



Load Balance Based on Scheduling and Virtual Machine

Neeraj

Department of CSE

HCTM CollegKAITHAL,KUK,India

MS.Alankrita Aggarwal

Assistant Professor,CSE Dept

HCTM CollegeKAITHALKUK,,India

Abstract— “CLOUD COMPUTING” is a new kind of computing model, is coming at the industry level. One of the major contribution of cloud computing is to avail all the resources at one place in the form a cluster and to perform the resource allocation based on scheduling. We will define the user request in the form of requirement query. In which manage the load factor and cpu utilization. In this paper we consider about cloud computing, its working, its problem definition, virtual machine requirements, virtualization, objectives and future development of cloud computing.

Keywords— Cloud Computing, virtual machine, migration, Resource scheduling, Cluster.

I. Introduction

Cloud computing, a new kind of computing model, is coming. This word is a new word that appears at the fourth season, 2007. It is an extend of changing with the need, that is to say the manufacturer provide relevant hardware, software and service according to the need that users put forward. With the rapid development of the Internet, user's requirement is realized through the Internet, different from changing with the need. In fact cloud computing is an extend of grid computing, distributed computing, and parallel computing. Its foreground is to provide secure, quick, convenient data storage and net computing service centered by internet. The factors that impel the occurring and development of cloud computing include: the development of grid computing, the appearance of high quality technology in storage and data transportation, and the appearance of Web2.0, especially the development of Virtualization.

The beauty of cloud computing is that another company hosts your application (or the suite of applications, for that matter). This means that they handle the costs of servers, they manage the software updates, and—depending on how you craft your contract—you pay less for the service. It's also convenient for telecommuters and traveling remote workers, who can simply log in and use their applications wherever they are[12].

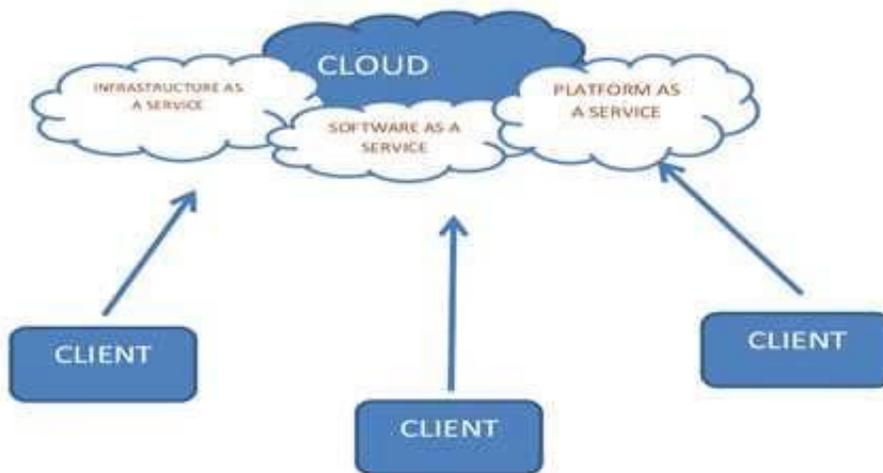


Fig. 1 different services provided by cloud

II. PROBLEM DEFINITION

Cloud computing is one of the emerging research area that is been used effectively at the industry level. One of the major contribution of cloud computing is to avail all the resources at one place in the form a cluster and to perform the resource allocation based on request performed by different users. In this work a cluster based approach is defined to reduce the user efforts to access the cloud resources and to achieve the maximum CPU utilization. The cluster based work gives the batch processing at the cloud process as well as make the effective search to identify the service over the cloud web. The presented work is the hybrid cloud service model with three main features. The first feature is to represent the cloud web in the form of

clusters. The cluster definition are based under different criteria. It can be based on requirement, type or resource etc. Once the clusted cloud is defined. The next work is to perform the scheduling within the cluster. The scheduling is to define the resource allocation criteria. The resource allocation will be performed under the criteria of load balancing and maximum utilization of resource. Number of parallel access to the system will also be controlled by the system. Finally a queue based system is defined to maintain the batch of the related requests to specific clusters or the clouds.

