Balancing Load of Cloud Data Center using Efficient Task Scheduling Algorithm

Subhadra Bose Shaw
Assistant Professor
AKS University, Satna

ABSTRACT
Cloud computing is one of the most popular terms of today's computer world. The pay-as-you-use model of cloud permits users to pay only according to their requirement. The enormous increase in popularity of cloud is due to its ubiquitous use through common hardware only. So it must provide high performance gain to the user and at the same time must be beneficial for the Cloud Service Provider (CSP). To achieve this goal many challenges have to be faced. Load balancing is one of them. To distribute the load evenly in cloud the resources and workloads must be scheduled efficiently. A variety of scheduling algorithms are used by load balancers to determine which backend server to send a request to. The selected server allocates resources and schedules the job dynamically on some virtual machine (VM) located on the same physical machine. In this paper, we have proposed a task scheduling algorithm which will distribute the task among all the available virtual machines in a way such that none of them become overloaded. Further we have simulated our algorithm in CloudAnalyst and compared it with the existing load balancing algorithms. Results show that the proposed method not only balances the load more efficiently but also improves the response time.

Keywords
Cloud Computing, Load Balancing, Task Scheduling, Virtualization

1. INTRODUCTION
Cloud provides a massively parallel internet-based distributed computing paradigm. Users can use the resources available in a cloud data center in a pay-as-you-go form. According to NIST [1] cloud computing is a model which enables ubiquitous, on-demand and convenient network access to a shared pool of configurable resources e.g., servers, storage, networks, services and applications. Three most important services offered by cloud are SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) [2]. The main technology behind cloud computing is virtualization. Different types of jobs can be simultaneously executed over a common hardware platform by the virtue of virtualization [3]. In spite of this resource sharing the applications of different users run in a totally isolated environment provided by the virtual machines (VM). VM is basically a logical machine that executes over a physical machine (PM) [4]. It prevents the co-located jobs from interacting with each other. Hypervisor or VMM (Virtual Machine Manager) manages the creation, updation and deletion of VMs by allocating required resources. It also provides a duplicate image of the actual hardware to the VMs [5].

Though virtualization tries to evenly distribute the load dynamically [6] there is always a possibility that some of the nodes become over utilized while some other remain under utilized. Overloaded server causes degradation of performance whereas underloaded server causes poor utilization of resources. The overloaded servers produce more heat. As a result the requirement of cooling system is also increased which in turn increases the release of CO_2 [7]. It has been observed that power consumed by cooling systems is greater than the core IT equipment. On the other hand underloaded servers increase the total consumption of power which is very much harmful for the environment. Not only that, it also increases the operational cost of the system. So to fulfill the Quality of Service (QoS) requirements at a reasonable cost, correct amount of resources must be provided dynamically to the tasks running in VMs. Proper task scheduling will also balance the load of the system.

Balancing load is one of the major research challenges in cloud computing. It distributes the dynamic workload equally across all the physical machines in the cloud data center so that none of them become over loaded or under loaded [8]. By optimal consumption of resources it helps to accomplish high throughput, less response time, scalability, improved fault tolerance, high user satisfaction, optimum power consumption, less heat production, less emission of CO_2 and reduced operational cost.

Load balancing or task scheduling algorithms are generally categorized into two types: static scheduling algorithms and dynamic scheduling algorithms. Static scheduling algorithm assigns tasks to virtual machines before the execution of program i.e. during compilation. All the information on which scheduling decision is based like resource requirement of the process, its processing time etc. must be known a priori i.e. at compile time [9]. Static scheduling algorithms are non-preemptive i.e. once a task has been assigned to a processor it can not be pre-empted until it releases the processor and other resources voluntarily. Static scheduling algorithms reduce the overall execution time but they are not adaptable to dynamic load changes [10]. To achieve this dynamic load balancing or dynamic scheduling algorithm is used. It redistributes the processes among processors during execution. This is achieved through VM migration. Though dynamic scheduling balances the load efficiently but VM migration increases the run-time overhead due to the transfer of load information among different processors and decision-making for the selection of processes and processors for job transfers. There is also a communication delay associated with each VM migration. In this paper we have discussed about the existing static load balancing or task scheduling algorithms and also proposed a novel method for balancing load in a cloud data center.

