Load Balancing Algorithm in Cloud Computing For Heterogeneous Configuration

Gaurang Sangani*, Shreyas Patel**
*(Department of Information Technology, PIET, GTU, Gujarat, India)
**(Asst. Professor, Department of Computer Science, PIET, Gujarat, India)

ABSTRACT
Cloud computing is a very rapidly growing technology in this days. An every organization is taking advantage of Cloud computing. Cloud computing is an Internet based technology in which virtual resources are provided as a service over the internet as per user demand. The more the number of users in the cloud, the greater load will be there. As the numbers of users are more in the cloud, balance the load has become the challenge for the cloud vender. Cloud service providers are required to balance this load effectively and efficiently so as not to degrade the performance. So, load balancing is a very important issue which aims at distributing the load across multiple machines in an even and fair manner so that no single node is overloaded. In this paper, we aim to manage the distribution of resources in such a manner so as to avoid system deadlock. This paper will be presenting the issues of existing load balancing algorithms and also presenting the algorithm to overcome those issues.

Keywords - Round Robin Algorithms, Throttled Algorithm, ESCE Algorithm, Load Balancing

I. INTRODUCTION
In recent years, the internet represent as cloud and its services. Cloud computing is a thing of distributed computing. Cloud computing is widely used in most organization. Overview of cloud computing is as shown on fig.1 in the cloud computing two issue is mostly important 1.security and 2. On demand service [1]. To achieve on demand service, cloud provider have to load balancing technique.[2] Now a days many load balancing technique is available to distribute load among the servers and virtual machine. Basic algorithms like round robin, throttled and ESCE (Equally Spread Current Execution). But some issue currently facing by cloud provider.

Load balancing is very important for efficient utilization of resources.[4] Load balancing is a mechanism which ensures that all the dynamic workload is distributed among all the multiple nodes in an evenly and efficient manner in such a way so as to avoid a condition where some nodes are heavily loaded while leaving some nodes to be idle or lightly loaded. Load balancing is the pre-requrement for increasing the cloud performance and completely utilizing the resources. It gives surety that every computing resource is distributed efficiently and evenly. Load balancing helps in reducing the bandwidth usage which results in decreasing the cost of machine and maximizing the services.[4] Load balancing aims to optimize the resource utilization, maximize the throughput, minimize the response time, and avoid overheads of any one of resource.

II. LOAD BALANCING ALGORITHMS
The effective load balancing algorithms and how to use of cloud computing resources effectively for effective and efficient is the goal of cloud computing service provider.

A. Round Robin Algorithm
It’s static load balancing algorithm, but, it does not save the previous load state of node. This is select first node randomly and then, allocates the job to all other as round robin manner.[5] Its work with random selection of VM.
DC (datacenter controller) assigns the request to VM based on circular order. When VM is allotted the request, it’s shifted at last in list.

Round robin is simplest algorithm to distribute the load among the nodes. Because of this it’s selected first when implementing a simple scheduler.

Step1. Round Robin VM load Balancer have an index of VMs. Initially all VM’s have zero allocation.
Step2.
a. The data center controller receives the user requests/cloudlets.
b. The requests are assigning to VMs in circular way.
c. The round robin VM load balancer will allocate the time quantum for user request execution.
Step3. After the execution of cloudlets/requests, the VMs are de-allocated by the Round Robin VM load balancer.
Step4. The data center controller checks for new/pending/waiting requests in queue.
Step5. Continue from step 2.[5]

B. Throttled Algorithm
Throttled load balancer is a dynamic load balancing algorithm. In throttled algorithm the client first requests the load balancer to find an appropriate Virtual Machine to process the task. The process first starts by maintaining a list of all the VMs. Each row is individually indexed to accelerate the lookup process. If a match is found on the basis of availability of the machine, then the load balancer accepts the request of the client and assigns that VM to the client. On the other hand, if there is no VM available, the load balancer returns -1 and the request is queued.[6] The throttled algorithm is as follows:

