ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 6, June 2014

Unified Load Balancing System for Cloud

Prasad P.Sawant[1], Varshapriya J.N[2]

MTech NIMS, VJTI Matunga, Mumbai, India[1]

Department of computer Engineering, VJTI Matunga, Mumbai, India[2]

Abstract: Cloud Computing adds more power to the existing Internet technologies. Virtualization harnesses the power of the existing infrastructure and resources. With virtualization we can simultaneously run multiple instances of different commodity operating systems. Since we have limited processors and jobs work in concurrent fashion, overload situations can occur. There are many types of load balancing algorithms available for cloud for increasing performance. In this paper we propose a Unified Load Balancing system which consists of static load balancer which forwards the incoming ip request to specified node and a workload manager which manages working load on all computing node. This system makes efficient use of power by moving VM from light weighted node thus making him idle.

Keywords: Resource Monitoring, IaaS, PaaS, SaaS, Cloud, Computing, Load Balancing.

I. INTRODUCTION

Cloud computing means that instead of all the computer hardware and software we are using sitting on our desktop, or somewhere inside our company's network, it's provided for us as a service by another company and accessed over the Internet, usually in a completely seamless way. Exactly where the hardware and software is located and how it all works doesn't matter to us, the user—it's just somewhere up in the nebulous "cloud" that the Internet represents on its own.

Cloud computing contains many content like virtualization, distributed computing, networking, advance version to grid computing, software and web services, etc. A cloud consists of components like Cloud controller, cluster controller, nodes, datacenters and servers. It composed of fault tolerance, high availability, scalability, flexibility, reduced cost of ownership, reduced overhead for users, less maintainability, on demand services.

Addressing to this issue needs the foundation of efficient 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.

There is no standard definition of Cloud computing. Generally it consists of a bunch of distributed servers known as masters, providing demanded services and resources to different clients known as clients in a network with scalability and reliability of datacenter. The distributed computers provided On-demand services. Services may be of software resources (e.g. Software as a Service, SaaS) or physical resources (e.g. Platform as a Service, PaaS) or hardware/infrastructure (e.g. Hardware as a Service, HaaS or Infrastructure as a Service, IaaS).In this paper we propose a Unified Load Balancing system which consists of static load balancer which forward the incoming ip request to specified node and workload manager that maintains the utilization of all compute nodes and distributes virtual machines in a way that is power efficient. The goal of this algorithm is to maintain availability to compute nodes while reducing the total power consumed by the cloud.

II. NEED FOR LOAD BALANCING IN CLOUD

Load balancing is a process by which inbound IP traffic can be distributed across multiple servers. It enhances the performance, leads to optimal utilization and ensures that no single server is overwhelmed. We can have multiple servers in a server farm or a data centre, which can host multiple guests. Each guest may dier on load, leading to a situation where some of the servers may become overwhelmed or capitulated in terms of computational resources, memory resources and I/O devices. For simplicity and without loss of generality, we will consider load in terms of CPU time. When a server A is unable to allocate su_cient CPU time slice due to heavy demand by other VMs running parallel and another server B is idle, we can redistribute the load from A to B by migrating few guests to B. This would be an ideal situation when we require load balancing policy in Cloud Computing scenario, improving the percentage of server idle times, marginal job response times, etc. In order to achieve, short response time and high system throughput, we need to consider the following characteristics:

- the load balancing process generates little trace overhead and adds low overhead on the computational and network resources.
- It keeps upto date load information of the participating systems.
- It balances the system uniformly and takes action instantaneously or on a periodic basis.
- It can run on a dedicated system, or it can be a de-centralized e_ort.
- The available server should have sufficient resources available to host and run the migrated guest.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 6, June 2014

- The live migration of complete operating system should take minimum acceptable time with minimum downtime.
- Network communication should be reliable and fast.

III. RELATED WORK

Load balancing in cloud environment is done at two stages.

1. Static load balancing at cloud controller.
2. Workload managing or dynamic load balancing throughout the system.

a) Static Load Balancing

The basic principle is that network traffic is sent to a shared IP in many cases called a virtual IP (VIP), or listening IP. This VIP is address that it attached to the load balancer. Once the load balancer receives a request on this VIP it will need to make a decision on where to send it. This decision is normally controlled by a "load balancing method/ strategy.

Common Strategies-

1. Round robin:

The most simple method, each server takes a turn. Define endpoints that are associated to specific ports. An endpoint can be assigned a protocol of TCP or UDP. Each endpoint defined for a virtual machine is assigned a public and private port for communication. Load balancing done on public ports**.**

2. Least number of connections:

The load balancer will keep track of the number of connections a server has and send the next request to the server with the least connections.

