Improving Performance in Public Cloud using Cloud Partitioning and Game Theory

Deepak B S
Dept. of CSE, NIE Mysore
Karnataka, India
bsdeepak012@gmail.com

Sanjay Kumar C K
Dept. of CSE, NIE Mysore
Karnataka ,India
sanjaykumarck@gmail.com

Radhika K R
Dept. of CSE, MIT Mysore
Karnataka, India
radhikakr2005@gmail.com

***Abstract*— Cloud computing is emerging as a new paradigm of large-scale distributed computing. Load balancing is one of the main challenges in cloud computing which is required to distribute the dynamic workload across multiple nodes to ensure that no single node is overwhelmed. In cloud computing, load balancing is required to distribute the dynamic local workload evenly across all the nodes. It helps to achieve a high user satisfaction and resource utilization ratio by ensuring an efficient and fair allocation of every computing resource. This paper introduces a better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. The algorithm applies the game theory to the load balancing strategy to improve the efficiency in the public cloud environment.**

Keywords—cloud; cloud computing; load balancing; game theory; cloud partitoning.

I. INTRODUCTION

Cloud computing is emerging as a new paradigm of large scale distributed computing. It has moved computing and data away from desktop to portable PCs into large data centers [1]. It has the capability to harness the power of Internet and wide area network to resources that are available remotely, thereby, providing cost effective solution to the most of the real life requirement. It provides the scalable IT resources such as applications and service, as well as infrastructure on which they operate, over the Internet, as pay-per- use basis to adjust the capacity quickly and easily. It helps to accommodate changes in demand. Thus cloud computing is a framework for enabling a suitable, and resource utility of the system. It also ensures for the fair distribution of work and resources.

Load balancing in computer networks is a technique used to spread workload across multiple network links of computers [2]. It facilitates networks and resources by providing a maximum throughput with minimum time, thus it helps to improve performance by optimally using available resources and helps in minimizing latency and response time. Load balancing is achieved by using multiple resources that is, multiple servers that are able to fulfill a request or by having multiple paths to a resource. Load balancing helps to achieve a high user satisfaction and resource utilization. When one or more components of any service fail, load balancing facilitates continuation of the service by implementing fair-over, that is, it helps in provisioning and de-provisioning of instances of applications without fail. It also ensures that every computing resource is distributed efficiently and fairly [3].

Consumption of resources and conservation of energy is not always a prime focus of discussion in cloud computing. However, resource consumption can be kept to minimum with proper load balancing which not only helps in reducing costs but making enterprise greener. One of the very important features of cloud computing, is also enabled by load balancing. Hence, improving resource utility and the performance of a distributed system in such a way will reduce the energy consumption and carbon footprints to achieve Green computing [1].

The real world example of load balancing can be a website which has thousands of users at the same time. If not balanced then the users have to face the problem of timeouts, response delays and long processing time. The solutions involve making use of duplicate servers to make the website available by balancing the network traffic.

II. CHARACTERISTICS

According to the National Institute of Standards and Technology (NIST) [4], cloud computing exhibits several characteristics:

On-demand Self-service- A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad Network Access- Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource Pooling- The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

There is a sense of location independence in that the customer generally has no control or knowledge over the exact

location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid Elasticity- Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured Service- Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

III. RELATED WORK

Before There have been many studies of load balancing for the cloud environment. Load balancing in cloud computing was described in a white paper written by Adler[5] who introduced the tools and techniques commonly used for IEEE TRANSACTIONS ON CLOUD COMPUTING YEAR 2013 load balancing in the cloud. However, load balancing in the cloud is still a new problem that needs new architectures to adapt to many changes. Chaczko et al.[6] described the role that load balancing plays in improving the performance and maintaining stability. some of the algorithms are discussed below.

Round Robin: In this algorithm [7], the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. This algorithm is mostly used in web servers where http requests are of similar nature and distributed equally.