Step1. Throttled VM Load Balancer keeps an index table of VMs and the state of the VM (BUSY/AVAILABLE). At the beginning all VM’s are available.
Step2. Data Center Controller receives a request from user base.
Step3. Data Center Controller queries the Throttled VM Load Balancer for the next allocation.
Step4. Throttled VM Load Balancer parses the allocation table from top until the first available VM is found or the table is parsed completely.
If found:
i) The Throttled VM Load Balancer returns the VM id to the Data Center Controller.
ii) The Data Center Controller sends the request to the VM identified by that id.
iii) Data Center Controller notifies the Throttled VM Load Balancer of the new allocation.
iv) Throttled VM Load Balancer updates the allocation table accordingly.
If not found:
i) The Throttled VM Load Balancer returns -1.
ii) The Data Center Controller queues the request.
Step5. When the VM finishes processing the request, and the Data Center Controller obtains the response cloudlet, it informs the Throttled VM Load Balancer of the VM de-allocation.
Step6. The Data Center Controller checks if there are any waiting requests in the queue. If there are, it carries on from step 3.
Step7. Continue from step 2.[6]

C. Equally Spread Current Execution Algorithm
As per the name, this algorithm takes effort to equally spreading the execution load on VMs. Load balancer maintain the index table of VM with the number of tasks currently allocate to VM.[6] Load balancer always return least loaded VM for the execution of task. If there are more than one VM found then first identified virtual machine selected.[6] After allocated the request load balancer update the table. For each and every request, load balancer need to update table. So, there is an additional computational needed for the search table again and again.

D. Weighted Active Monitoring Load Balancing Algorithm
The ‘Weighted Active Monitoring Load Algorithm’ is implemented; modifying the Active Monitoring Load Balancer by assigning a weight to each VM as discussed in Weighted Round Robin Algorithm of cloud computing in order to achieve better response time and processing time.[7]
Step to be followed as below.

STEP 1: Create VMs of different Datacenter according to computing power of host/physical server in terms of its core processor, processing speed, memory, storage etc.

STEP 2: Allocate weighted count according to the computing power of the VMs in Datacenter. If one VM is capable of having twice as much load as the other, the powerful server gets a weight of ‘2’ or if it can take four times load then server gets a weight of ‘4’ and so on. For example:

• Host server with single core processor, 1GB of memory, 1TB of Storage space, 1000000 bandwidth will have weighted count=1
• Host server with 2 core processor, 4GB of memory, 2TB of Storage space and 1000000 bandwidth will have weighted count=2
• Host server with quard core processor, 8GB of memory 4TB of Storage space and 1000000 bandwidth will have weighted count=4 and so on..

STEP 3: Weighted Active Vm Load Balancer maintains an index table of VMs, associated weighted count and the number of requests currently allocated to the VM. At start all VM's have 0 allocations.

STEP 4: When a request to allocate a new VM from the Data Center Controller arrives, it parses the table and identifies the least loaded VM.

STEP 5: After identifying the least loaded VM’s in different datacenters, it allocate requests to the most powerful VM according to the weight assigned. If there are more than one, the first identified is selected.

STEP 6: Weighted Active Vm Load Balancer returns the VM id to the Data Center Controller.

STEP 7: The Data Center Controller sends the request to the VM identified by that id.

STEP 8: Datacenter Controller notifies the Weighted Active Vm Load Balancer of the new allocation.

STEP 9: Weighted Active Vm Load Balancer updates the allocation table increasing the allocations count for that VM.

STEP 10: When the VM finishes processing the request, and the Data Center Controller receives the response cloudlet, it notifies the Weighted Active Vm Load Balancer of the VM de-allocation.

STEP 11: The Weighted Active Vm Load Balancer updates the allocation table by decreasing the allocation count for the VM by one.

STEP 12: Continue from step 4.[7]

III. PROPOSE ALGORITHM

The different approaches offer specific solutions for load balancing that not suit at every situation. The static algorithms are usually very efficient in terms of overhead as they do not need to monitor the resources during run-time. Therefore, they would work very well in a stable environment where operational properties do not change over time and loads are generally uniform and constant.

Flowchart of proposed approach is as below.

In proposed approach we follow below steps.
1. Create queue of incoming task/cloudlet.
2. Create queue of VM.
3. Allocate the request to the VM.

IV. CONCLUSION AND FUTURE WORK

As such cloud computing being wide area of research and one of the major topics is dynamic load balancing in heterogeneous configuration. Proposed algorithm provides efficient approach to handle the request in heterogeneous configuration. Proposed algorithm mainly focuses on to efficient use of virtual machine. The future work is to Implement this algorithm and compare the result with all basic algorithms.

REFERENCES


www.ijaert.org