Surabhi Jain, Deepak Sharma / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 2, March -April 2013, pp.1605-1610

Elastic Load Balancing in Cloud Storage

Surabhi Jain, Deepak Sharma
(Lecturer, Department of Computer Science, Lovely Professional University, Phagwara-144402)
(Assistant Professor, Department of Computer Science, Adesh Institute of Engineering and Technology,Faridkot-151203)

ABSTRACT :-Cloud Computing is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, datacenter and distributed servers. Central to these issues lies the establishment of an effective load balancing algorithm. The load can be CPU load, memory capacity, delay or network load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time.

Keywords **–** Cloud Computing, Distributed Systems, Elastic Compute Cloud, Load Balancing, Virtualization.

I. INTRODUCTION

Today, computing becomes steadily more important and more used. The amount of data exchanged over the network or stored on a computer is in constant increasing. Thus, the processing of this increasing mass of data requires more computer equipment to meet the different needs of organizations. To better capitalize their investment, the over equipped organizations open their infrastructure to others by exploiting the Internet and related technologies and other emerging technologies such as virtualization by creating a new computing model: the Cloud Computing. In [1] Cloud computing is defined as a model for delivering dynamically to IT end users, computing services (computing power,

Data, storage, software packages, programming environments ...) by a third party provider through a private or public network, using various advanced technologies and virtualization. Load balancing is also required to minimize the cost of machine and maximize the profit for the service being offered. The example of the MIT class in biological computing is given to explain the scenario. The professor of the course created a 10-node cluster to which students could submit work at any time, day or night. As students are known for procrastination, for the vast majority of the semester, the cluster sat idle with 10 EC2 nodes wasting money continuously for weeks. Only as the project's due date approached all 10 nodes were put to full use. These idle nodes should have been shut down, with only the master waiting for tasks. Thus, this proves that some of the nodes were heavily loaded while some others were just idle and wastage of all the resources used in here. To solve this problem, we use load balancing algorithms for distributed systems, but they are not fully adapted to the system of cloud computing which requires development of new algorithms or adaptation of those already existing for distributed systems.

Load balancing in [2] is the mechanism that decides which requesting nodes/client will use the virtual machine and which requesting machines will be put on hold. Load balancing can be done individually as well as on grouped basis. Load balancing is also required to minimize the cost of machine and maximize the profit for the service being offered.

Section II gives an overview about Previous work and other prerequisites for the setup. Section III explains load balancing concept and phases involved in it. Section IV proposes the algorithm for dynamic load balancing based upon the framework given in Section III. Results are shown in Section V. Finally, some conclusions are drawn in Section VI.

II. PREVIOUS WORK

A number of load balancing algorithms have been developed since the inception of this concept. A number of algorithms have been studied in order to implement this technique of load balancing. Some of the types of load balancer algorithms are as follows:

- **Sender Initiated**: If the load balancing algorithm is initialized by the sender.
- **Receiver Initiated**: If the load balancing algorithm is initiated by the receiver.
- **Symmetric**: It is the combination of both sender initiated and receiver initiated.

Depending on the current state of the system, load balancing algorithms can be divided into 2 categories as given in [4]:

Static: It does not depend on the current state of the system. Prior knowledge of the system is needed.
Dynamic: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach. Here we will discuss on various
dynamic load balancing algorithms for the clouds of different sizes.
Each one of them has their own advantages and some disadvantages but none of them would be complete without the discussion of the concept of Virtualization and virtual machines and categories of virtualization.

2.1 Previous Algorithm

The five phases of load balancing as described in [5] are:

- Load Evaluation
- Profitability Determination
- Work Transfer Vector Calculation
- Task Selection
- Task Migration

This Central Scheduler Load Balancing (CSLB) [6] uses a central node that makes all load balancing decisions. It decides when to migrate virtual machines between hosts and runs as a normal virtual machine. The aim behind this is, it can move itself to a different host like any other virtual machine, depending on the load.