Equally Spread Current Execution Algorithm:

Equally spread current execution algorithm [8] process handle with priorities. it distribute the load randomly by checking the size and transfer the load to that virtual machine which is lightly loaded or handle that task easy and take less time , and give maximize throughput. It is spread spectrum technique in which the load balancer spread the load of the job in hand into multiple virtual machines.

Throttled Load Balancing Algorithm: Throttled algorithm [8] is completely based on virtual machine. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is give by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

Token Routing: The main objective of the algorithm [2] is to minimize the system cost by moving the tokens around the system. But in a scalable cloud system agents cannot have the enough information of distributing the work load due to communication bottleneck. So the workload distribution among the agents is not fixed. This algorithm provides the fast and efficient routing decision. In this algorithm agent does not need to have an idea of the complete knowledge of their global state and neighbor's working load. To make their decision where to pass the token they actually build their own knowledge base. This knowledge base is actually derived from the previously received tokens. So in this approach no communication overhead is generated.

IV. SYSTEM ARCHITECTURE

In this paper the main focus is on the public cloud. A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. The architecture is shown in the Figure 1.

Figure 1: Cloud partition based on geographical location

A. Main controller and balancers

The load balancing strategy is based on the cloud partitioning concept. After partitioning the cloud, the load balancing will starts: when a job arrives at the system, with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally. If the cloud partition load status is not normal, this job should be transferred to another partition. The load balance solution is done by the main controller and the balancers. The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals

with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs. The relationship between the balancers and the main controller is shown in Figure 2.

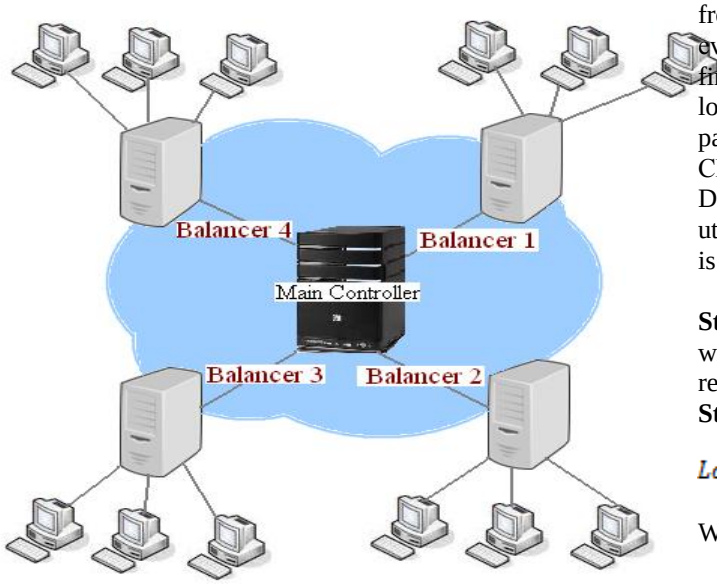

Figure 2 : Load balancing using main controller and balancers

B. Assigning jobs to the cloud partition

When a job arrives at the public cloud, the first step is to choose the right partition. The cloud partition status can be divided into three types:

(1) **Idle:** When the percentage of the nodes exceeds the parameter α then it is in idle state.

(2) **Normal:** When the percentage of the nodes exceeds the parameter β then it is in normal state.

(3) **Overload**: When the percentage of the nodes exceeds the parameter γ then it is in overload state.

The parameters α, β and γ are set by the cloud partition balancers. The main controller has to communicate with the balancers frequently to refresh the status information. The main controller then dispatches the jobs using the following strategy: When job i arrives at the system, the main controller queries the cloud partition where job is located. If this location's status is idle or normal, the job is handled locally. If not, another cloud partition is found that is not overloaded. The algorithm is shown below:

Algorithm 1: Best Partition Searching

step 1: begin
step 2: when job arrives for partition at main controller it searches for best job partition
step 3: if partion_status == idle or partition_status == normal
then send job to partition
else
search for another partition
step 4: go to step 3
step 5: end