3. Weighted:

Typically servers are allocated a percentage capability as one server could be twice as powerful as another. Weighted methods are useful if the load balancer does not know the real and actual performance of the server.

4. Fastest response time

This method is normally only available on more advanced products. The request will be sent to the fastest responding server

b) Workload managing or dynamic load balancing

- The more sophisticated load balancers are workload managers. They determine the current utilization of the resources in their pool
- Feature of these load balancers are polling resources for their health, the ability to bring standby servers online, workload weighting based on a resource's capacity, HTTP traffic compression, TCP offload and buffering, security and authentication, and packet shaping using content filtering and priority queuing.
- An Application Delivery Controller (ADC) is a combination load balancer and application server that is a server placed between a firewall or router and a server farm providing Web services.

Common Technique-

i. Global:-

- Every P seconds, one of the hosts, designated as the load information center (LIC), receives load updates from all the other hosts and assembles them into a loud vector, which is then broadcast to all the other hosts.
- The local version of the load vector is searched for a host with the lowest load, and if this load is lower than that of the local host by A' or more, the job is sent to that host

ii. Disted:-

- Instead of reporting the local load to a centralized LIC as in GLOBAL, each host broadcasts its load periodically for the other hosts to update their locally maintained load vector.

iii. Random:-

- This algorithm uses only local load information. When a job is found to be eligible for load balancing, it is sent to a randomly selected host.

iv. Threshold

- A number of randomly selected hosts, **up** to a limit *Lp* (probing limit), are polled when an eligible job arrives.
- The job is transferred to the first host whose load is below load threshold *TI.*
- If no such host is found, the job is processed locally.

v. Lowest

- In this a fixed number of hosts *L,,* are polled and the most lightly loaded host is selected. the job is transferred to the first host whose load is below load threshold *TI.*
- The probing stops if an empty host is found.

vi. Central:-

- In the CENTRAL algorithm, the LIC acts as a central scheduler for all the hosts.
- When a host decides that a job is eligible for load balancing, it sends a request to the LIC, together with the current value of its load.
- The LIC selects a host with the shortest queue length and informs the originating host to send the job there.

c) Load Balancing using agent aid model:-

- There is series of agents, which will monitor the data nodes and conduct information acquisition.
- Overload nodes requests computing resources from other nodes, and copy their data to the new nodes that have more resources.
- Before a data migration agreement is reached, agents are required to interact with others to find who can provide those resources.

TOKEN system is used

- Let **TOKEN**=*{Δ1,Δ2,Δ3, . . . }*
- Each token contains four parts: Δ=< Resource, Path, Threshold, Lifetime>.
- **Resource** identifies the block size of data for copy.
- **Path** records the sequences of agents that the token visited.
- **Threshold** defines the threshold that an agent is required to accept token.
- **Lifetime** defines its time length allowed to exist in the network.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 6, June 2014

- The basic decision model of agent *a for a token Δ can be* written as a POMDP <*S, Actiona, T, Θa, O, R*>.

d) Biased Random Sampling:-

This approach use a virtual graph for load balancing. The network is constructed by creating links between randomly selected nodes. When a job arrive at an node the walk starts and continue until log(n) nodes are not traversed, where n is no of processing nodes. At each hop a random neighbor is selected and walk length is incremented by one. when a node receive a job it will do the final processing if the total walk length is greater than the walk length threshold otherwise continuing the Random Sampling.

Random Sampling is done on two basis-

a. Pending job queue length
b. Pending job processing time

IV. PROPOSED SYSTEM

A. Static Load Balancer-

A Cloud controller Node will accept the incoming HTTP request and forward it in ROUND ROBIN fashion.

B. System utilization monitor.

1) Monitor server –

This will receive system utilization information from Monitor client on periodic basis. Based on Data gathered Source Virtual Machine and Destination Compute Node will be decided.

2) Monitor Client –

- These processes will be running on Compute Nodes.
- These processes will gather data about current system utilization and send report periodically to Monior server.

C. VM Migrator-

At certain interval of time a **power management algorithm** will run and decide which VM to be migrated. The algorithm also finds source node and destination node After receiving source VM and Destination Node this process will migrate the VM.