2.2 VIRTUALIZATION AND LIVE MIGRATION

Virtualization [3] is commonly defined as a technology that introduces a software abstraction layer between the hardware and the operating system and applications running on top of it. Core of any virtualization technology is Hypervisor or Virtual Machine Manager (VMM) [8]. Hypervisor is a piece of software which allows each virtual machine to access and schedule the task on resources like CPU, disk, memory, network, etc. At the same time hypervisor maintains the isolation between different virtual machines. A computer on which a hypervisor is running one or more virtual machines is defined as a *host machine*. Each virtual machine is called a *guest machine*. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources. Virtualization can be classified by the method in which hardware resources are emulated to the guest operating system. Types are as follows:

2.2.1 Full Virtualization

Hypervisor controls the hardware resources and emulates it to guest operating system. In full virtualization, guest does not require any modification. KVM is an example of full virtualization technology.

2.2.2 Para Virtualization

In paravirtualization, hypervisor controls the hardware resources and provides API to guest operating system to access the hardware. In para virtualization, guest OS requires modification to access the hardware resources. Xen is an example of Para virtualization technology.

III. LOAD BALANCING

Load balancing is the process of reallocating VMs on another host in the network in order to improve resource and network utilization. Common goals of load balancing include maximizing throughput, minimizing response time, and/or minimizing communication time and avoiding the scenario in network that, some hosts are under-utilized and some over-utilized. The important factors to consider while developing such algorithm are estimation of load, comparison of load, performance of systems, nature of work to be transferred and selection of hosts [4].
Static load balancing is VM placement problem. Here, the host on which VM will be placed is decided before it starts running depending upon the load on the network i.e. host with least system usage runs the VM [9]. Dynamic load balancing reassigns VMs based on system performance at run time using the feature of live migration.

IV. PROPOSED ALGORITHM (ELASTIC LOAD BALANCER)

The Elastic Load Balancer algorithm is modified version of Central Scheduler Load Balancing (CSLB) algorithm [4]. The algorithm uses the six phases for load balancing as under:
1) Get Load Status of All the Nodes*:* In this paper, we set a scheduler which contains a Monitor to gain and read load status, and also a Database to store the load status and work request historical data of user access to the server (PM). Most of the current methods of nodes load status collection divided the system resource into several types: CPU utilization, Memory, Disk I/O and network bandwidth Etc. But with different size of servers or provide different services we cannot propose a unified set of those parameters.
2) Evaluate the Status Of nodes*:* We set a threshold that when the resource utilization beyond the threshold, we can considered compute as a over-load node, also if the resource utilization is under the threshold we know that the node is in a light-load status use and to represent those two statuses.
3) Predict The Future Load Flow: Based on the statistics, system's load status could show seasonal changes, which help to predict future load of nodes.

4) Benefit Estimates: When a load status of N is signed as which caused by transient spike, in this condition we cannot make the decision that whether we should perform migration.

5) Choose Receiver Nodes: We use the forward probability method to help us to choose a receiver host, every candidate nodes' probability to receive a job or VM is mainly depends on the result of load status evaluation.

6) Migration: Helps migration of the heavily loaded nodes to the lighter ones.

4.1 Overview of Load Balancing Algorithm

Once per polling interval, the load balancer will make a decision whether to add nodes, remove nodes, or do nothing. the balancer gets an accurate sense of what the clusters' load looks like.

These are the important questions to be answered:

1. Are there jobs queued and waiting? If so,
   a. How long have those jobs been waiting?
   b. Does past job history suggest that the queued waiting jobs will be finished quickly?
   c. Is the slave count already at the maximum count allowed?
2. If there are no jobs waiting,
   a. Are any slaves completely idle?
   b. Have those slaves been up for longer than 45 minutes past the hour?