C. Assigning jobs to the nodes in the cloud partition

The cloud partition balancer gathers load information from every node to evaluate the cloud partition status. This evaluation of each node's load status is very important. The first task is to define the load degree of each nodes. The node load degree is related to various static parameters and dynamic parameters. The static parameters include the number of CPU's, the CPU processing speeds, the memory size, etc. Dynamic parameters are the memory utilization ratio, the CPU utilization ratio, the network bandwidth, etc. The load degree is computed from these parameters as below:

Step 1: Define a load parameter set: F = {F_1, F_2,……., F_m} with each parameter being either static or dynamic. m represents the total number of the parameters.
Step 2: Compute the load degree as:

$$Load\_Degree(N) = \sum_{i=1}^{m} \alpha_i F_i$$

Where,
-weights for jobs.
N-current node.
Step 3: Calculate the average cloud partition degree from the node load degree statistics as:

$$Load\_Degree_{avg} = \frac{\sum_{i=1}^{n} Load\_Degree(N_i)}{n}$$

Step 4: Three nodes load status levels are then defined as:

- Idle When Load_degree(N) = 0; there is no job being processed by this node so the status is charged to Idle.
- Normal For 0 < Load degree(N) ≤ $Load\_degree_{high}$, the node is normal and it can process other jobs.
- Overloaded When $Load\_degree_{high}$ ≤ Load_degree(N) the node is not available and can not receive jobs until it returns to the normal.

The load degree results are input into the Load Status Tables created by the cloud partition balancers. Each balancer has a Load Status Table and refreshes it each fixed period T. The table is then used by the balancers to calculate the partition status. Each partition status has a different load balancing solution. When a job arrives at a cloud partition, the balancer assigns the job to the nodes based on its current load strategy. This strategy is changed by the balancers as the cloud partition status changes.

V. IMPLEMENTATION

A. Load balance strategy for the idle status

When the cloud partition is idle, many computing resources are available and relatively few jobs are arriving. In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method

can be used. But here since the public cloud is used, the configuration and the performance of each node will be not the same; thus, an improved Round Robin algorithm is used , which is called "Round Robin based on the load degree evaluation". Initially, the nodes in the load balancing table are ordered based on the load degree from the lowest to the highest. The system builds a circular queue and walks through the queue again and again. Jobs will then be assigned to nodes with low load degrees. The node order will be changed when the balancer refreshes the Load Status Table. However, there may be read and write inconsistency at the refresh period. To resolve this problem, two Load Status Tables should be created as: Load Status Table 1 and Load Status Table 2. A flag is also assigned to each table to indicate Read or Write. When the flag = "Read", then the Round Robin based on the load degree evaluation algorithm is using this table. When the flag = "Write", the table is being refreshed, new information is written into this table.

Algorithm 2: Round Robin

Step 1: Let Server A is an Overloaded
Step 2: Let s[n] is an array consists of server which are in **Idle** state.
Step 3: Let c=1.
Step 4: if new connection Came for Server A
Then
Send the connection to s[c]
After that make c=c+1;
If c==n
then c=1
Else
Wait;
Step 5: go to step 3.

B. Load balancing strategy for the normal status

When the cloud partition is normal, jobs are arriving much faster than in the idle state and the situation is far more complex, so a different strategy is used for the load balancing. The load balancing in the cloud computing environment can be viewed as a game. Game theory has non-cooperative games and cooperative games. In cooperative games, the decision makers eventually come to an agreement which is called a binding agreement. Each decision maker decides by comparing notes with each others. In non-cooperative games, each decision maker makes decisions only for his own benefit. The system then reaches the Nash equilibrium, where each decision maker makes the optimized decision. The Nash equilibrium is when each player in the game has chosen a strategy and no player can benefit by changing his or her strategy while the other players strategies remain unchanged. So the load balancing strategy for a cloud partition in the normal load status can be viewed as a non cooperative game. In game theoretic approach we are using three different algorithms based on three different network parameters no of connections with server, distance between the overloaded server and other servers and bandwidth between the servers. these three servers are as shown below

Game Theoretic algorithms:

Algorithm 3: (based on distance between servers)

Step 1: Let Server A is an Overloaded
Step 2: Let w[n] is an array consists of server which are in *Normal* state. n is the total number of server.
Step 3: if new connection Came for Server A
Then
Calculate distance of server A and w[n]'s.
Select minimum distance server, let it be i
Send the connection to w[i]
Else
Wait;
Step 4: go to step 3.