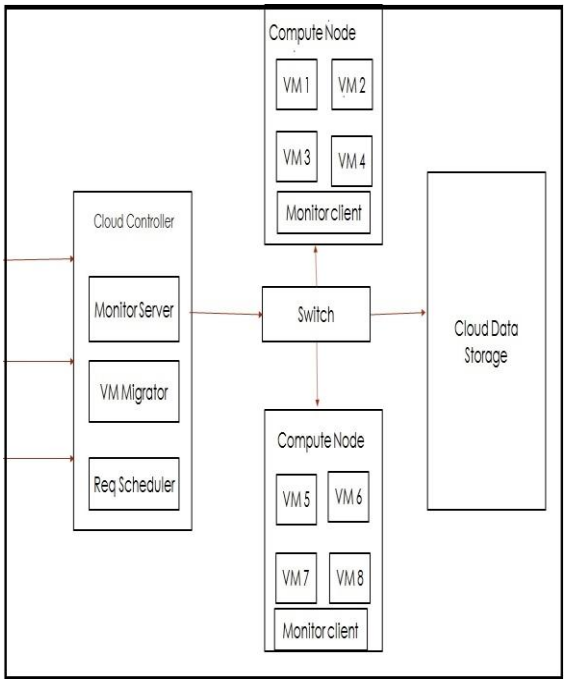
POWER MANAGAMENT ALGORITHM

```
balance:
for all active compute nodes j ∈ [m] do
nj   current utilization of compute node j
end for
if all nj > 75% utilization //all available nodes are
highly utilized
        do nothing
end if
else
        for all nj < 25% utilization
        //underutilized node
        find [m] belong to 25% < nj < 50%

                if found
                  migrate VM to [m]
                  shut down Node
                end if

                else
                  find [m] belong to nj < 25%
                        if found
                        migrate VM to [m]
                        shut down Node
                        end if
                        else
                        do nothing
```

D. USING THE TEMPLATE

After the text edit has been completed, the paper is ready

SYSTEM MODEL



IV. CONCLUSION AND FUTURE WORK

Thus in this project we are creating a unified system consist of "Request scheduler" and " Workload manager" which will provide better efficiency of cloud system through better utilization of all the system.

Basis of the system is "Monitor server" which keeps the track of "utilization of the nodes.

This data can be used by cloud controller for performing different functions.

In future we can make this user configurable so that user can choose the parameter on which system is based like throughput. Also this system can be generalized for large enterprise cloud where there are many cloud controller and large no of compute node.

ACKNOWLEDGMENT

I want to thank Dr Bandu Meshram. of VJTI. Mumbai for his valuable guidance.

REFERENCES

[1] Abhay Bhadani , Sanjay Chaudhary, Gandhinagar, INDIA Performance Evaluation of Web Servers using Central Load Balancing Policy over Virtual Machines on Cloud. Compute'10, Jan 22-23, 2010, Bangalore, Karnataka, India
[2] Yang Xu, Lei Wu, Liying Guo, Zheng Chen, Lai Yang, Zhongzhi Shi, An Intelligent Load Balancing Algorithm Towards Efficient Cloud Computing
[3] IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 18, NO. 2, FEBRUARY 1992, A Dynamic Load-Balancing Policy With a Central Job Dispatcher (LBC), Hwa-Chun Lin and C. S. Raghavendra, Senior Member, IEEE
[4] IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 9, SEPTEMBER 2011, MAP-JOIN-REDUCE: Toward Scalable and Efficient Data Analysis on Large Clusters, Dawei Jiang, Anthony K. H. Tung, and Gang Chen.

[5] IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Hint-based Execution of Workloads in Clouds with Nefeli, Konstantinos Tsakalozos, Mema Roussopoulos, and Alex Delis.
[6] ACM 978-1-4503-0976-9/11/10, Small Cache, Big Effect: Provable Load Balancing forRandomly Partitioned Cluster Services, Bin Fan, Hyeontaek Lim, David G. Andersen, Michael Kaminsky.
[7] IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 61, NO. 5, JUNE 2012, An SMDP-Based Service Model for Interdomain Resource Allocation in Mobile Cloud Networks, Hongbin Liang, Student Member, IEEE, Lin X. Cai, Member, IEEE, Dijiang Huang, Senior Member, IEEE, Xuemin (Sherman) Shen, Fellow, IEEE, and Daiyuan Peng, Member, IEEE
[8] ACM 978-1-60558-326-6/09/06, The Cloud-based Framework for Ant Colony Optimization, Zhiyong Li, Yong Wang, Kouassi K. S. Olivier, Jun Chen, Kenli Li
[9] IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14, NO, 9, SEPTEMBER 1988, A Trace-Driven Simulation Study of Dynamic Load Balancing, SONGNIAN ZHOU
[10] WWW.Wikipedia.org/Loadbalancing
[11] Document and reference provided by www.openstack.org