Ivo Maljevic and Richard Anthony. "Dynamic Scheduling of Virtual Machines Running HPC Workloads in Scientific Grids" IEEE,2009.

[9] Omer Khalid, Ivo Maljevic, Richard Anthony, Miltos Petridis, Kevin Parrott and Markus Schulz. "Deadline Aware Virtual Machine Scheduler for Grid and Cloud Computing" , *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pp.85-90,2010.

[10] V. Buge, Y. Kemp, M. Kunze, and G. Quast, "Application of Virtualisation Techniques at a University Grid Center," in e-Science, pp.155,2006.