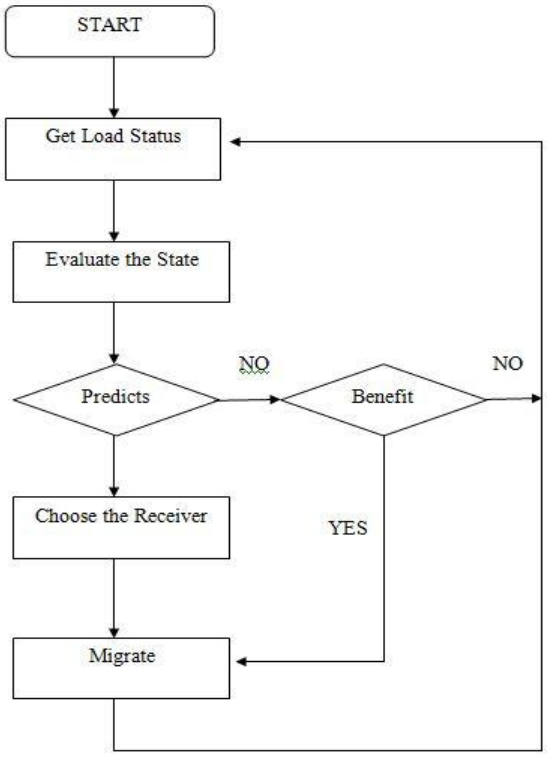

Fig 1: Flowchart showing six phases of algorithm

The relation between all the phases of the algorithm is explained by the flowchart (Fig 1).

V. PERFORMANCE AND RESULTS ANALYSIS

In order to assess the performance of the algorithm, a prototype system of VM management was developed. Virtual platform: KVM [7] and storage system: NFS was used. A physical machine was chosen as the host machine. VMM was installed to manage and schedule VM; and its operating system is UBUNTU. The MINIMUM characteristics of the host system are as follows: Intel Core i3 - 3110M CPU @ 2.40 GHz , 6 GB DDR3 RAM, 500 GB hard disk (3 spindles in RAID 0 configuration on Intel ICH8R SATA RAID controller), Windows 7 OS x64 (64-bit), VMware on Workstation 8.

Here, this section will compare two common operating scenarios for Star Cluster. In the first scenario, It will start a cluster and queue a series of jobs as if it were a scientist, and then leave the cluster to complete the jobs. This will run with no Elastic Load Balancing, and again with Elastic Load Balancing turned on. ELB will identify and terminate idle nodes, saving money for the scientist. In the second scenario, It will enqueue large sets of tasks in a seemingly random pattern. The pattern will be the same for the control case and for the ELB-enabled case. It will show how ELB launches new hosts to increase job throughput and terminates idle hosts to save money. Here, two test scenarios have been chosen because they are two commonly used cases for scientists that additionally demonstrate the true value of the Elastic Load Balancer.

5.1 Test Scenario 1

This scenario can be nicknamed "set it and forget it". The scientist queues up a large number of jobs, which the cluster immediately starts to execute. The jobs are executed serially by each of the nodes in the cluster. When one job completes, the node informs the master and requests another job. If a job or a node fails, the master re-queues that job so that the job will still be executed. The job count will be monotonically decreasing from the peak job count, when the scientist has just finished queuing jobs, to the low point, 0, when all jobs are complete. This is a common scenario because it allows the scientist maximum freedom. It is unlikely that a Star Cluster operator would be able to sit beside the computer and wait for all of his or her jobs to complete. If the nodes in the cluster are left running after the jobs are completed, their idle time is wasting money and unnecessarily tying up EC2 resources. As stated earlier, a 20-node High Performance Computing cluster will accrue a charge of $45.60 per hour. If the cluster were idle and all of the idle nodes were to be terminated, the cluster would only accrue a charge of $2.28 per hour for the master, until the scientist logged in to

terminate the cluster. This is desirable, and the $2.28 hourly charge is a negligible cost for keeping the master running.

In this scenario, the scientist would queue up a large number of jobs and leave the cluster unattended. He could come back at a later date to see that the jobs completed, and only the master would be running and can provide the results. The idle slaves have been terminated. The unattended job execution period could be as short as a few hours or as long as a few weeks. There are no limits.