The work has performed an improvement over the cloud resource allocation under the certain factor. In this work, we will define the user request in the form of requirement query and a content based analysis will be performed to generate the batch queue. The scheduling parameter will include the three vectors to perform load balancing, cpu utilization and the content match of requested cloud query. The presented work is the improvement of existing work by performing a user oriented request allocation system under the cloud environment to achieve the better user satisfaction.

III. Essential Requirements[1]

- 1) Service-centric issues
- 2) Quality of Service (QoS)
- 3) Interoperability
- 4) Fault-tolerance
- 5) Load balancing
- 6) Virtualization management

IV. Virtualization and Virtual Machines

It is the technique that removes linking together the hardware and operating system. It directs to the source of the logical resources abstraction away from their physical resources to be more flexible, reduce costs and make a good improvement in business value. Essentially virtualizations in cloud have so many different types, such as, network virtualization, server virtualization and storage virtualization. Server virtualization can be described as an associating of single physical resources to several logical partitions or representations. In a virtualized environment, computing environments can be produced in a forceful dynamic manner, enlarged, become smaller or go in a specified direction or manner as demand varies. Virtualization is therefore highly suitable to a dynamic cloud infrastructure, because it provides important advantages in isolation, manageability and sharing.[8]

Types of Virtualization

A. Hardware virtualization

Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine.

In hardware virtualization, the host machine is the actual machine on which the virtualization takes place, and the guest machine is the virtual machine. The words host and guest are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine. The software or firmware that creates a virtual machine on the host hardware is called a hypervisor or Virtual Machine Manager.

Different types of hardware virtualization include:

- 1) Full virtualization: Almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.
- 2) Partial virtualization: Some but not all of the target environment is simulated. Some guest programs, therefore, may need modifications to run in this virtual environment.
- 3) Para virtualization: A hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.

Hardware-assisted virtualization is a way of improving the efficiency of hardware virtualization. It involves employing specially designed CPUs and hardware components that help improve the performance of a guest environment.

Hardware virtualization is not the same as hardware emulation. In hardware emulation, a piece of hardware imitates another, while in hardware virtualization, a hypervisor (a piece of software) imitates a particular piece of computer hardware or the entire computer. Furthermore, a hypervisor is not the same as an emulator; both are computer programs that imitate hardware, but their domain of use in language differs.

B. Desktop virtualization

Desktop virtualization is the concept of separating the logical desktop from the physical machine. One form of desktop virtualization, virtual desktop infrastructure (VDI), can be thought as a more advanced form of hardware virtualization. Rather than interacting with a host computer directly via a keyboard, mouse, and monitor, the user interacts with the host computer using another desktop computer or a mobile device by means of a network connection, such as a LAN, Wireless

LAN or even the Internet. In addition, the host computer in this scenario becomes a server computer capable of hosting multiple virtual machines at the same time for multiple users.

- C. Software virtualization
- D. Storage virtualization
- E. Memory virtualization

V. Virtual Machines

A **virtual machine** (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Virtual machines are separated into two major categories, based on their use and degree of correspondence to any real machine:

1) A **system virtual machine** provides a complete system platform which supports the execution of a complete operating system (OS) . These usually emulate an existing architecture, and are built with either the purpose of providing a platform to run programs where the real hardware is not available for use (for example, executing software on otherwise obsolete platforms), or of having multiple instances of virtual machines lead to more efficient use of computing resources, both in terms of energy consumption and cost effectiveness (known as hardware virtualization, the key to a cloud computing environment), or both.

2) A **process virtual machine** (also, language virtual machine) is designed to run a single program, which means that it supports a single process. Such virtual machines are usually closely suited to one or more programming languages and built with the purpose of providing program portability and flexibility (amongst other things). An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual environment.

VI. Deadline Aware Virtual Machine Scheduler[6]

A novel approach to optimize job deadlines when run in virtual machines by developing a deadline-aware algorithm that responds to job execution delays in real time, and dynamically optimizes jobs to meet their deadline obligations.