Algorithm 4: (based on number of connection of servers)

Step 1: Let Server A is an Overloaded
Step 2: Let w[n] is an array consists of server which are in *Normal* state. n is the total number of server.
Step 3: if new connection Came for Server A
Then
Calculate C1,C2 ... Cn of server w[1] ... w[n]'s.
Select server with minimum C, let it be i
Send the connection to w[i]
Else
Wait;
Step 4: go to step 3.

Algorithm 5: (based on bandwidth between servers)

Step 1: Let Server A is an Overloaded
Step 2: Let w[n] is an array consists of server which are in *Normal* state. n is the total number of server.
Step 3: if new connection Came for Server A
Then
Calculate bandwidth of w[n]'s.
Select maximum bandwidth server,let it be i
Send the connection to w[i]
Else
Wait;
Step 4: go to step 3.

VI CONCLUSION

In this paper we have proposed a game theoretic approach which provides better efficiency towards the resolution of load balancing in large public clouds. It is believed that designing of network matches characteristics of clouds, the efficiency and performance can be greatly improved. This approach also has measurably improved availability of resource and its utilization in the cloud environment by using the algorithms like round robin and game theoretic algorithms based on network parameters like bandwidth, distance and number of connections with server. Thus this approach has achieved better load balancing by improving the efficiency, performance than other algorithms and has improved the user satisfaction.

VII FUTURE WORK

Since this work is just a conceptual framework, more work is needed to implement the framework and resolve new problems. Some important points are:

a. Cloud division rules: Cloud division is not a simple problem. Thus, the framework will need a detailed cloud division methodology. For example, nodes in a cluster may be far from other nodes or there will be some clusters in the same geographic area that are still far apart. The division rule should simply be based on the geographic location (province or state).

b. How to set the refresh period: In the data statistics analysis, the main controller and the cloud partition balancers need to refresh the information at a fixed period. If the period is too short, the high frequency will influence the system performance. If the period is too long, the information will be too old to make good decision. Thus, tests and statistical tools are needed to set a reasonable refresh periods.

c. A better load status evaluation: A good algorithm is needed to set Load degreehigh and Load degreelow, and the valuation mechanism needs to be more comprehensive.

d. Find other load balance strategy: Other load balance strategies based other network parameters may provide better results, so tests are needed to compare different strategies. Many tests are needed to guarantee system availability and efficiency.

.

REFERENCES

[1] Nidhi Jain Kansal, Inderveer Chana, Cloud Load balancing techniques:A step towards green computing, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[2] Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanzadeh and Christopher Mcdermid, Availabilty and load balancing in cloud computing, 2011 International Conference on Computer and Software Modeling, IPCSIT vol.14 (2011) ACSIT Press, Singapore.

[3] Tanveer Ahmed, Yogendra Singh, Analytic study of load balancing techniques using tool cloud analyst.

[4] Peter Mell, Timothy Grance, "The NIST Definition of Cloud Computing", NIST Special Publication 800-145, September 2011.

[5] B. Adler, Load balancing in the cloud: Tools, tips and techniques, http://www.rightscale.com/infoenter/whitepapers /Load-Balancing-in-the-Cloud.pdf, 2012

[6] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, Availability and load balancing in cloud computing, presented at the 2011 International Conference on Computer and Software Modeling, Singapore, 2011

[7] Zhong Xu, Rong Huang,(2009) "Performance Study of Load Balanacing Algorithms in Distributed Web Server Systems", CS213 Parallel and Distributed Processing Project Report.

[8] Ms.NITIKA, Ms.SHAVETA, Mr. GAURAV RAJ; "Comparative Analysis of Load Balancing Algorithms in Cloud Computing", IJARET Volume 1, Issue 3, May 2012.