5.2 Test Scenario 2

Test Scenario 2 models an unpredictable workload. This scenario would occur when a scientist creates a cluster to aid others, or enqueue a large number of tasks without knowing their durations.

5.3 RESULTS

5.3.1 Test Scenario 1: Case 1: Elastic Load Balancer Disabled

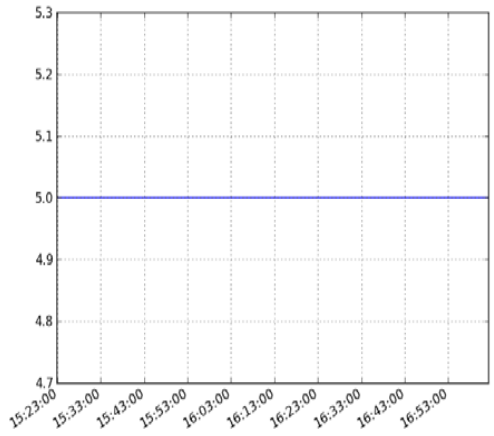

Fig 2:Test Scenario 1 showing idle nodes

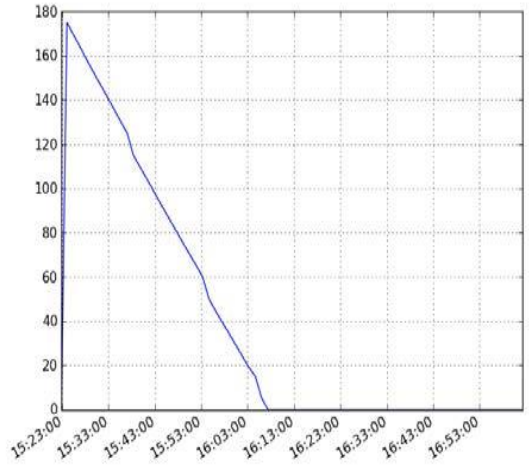

Fig 3: Test Scenario 1 showing idle running nodes

In the above graphs, the scientist has queued a large number of jobs and left the cluster unattended. The cluster of 5 nodes executes the jobs steadily, driving the number queued from 280 to 0. When the number of jobs queued reaches 0, the idle nodes are left running.

The above graphs describe the control case. At the beginning of the test, the tester queued 180 jobs. The cluster consists of 5 nodes, including the master. Each node in the cluster executes jobs at full speed, completing one job and requesting another job immediately. Since Elastic Load Balancing is not enabled, the nodes continue to sit idly until the tester comes back and shuts down the cluster when the test concludes at 17:00.

Case 2: Elastic Load Balancer Enabled

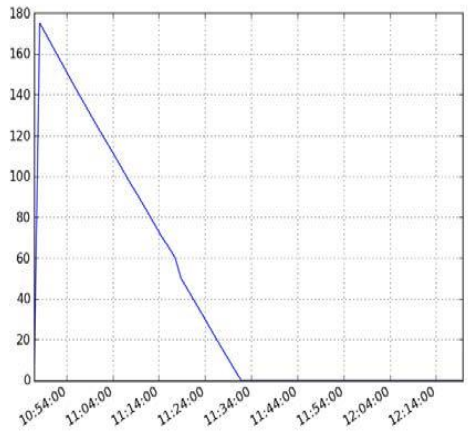

Fig 4: Test Scenario 2 showing large jobs enqueued again

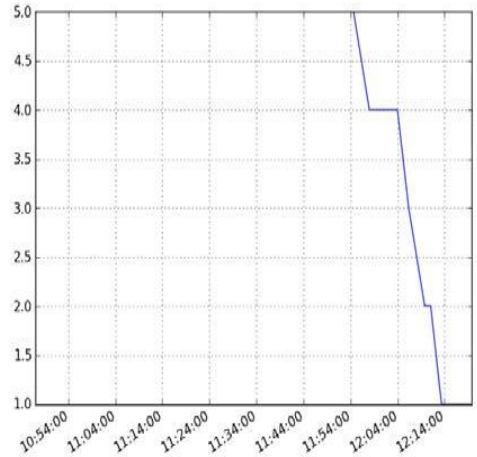

Fig 5: Test Scenario 2 shows nodes shut down acc to rule.