We have organized the paper as follows: Different load balancing algorithms with their pros and cons are described in Section II. Section III gives the details working of the proposed method. Section IV illustrates performance analysis of the proposed method. Finally, the conclusion and some future research directions are presented in Section V.

2. LITERATURE SURVEY OF LOAD BALANCING ALGORITHMS

Task, also known as cloudlet, submitted by user is managed by the data center controller [11]. The controller uses a VmLoadBalancer to decide where the next request should be assigned for processing. The load balancer uses one of the following algorithms to balance the load.

- Round Robin - The requests are assigned by the datacenter controller to a VM which is in the front of a queue. After that the VM will be placed at the end of the queue [12]. It distributes the workload equally among the VMs but the method is not concerned about the execution time of different jobs. As a result at some point of time some of the nodes become over loaded while other remain under utilized [8].
- Weighted Round Robin – It is the updated version of Round Robin in which each VM gets a weight according to its capacity so that if one VM can handle double load than other, then the powerful machine is assigned a weight of 2. The powerful VM will be assigned 2 tasks for each task being allotted to the weaker one. Advanced load balancing requirements such as execution time is not considered for individual requests [12].
- Dynamic Round Robin [13] – The primary concern of this algorithm is to reducing the power consumption of physical machine. This algorithm uses the following two rules to balance the load:

i) If a VM has completed its execution and there are remaining VMs located on the same PM, this PM is said to be in "retiring" state and will accept no more new VM.

ii) From such physical machines all the VMs are migrated to other PM. After successful migration of all VMs, the PM is switched off.

Though this algorithm is capable of reducing power consumption cost it is not scalable and is not suitable for large data centers.

- Randomized – The selected jobs is randomly allocated to any available VM. It is a simple algorithm but does not consider the present load of the VM. Hence, it can select a VM which is already loaded [14].
- Equally Spread Current Execution (ESCE) Algorithm - The job queue and the list of VMs are continuously scanned by the load balancer to find a free VM. Request is assigned to an available VM which is found first and is capable of handling it [15]. The load balancer also distributes some of the tasks from overloaded VM. The balancer improves the response time but it is not fault tolerant [13].
- Throttled - A table is maintained by the TLB (Throttled Load Balancer) to keep track of the state of each VM i.e. whether it is busy or idle [12]. As soon as a request arrives, the table is searched. If a match is found on the basis of size and availability then the request is accepted otherwise -1 is returned and the request is put into a queue. It does not consider the current load on the VM while assigning a request.
- Modified Throttled - The first VM selection is similar to Throttled algorithm. The next request is assigned to the VM which is at index next to previously allocated VM depending on its state and the usual steps are followed [16]. Compare to the above method it produces better response time. But in index table the state of some VM may be updated during the allocation of next request due to completion of some tasks.
- Active Monitoring Load Balancing (AMLB) Algorithm – The load balancer identifies the least loaded VM to assign a request. If multiple least loaded VMs are found then the first one is selected [17]. The processing power of VM is not considered.
- VM-Assign Load Balancing Algorithm – It is an updated version of the above algorithm. First request is assigned in the same way as described above. Then in the next request the least loaded VM which is not allocated in the last round, is assigned [18].
- Min-Min Scheduling Algorithm [19] - The resource having minimum completion time for all tasks is selected. Then the smallest task is found and allocated to the corresponding resource. So this method is known as Min-Min algorithm. This task is removed from set and the same process is repeated for all the remaining tasks. It is straight forward method but it does not take care of the existing load on a resource.
- Max Min Algorithm - It is almost similar to the above algorithm. The only difference is that it gives more priority to the larger tasks [13].
- Double Threshold Energy Aware Load Balancing Algorithm (DT-PALB)[20] – It consists of three basic sections. The first section decides where VMs will be instantiated. It is called the balancing section. Extra compute nodes are switched on during peak period by the second section. The last switch off servers which are idle. This method is efficient for load balancing but initiates unnecessary VM migrations.