Performance Metrics

To access the performance of our scheduling algorithm, we define the following metrics in our systems:

- 1) System performance to measure total number of jobs completed during a period of time.
- 2) Deadline miss rate representing the number of jobs missing their deadline, thus being terminated by the scheduler.
- 3) Utilization rate for the CPU and memory to measure how long each resource have been active

Questions to be answered

- 1) How dynamic scheduling of workloads at the machine level will work once the batch system have scheduled a job to a particular node given that job will be executed in a virtual machine container?
- 2) What kind of scheduling technique could be used to optimize multiple virtual machines running HPC jobs?
- 3) Which parameter of the job could be considered as pivotal in scheduling policy; deadline-duration ratio or failure rate? Or both of them could be part of the same scheduling technique?
- 4) What would be the mix of executing job types on the machine to increase resource utilization?

Utility Based Job Scheduler[7]

Cloud computing system is featured by its workload, deadline and corresponding utility obtained by its completion before deadline, which also are factors considered in devising an effective scheduling algorithm. Utility is attained from completed jobs. So amount of utility that cloud system obtained is defined in terms of completion of jobs.

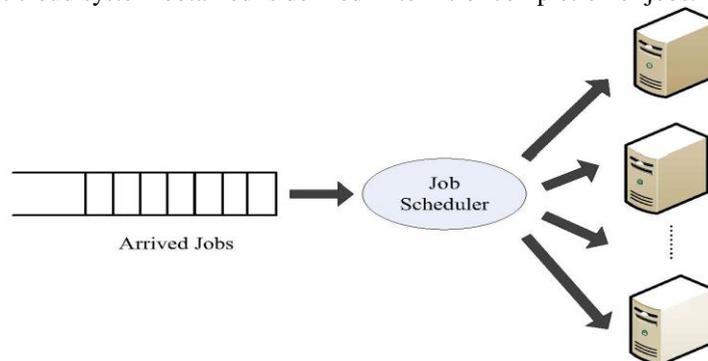


Figure 4 : Job Scheduler[7]

Dynamic Virtual Machine Job Scheduler[2][5]

To serve to diverse user communities with often competing Quality of Service (*QoS*) requirements for their jobs/virtual machines, some jobs being more CPU or memory intensive than the others and vice versa, requires a dynamic and intelligent resource scheduling which is adaptive as the nature of workloads at any given moment changes. *QoS* varies from different utility context such as its different for EC2 user community.

X factor is a ratio of job I that is projected to miss its current deadline and is determined by

$X_i = (\text{job duration remaining-time to deadline}) / \text{job duration remaining}$. It uses three approaches for determining the x value.

- 1) Using a static threshold value
- 2) Using a threshold value which adapts according to the existing conditions
- 3) Using c

New Metrics Of Job Scheduling Of Virtual Machines

In this paper, we propose a new set of metrics, called potential capacity (PC) and equilibrium capacity (EC), of resources that incorporate these dynamic, elastic, and sharing aspects of co-located virtual machines. We then show that we mesh this set of metrics *smoothly into traditional scheduling algorithms*.

VII. Scheduling

Scheduling mechanism is the most important component of a computer system. Scheduling is the strategy by which the system decides which task should be executed at any given time. There is difference between real-time scheduling and multiprogramming timesharing scheduling. It is because of the role of timing constraints in the evaluation of the system performance.

A. Cloud Scheduling

The main reason of using a Cloud system is to improve the performance of large-scale application programs, high scalability and fault tolerance. Therefore, it is essential to keep the cost of creating and managing parallelism as low as possible.

As for scheduling in a Cloud system, there are three goals to lower down the cost:

- a) Good processor utilization: all processors have work in their queues at all times. All processors, which have tasks assigned to them from the same program finish execution at the same time, thus the user gets the expected speedup. Processors spend most of their time doing useful work rather than coordination the division of work.
- b) Good synchronization effectiveness: Tasks are scheduled in such a way that interacting tasks across with fine-grained interaction should be running at the same time.
- c) Low communication/memory-access cost: Tasks are scheduled in such a way that communication time, either message passing or shared-memory latency is accounted for, and minimized. Scheduling data structures should be arranged so that they are no a source of contention.