According to this 45-minute rule:

Load balancer grows and shrinks the cluster according to the length of the cluster's job queue. When the cluster is heavily loaded and

processing a long job queue, the load balancer can gradually add more nodes, up to the specified max_nodes, to distribute the work and improve throughput.. It helps in achieving the following goals:

1. To increase the size of the cluster to some user-defined maximum number of nodes when there is a large queue of waiting jobs
2. To decrease the size of the cluster to a single node or some minimum number of nodes when there are no jobs waiting to optimize for cost

5.3.2 Test Scenario 2 Results: Case 1: Elastic Load Balancer Disabled

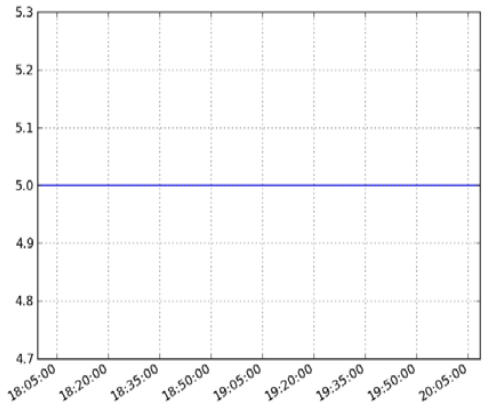

Fig.6: Case Scenario 2 showing cluster enqueued with larger number of nodes sitting idle

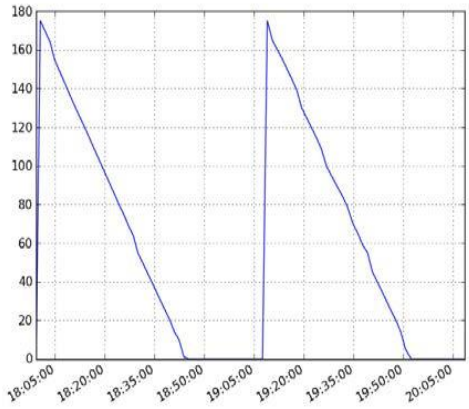

Fig. 7: Case Scenario 2 showing cluster enqueued with larger number of jobs (180+180 jobs).

In this case, over 2 hours, the number of hosts was constant. At approximately 18:05, 180 jobs were added to the queue. At approximately 19:05, 180 more jobs were added to the queue. The cluster was idle for approximately 25 minutes and no hosts were removed. Test Scenario 2 Control Case shows jobs queued at two distinct times: approximately 20:42 and 21:12. During the idle period between 18:45 and 19:10, the cluster sits idly. It is important to note that at 18:45, neither SGE nor ELB know that more work will be enqueued soon. There is no reason to keep slave nodes running in anticipation of future work.