3. PROPOSED WORK

From the above discussion it is clear that all the load balancing algorithms have some positive and negative sides. In this paper we have concentrated on the working of Active Monitoring Load Balancing (AMLB) Algorithm and VM-Assign Load Balancing Algorithm. According to the authors of [18] the later algorithm will properly utilize all the VMs unlike AMLB where few VMs will be over utilized with many requests and rest will remain under loaded [18]. But the reason is not clearly mentioned in the paper. This algorithm will not use the VM if it is already assigned in the previous round. But it does not seem to be logical. Because it may become free when the next request arrives. So it remains as the least loaded VM and more tasks can be allocated to it. Finding the next least loaded VM will distribute the tasks equally only when there are more than one VMs which are equally loaded or the next least loaded VM has a high processing speed compare to the previous one. But the algorithm is only concerned about the load and if the VMs are equally loaded then the task can be allocatd to any of them irrespective of the fact that whether the VM is used in the last iteration or not.

To remove this shortcoming we have proposed a method which will find the virtual machine which is least loaded as well as available. The concept of availability will prevent the assignment of the same VM repeatedly. In VM-Assign Load Balancing Algorithm the least loaded VM is not considered in the current round if it is used in the previous round. But it does not prevent its assignment in the next round as well. So

there is a chance that load will be distributed between the first two least loaded VMs. Other VMs will remain under-utilized. In contrast to this our proposed algorithm will consider all the available VMs having little load while task scheduling.

Algorithm: Efficient Load Balancer

1. Input : List of incoming jobs, list of VMs Output : Id of the VM where job will be allocated
2. initialize minCount to MAX_VALUE;
3. initialize vmId to -1
4. for each vm in VmList do
5. find the number of tasks assigned to vm

   if vm is being allocated for the first time

   then

   currCount =0;

   else

   currCount=vm.getCurrentAssignedJobCount;
6. find the current state of vm

   state = vm.getState();
7. if (currCount <= minCount && state.equals(AVAILABLE)){

   minCount = currCount;

   vmId = vm; }
8. end for
9. return vmId

The above algorithm takes list of incoming jobs and list of VMs as input. It outputs the Id of the VM where job will be allocated (line no.1). In line 2 and line 3 minCount and vmId are initialized respectively. A loop is executed (from line no. 4 to 8) to check all the VMs in the vmList and find the most suitable one. currCount will hold the number of tasks assigned to vm (line no. 5). The vm can be either in Available or in Busy state which is stored in the state variable (line no.6). After the execution of the loop vmId will hold of that vm which is currently available as well as least loaded.

4. PERFORMANCE ANALYSIS

To do experiments in a repeatable and dependable fashion we have selected CloudAnalyst tool [11]. Simulation and visual modeling of large-scale applications are supported by this tool. It is built on the top of CloudSim [21]. CloudAnalyst generates information about processing time and response time of requests and cost of requests. By performing various simulations on CloudAnalyst, application developers can find the most efficient way to allocate resources and can optimize the cost of services.

4.1 Experiment Setup

Six main regions of the world are represented using six user bases. For the sake of simplicity we have assumed each user base is contained within a single time zone and most of the users use the application in the evening hours for about 2 hours. It is also assumed that only one tenth of the number of peak users remain on line during the off-peak hours. Furthermore, each user makes a new request every 5 minutes when he or she is online. VMs have 1GB of RAM and have 10MB of available bandwidth. Data center has 50 VMs. Simulated hosts have x86 architecture, Linux operating system and Xen as VMM. Physical machines have 2 GB of primary memory and 100GB of storage. Each machine contains 4 processors. Each CPU possesses a capacity of 10000 MIPS. Resource scheduling is based on time-shared policy. Users and requests are grouped by a factor of 1000 and 100 respectively. 250 instructions are to be executed to fulfill each user's request. The details of the User bases can be obtained from [11].

4.2 Performance metrics

We have compared the performance of our proposed algorithm with the performance of AMLB and Throttled. The three parameters used for comparison are: average response time (in ms), data center processing time (in ms) and distribution of load in each VM.

4.3 Simulation results

Table 1: Comparison of the average response time and data center processing time of different task scheduling algorithms

Algorithm	Average response time (ms)	Data Center processing time (ms)
Proposed	**326.56**	**86.63**
AMLB	333.61	93.66
Throttled	339.36	99.34

Table 1 shows that the average response time as well as the processing time are decreased by the proposed algorithm. Throttled algorithm only considers the state of the VM i.e. whether it is busy or available. It does not consider the current load on the VM. On the contrary, AMLB always finds the least loaded VM irrespective of its current state. So both of them are not capable of evenly distributing the load among all the VMs.