B. Single Shared Ready Queue

It is a simple approach. A single global queue shared by all the processors in the system. Whenever a processor is available and the queue is not empty, the processor will be scheduled with a new task. The scheduling policies such as First Come First Serve (FCFS) and Shortest Job First (SJF) can be easily implemented. [9] It seems as if this approach may yield good processor utilization. However, this approach does not provide any mean to schedule fine-grained interacting tasks to run at the same time. Consider the following scenario: Task A already scheduled for a processor, but it must wait for completion of Task B, which is still in the queue. Even though Task A keeps the processor busy, all it does is to wait. This approach also has potential high communication/memory access cost, because this global queue occupies a region of memory that can be accessed by all the processors simultaneously.

C. Coscheduling

In coscheduling, all runnable tasks of an application are scheduled on different processors simultaneously. Without coordinated scheduling, the processes constituting parallel tasks may suffer communication latencies because of processor thrashing. In recent years, researchers have developed parallel scheduling algorithms that can be loosely organized into three main classes, according to the degree of coordination between processors: explicit or static coscheduling, local scheduling, and implicit or dynamic coscheduling.

D. Explicit Coscheduling

Explicit coscheduling ensures that the scheduling of communication jobs is coordinated by creating a static global list of the order in which jobs should be scheduled and then requiring a simultaneous context-switch across all processors. This may be accomplished statically by agreeing upon a global schedule in advance or dynamically by having a "master" local scheduler direct other schedulers by communicating with them at each context switch. However, this approach has quite a few drawbacks. Since it requires identifying parallel tasks in the application in advance, it complicates the implementation. Furthermore, it interacts poorly with interactive use and load imbalance.

E. Local Coscheduling

Conversely, local scheduling allows each processor to independently schedule its processes. Although attractive due to its ease of construction, the performance of fine-grain communicating jobs degrades significantly because scheduling is not coordinated across processor.

F. Implicit Coscheduling

An intermediate approach, implicit coscheduling, allows each of the local schedulers to make decisions independently, but relies on local schedulers to take the communication behavior of local processes into account when making decisions. Local schedulers can converge on coscheduling behavior since each sees similar or related communication behavior by local processes that are part of parallel applications. There are many forms of implicit coscheduling developed over the years.

G. D_EDF

Deadline Monotonic is fixed-priority scheduling algorithm whereas EDF is dynamic-priority algorithm. A fixed-priority scheduler assigns the same priority to all instances of the same task, thus the priority of each task is fixed with respect to other tasks. However, a dynamic-priority scheduler may assign different priorities to different instances of the same task, thus the priority of each task may change with respect to other tasks as new task instances arrive and complete

VIII. Objectives

- 1) The main objective of the work is divide the available cloud resources in the clustered form. The clustering criteria will be dynamic as well as parametric. The parameters taken here are the user request analysis and the resource type.
- 2) The second objective of the work is to perform the resource allocation based on the user requirement, load balancing status and the resource utilization.
- 3) The main objective of the work is to perform the reliable and user oriented resource allocation in a clustered cloud environment.
- 4) Implementation of the proposed work in matlab environment.

IX. Research Design

The proposed system is a middle layer architecture to perform the cloud allocation in case of underload and overload conditions. The over load conditions will be handled by using the concepts of process migration. The middle layer will exist between the clouds and the clients. As the request will be performed by the user this request will be accepted by the middle layer and the analysis of the cloud servers is performed by this middle layer. The middle layer is responsible for three main tasks

1. Scheduling the user requests
2. Monitor the cloud servers for its capabilities and to perform the process allocation
3. Process Migration in overload conditions

Algorithm

1. Input the M number of Clouds with L number of Virtual Machines associated with Each cloud.
2. Define the available memory and load for each virtual machine.
3. Assign the priority to each cloud.
4. Input N number of user process request with some parameters specifications like arrival time, process time, required memory etc.
5. Arrange the process requests in order of memory requirement
6. For i=1 to N
7. {
8. Identify the priority Cloud and Associated VM having AvailableMemory > RequiredMemory(i)
9. Perform the initial allocation of process to that particular VM and the Cloud
10. }
11. For i=1 to N
12. {
13. Identify the Free Time slot on priority cloud to perform the allocation. As the free slot identify, record the starttime, process time, turnaround time and the deadline of the process.
14. }
15. For i=1 to N
16. {
17. If finishtime(process(i))> Deadline(Process(i))
18. {
19. Print "Migration Required"
20. Identify the next high priority cloud, that having the free memory and the time slot and perform the migration of the process to that particular cloud and the virtual machine.
21. }
22. }

X. FUTURE DEVELOPMENT

The presented work is about to perform the scheduling and the allocation of the processes by the virtual machine. In this job scheduler to use with research design algorithm. The presented work is defined the overload conditions in terms of deadline

as well as the memory limit of the clouds. In future some other parameters can also be taken to decide the migration condition. The presented work is defined for the public cloud environment, but in future, the work can be extended to private and the hybrid cloud with fast processing.