Case 2: Elastic Load Balancer Enabled

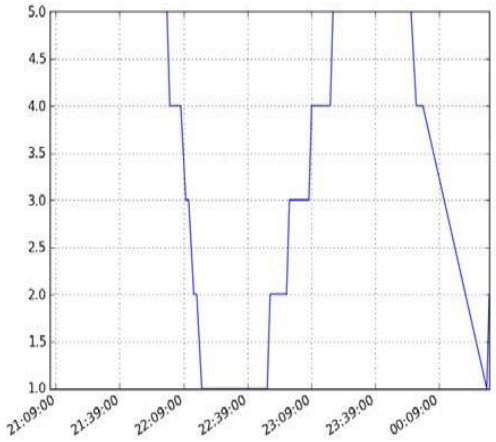

Fig. 8: Jobs enqueued at two different times

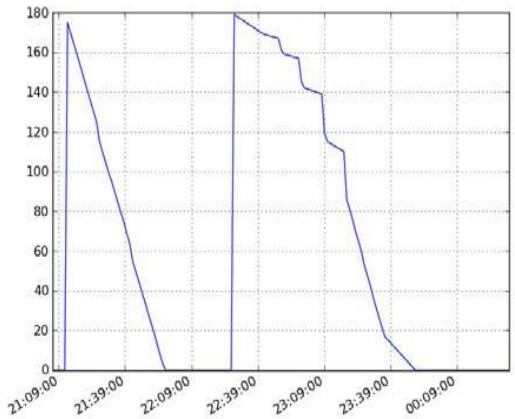

Fig. 9: Case Scenario 2 showing cluster enqueued with larger number of jobs (180+180 jobs) and ELB successfully shuts down the slave nodes.

VI. CONCLUSIONS

As we can see from the test results, ELB operates according to design. ELB can maintain a cluster to execute jobs with maximum possible throughput while the cluster is heavily loaded, and it removes nodes when the cluster is idle. The parameters to the load-balancing algorithm such as job wait threshold and stabilization time have been extensively tested in the laboratory and provide the ideal performance.

Table 1: Table Showing Pay Per Use Charges using Test Case Scenario 1

Conditions	ELB Disabled	ELB Enabled
Hours of compute time used	9	6
Total EC2 Charges	0.85	$0.595

Table 2: Table Showing Pay Per Use Charges using Test Case Scenario 2

Conditions	ELB Disabled	ELB Enabled
Hours of compute time used	12	8
Total EC2 Charges	$1.175	$0.835

ELB has the capability to save scientists' money and allow EC2 to optimally allocate resources by terminating idle nodes and freeing them for use by other customers. ELB will add new nodes and scale up the throughput of the cluster in response to heavy demand. Through the testing, even for tests of short duration, a noticeable cost savings is measured as it is clear from (Table 1 and 2).

Table 1: It shows pay per use charges for Case 1 in which the scenario is "set it and forget it". Here, number of jobs are (180) and compute time used was 9 hrs and 6 hrs for different cases. Cost of usage was depending upon its use. However,

Table 2: It shows pay per use charges for Case2 in which the scenario is "unpredictable workload". Here, number of jobs are (180+180) , double than previous scenario but compute time used was 12 hrs and 8 hrs for different cases.

Which would have been double otherwise.

Thus, this technique saved enormous amount of time and money making complete usage of Utility Computing.

Further dynamic load balancing can be improved and the Live Migration algorithm implemented in QEMU-KVM can be optimized, so that migration time will be reduced and performance will also be improved.

Journal Papers

[1] A. Khiyaita, M. Zbakkh, H. El Bakkali, and Dafir El Kettani. Load Balancing cloud Computing: State of Art. *IEEE computer society, May 2012*, 106-109.

[2] A Survey on Open-source Cloud Computing Solutions Patrícia Takako Endo, Glauco Estácio Gonçalves, Judith Kelner.

[3] Jyotiprakash Sahoo, Subasish Mohapatra, Radha Lath, Virtualization: A Survey On Concepts, Taxonomy And Associated Security Issues, Second International Conference on Computer and Network Technology, 2010.

[4] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security,VOL.10 No.6, June 2010.

[5] Jerrell Watts, Stephen Taylor, A Practical Approach to Dynamic Load Balancing, IEEE Transactions on Parallel and Distributed Systems, Feb 1998.

[6] Youran Lan, Ting Yu, A Dynamic Central Scheduler Load Balancing Mechanism, Computers and Communications, pp 734-740, May 1995.

[7] KVM Kernel Based Virtual Machine Red Hat, Inc. 2009.

[8] Geoffroy Vallee, Thomas Naughton, Christian Engelmann, Stephen L Scott, Hong Ong, System-level Virtualization and Virtual Machine Manager For High Performance Computing, 2008.

[9] Willebeek-LeMair M.H and Reeves A.P. "Strategies for Dynamic Load Balancing on Highly Parallel Computers," IEEE Transactions on Parallel and Distributed Systems, 1993, pp979.