Our proposed method considers both the current state and load on the VM. So it distributes the load almost equally among all the VMs. This can be verified from figure 3 which shows the total number of tasks assigned to each of the 50 VMs by the above mentioned three algorithms. Figure 1 and 2 show the average response time and processing time of these three task scheduling algorithms.

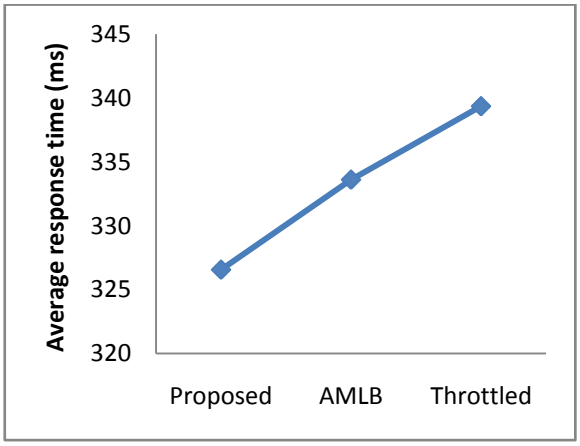

Fig 1: Comparison of average response time of different task scheduling algorithms

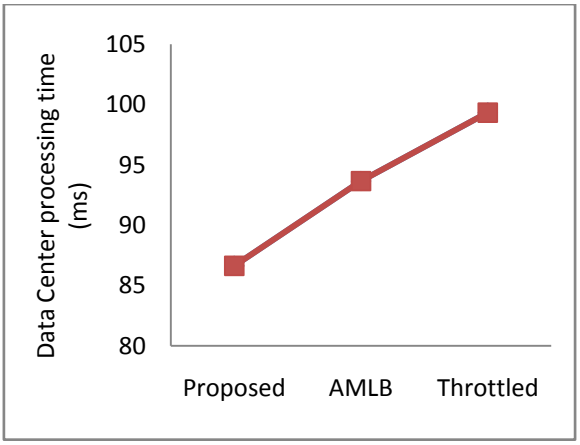

Fig 2: Comparison of the data center processing time of different task scheduling algorithms

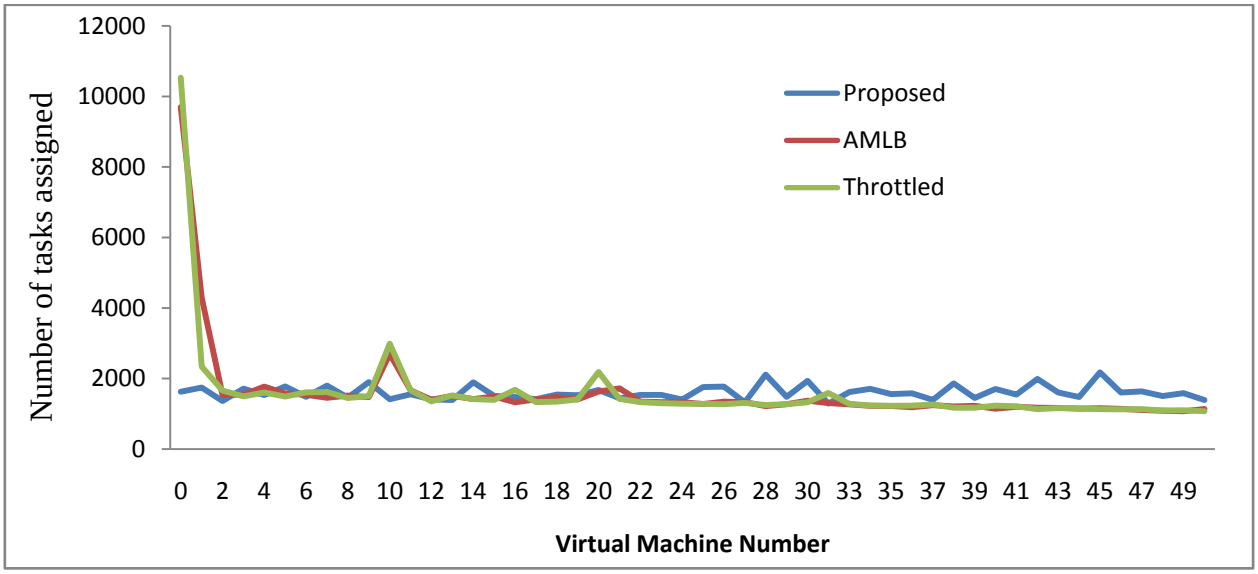

Fig 3: Total number of tasks assigned to each VM by different task scheduling algorithms

5. CONCLUSION AND FUTURE WORK

From the above discussion we can conclude that our proposed algorithm performs better than both the AMLB and Throttled algorithm. It removes the shortcomings of both of these algorithms by combining them together. As a result it is able to balance the load by distributing it equally among all the virtual machines. Even distribution of load on the other hand reduces the response time and processing time of the tasks. This will increase the through put of the system and also reduces the operational cost of the cloud data center.