XI. Conclusion

The presented work is about to define a new clustered architecture to perform the effective and the reliable resource allocation. In this work at the initial stage the dynamic parametric approach is suggested to divide the cloud resources under a parametric clustered approach. The clustering is here defined under the criteria of user request analysis and the user request will be performed to these clustered architecture. The resource allocation will be performed based on the user query content analysis as well as based on the resource allocation vector and the resource utilization. In this work, the main stress is given to the user satisfaction under the two main processes of clustering and the resource allocation.

ACKNOWLEDGEMENT

It is a great pleasure for me to record my deep sense of gratitude and appreciation to Ms Alankrita, the Paper supervisor for his valuable guidance, keen interest, encouragement and friendly discussion during the paper of the present research work. He has guided my endeavors and encouraged me to explore the joys of learning.

References

- [1]. Xun Xu, "From cloud computing to cloud manufacturing", 2011 Elsevier Ltd
- [2]. Omer Khalid, Ivo Maljevic, Richard Anthony, Miltos Petridis, Kevin Parrott, Markus Schulz, "Dynamic Scheduling of Virtual Machines Running HPC Workloads in Scientific Grids" 2009 IEEE.
- [3]. Yanbin Liu, Norman Bobroff, Liana Fong, Seetharami Seelam, Javier Delgado, "New Metrics for Scheduling Jobs on a Cluster of Virtual Machines", 2010
- [4]. Chuliang Weng, Zhigang Wang, Minglu Li, and Xinda Lu, "The Hybrid Scheduling Framework for Virtual Machine Systems", 2009 ACM
- [5]. Jeongseob Ahn, Changdae Kim, Jaeung Han, Young-ri Choi†, and Jaehyuk Huh, "Dynamic Virtual Machine Scheduling in Clouds for Architectural Shared Resources", 2011 IEEE
- [6]. Omer Khalid, Ivo Maljevic, Richard Anthony, Miltos Petridis, Kevin Parrott, Markus Schulz, "Deadline Aware Virtual Machine Scheduler for Grid and Cloud Computing", 2010 IEEE
- [7]. Zheng Hu, Kaijun Wu, Jinsong Huang, "An Utility-Based Job Scheduling Algorithm for Current Computing Cloud Considering Reliability Factor", 2012 IEEE
- [8]. Masaya Yamada, Yuki Watanabe, Saneyasu Yamaguchi, "An Integrated I/O Analyzing System for Virtualized Environment, 2011
- [9]. Loganayagi.B, S.Sujatha, "Enhanced Cloud Security by Combining Virtualization and Policy Monitoring Techniques", 2011 Published by Elsevier Ltd.
- [10]. Xiaoli Wang, Yuping Wang, Hai Zhu, "Energy-Efficient Multi-Job Scheduling Model for Cloud Computing and Its Genetic Algorithm", Hindawi Publishing Corporation Mathematical Problems in Engineering, Volume 2012
- [11]. Ilango Sriram, Ali Khajeh-Hosseini, "Research Agenda in Cloud Technologies", 2010
- [12]. Anthony T. Velte, Toby J. Velte, Robert Elsenpeter, "Cloud Computing, A Practical Approach"
- [13]. Michael Menzel, "CloudGenius: Decision Support for Web Server Cloud Migration", WWW 2012, April 16–20, 2012, Lyon, France. ACM 978-1-4503-1229-5/12/04.
- [14]. Kento Sato, "A Model-Based Algorithm for Optimizing I/O Intensive Applications in Clouds using VM-Based Migration", 9th IEEE/ACM International Symposium on Cluster Computing and the Grid 978-0-7695-3622-4/09© 2009 IEEE
- [15]. Takahiro Hirofuchi, "Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension", 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing 978-0-7695-4039-9/10© 2010 IEEE