To perform the experiments in a repeatable manner we have simulated the algorithms using CloudAnalyst tool. The experiments are done on static data. But in future we want to implement the proposed method in real cloud platform using dynamic workload.

6. REFERENCES

[1] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing (Draft). NIST. 2011.

[2] Qi Zhang, Lu Cheng, Raouf Boutaba; Cloud computing: sate-of-art and research challenges; Published online: 20th April 2010, Copyright : The Brazillian Computer Society 2010.

[3] Lazaros Gkatzikis, Iordanis Koutsopoulos, "Migrate or Not? Exploiting Dynamic Task Migration in Mobile Cloud Computing Systems", IEEE Wireless Communications 2013, pp. 24-32.

[4] Raghavendra Achar, P. Santhi Thilagam, Nihal Soans, P. V. Vikyath, Sathvik Rao and Vijeth A. M., "Load Balancing in Cloud Based on Live Migration of Virtual Machines ", 2013 Annual IEEE India Conference (INDICON) .

[5] Akshay Jain, Anagha Yadav, Lohit Krishnan, Jibi Abraham , "A Threshold Band Based Model For Automatic Load Balancing in Cloud Environment ".

[6] Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment", 3rd International Symposium on Parallel Architectures, Algorithms and Programming, IEEE, 2010, pp. 89-96.

[7] Anton Beloglazov, Rajkumar Buyya "Energy Efficient Resource Management In Virtualized Cloud Data Centers," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 826-831.

[8] Amandeep Kaur Sidhu, Supriya Kinger , "Analysis of Load Balancing Techniques in Cloud Computing ", International Journal of Computers & Technology , Volume 4 No. 2, March-April, 2013, pp. 737-741.

[9] X. Evers , "A Literature Study on Scheduling in Distributed Systems ", 1992

[10] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi , "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms ", IEEE Second Symposium on Network Cloud Computing and Applications , 2012, pp. 137-142.

[11] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, Rajkumar Buyya,"CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", 20-23, April 2010, pp. 446-452.

[12] Jasmin James, Dr. Bhupendra Verma, "Efficient VM Load Balancing Algorithm for a Cloud Computing Environment ", International Journal on Computer Science and Engineering (IJCSE) , Vol. 4 No. 09 Sep 2012 , pp. 1658-1663.

[13] M.Aruna , D. Bhanu, R.Punithagowri , "A Survey on Load Balancing Algorithms in Cloud Environment ", International Journal of Computer Applications, Volume 82 – No 16, November 2013 , pp. 39-43.

[14] Isam Azawi Mohialdeen , "Comparative Study of Scheduling Algorithms in Cloud Computing Environment ", Journal of Computer Science , 2013, pp. 252-263.

[15] Tanvee Ahmed, Yogendra Singh "Analytic Study Of Load Balncing Techniques Using Tool Cloud Analyst" , International Journal Of Engineering Research And Applications, 2012, pp. 1027-1030.

[16] Shridhar G. Domanal, G. Ram Mohana Reddy, "Load Balancing in Cloud Computing Using Modified Throttled Algorithm", IEEE, International conference. CCEM 2013.

[17] Hemant S. Mahalle, Parag R. Kaveri, Vinay Chavan, "Load Balancing On Cloud Data Centres", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, issue 1, January 2013.

[18] Shridhar G.Damanal and G. Ram Mahana Reddy , "Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines ", IEEE 2014.

[19] Huankai Chen, Professor Frank Wang, Dr Na Helian, Gbola Akanmu, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing", IEEE, 2013.

[20] Jayant Adhikari, Prof. Sulabha Patil, "Double Threshold Energy Aware Load Balancing in Cloud Computing", 4th ICCCNT, July 2013.

[21] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, Rajkumar Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software – Practice and Experience, pp. 23–50, 2011.