THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Efficient Parallel Processing on Public Cloud Servers using Load Balancing

Manjunath K. C
M.Tech IV Sem, Department of CSE, SEA College of Engineering & Technology, Bangalore, India
Dr. B. R Prasad Babu
Professor & HOD, Department of CSE, SEA College of Engineering & Management, Bangalore, India
Shaik Md Ghouse
Assistant Professor, Department of CSE, SEA College of Engineering & Management, Bangalore, India

Abstract:
Load balancing in the cloud differs from classical thinking on load-balancing architecture and implementation by using commodity servers to perform the load balancing. This provides for new opportunities and economies-of-scale, as well as presenting its own unique set of challenges. This paper proposes load balancer model based on cloud partitioning concept. The model uses switch mechanism depending upon the load status at the cloud partition when the request is made. Switch mechanism allows choosing different strategies in different situations. This load balancing strategy applies to the public cloud to improve efficiency in the public cloud environment.

Keywords *: Cloud computing, Equally Spread Current Execution, Load balancing, Round robin, public cloud*

1. Introduction

Cloud computing is an attracting technology in the field of computer science. In Gartner's report [1], it says that the cloud will bring changes to the IT industry. The cloud is changing our life by providing users with new types of services. NIST gave a definition of cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

Since the job arrival pattern is not predictable and the capacities of each node in the cloud differ, for load balancing problem, workload control is crucial to improve system performance and maintain stability. Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic [2].A dynamic scheme is used here for its flexibility. The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy. The load balancing model given in this paper is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy.

2. Load balancing

Load balancing is a relatively new technique that facilitates networks and resources by providing a maximum throughput with minimum response time [2]. Dividing the traffic between servers, data can be sent and received without major delay. Different kinds of algorithms are available that helps traffic loaded between available servers. A basic example of load balancing in our daily life can be related to websites. Without load balancing, users could experience delays, timeouts and possible long system responses. Load balancing solutions usually apply redundant servers which help a better distribution of the communication traffic so that the website availability is conclusively settled [2].There are many different kinds of load balancing algorithms available, which can be categorized mainly into two groups.

3. Static Algorithms

Static algorithms divide the traffic equivalently between servers. By this approach the traffic on the servers will be disdained easily and consequently it will make the situation more imperfectly. This algorithm, which divides the traffic equally, is announced as round robin algorithm. However, there were lots of problems appeared in this algorithm. Therefore, weighted round robin was defined to improve the critical challenges associated with round robin. In this algorithm each servers have been assigned a weight and according to the highest weight they received more connections. In the situation that all the weights are equal, servers will receive balanced traffic [2].

4. Dynamic Algorithms

Dynamic algorithms designated proper weights on servers and by searching in whole network a lightest server preferred to balance the traffic. However, selecting an appropriate server needed real time communication with the networks, which will lead to extra traffic added on system.

In comparison between these two algorithms, although round robin algorithms based on simple rule, more loads conceived on servers and thus imbalanced traffic discovered as a result [2]. However, dynamic algorithm predicated on query that can be made frequently on servers, but sometimes prevailed traffic will prevent these queries to be answered, and correspondingly more added overhead can be distinguished on network.

5. Need Of Parallel Processing In Cloud Computing

Parallel processing can be achieved on cloud servers with the help of Load balancing. Load Balancing distributes workloads across two or more servers and other resources to maximize throughput, minimize response time and avoid overload. Load balancers allow us to quickly load balance multiple cloud servers for optimal resource utilization.

Load balancer ensures that none of the server is overloaded. It increases availability and in turns, ensures business continuity.

Load balancing, routing, data partitioning are the key factors in achieving parallel processing. Routing is important to forward incoming request to computing resources. Data partitioning is useful for dividing a particular analysis problem into workable chunks of data.

6. Cloud Partitioning

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 Figure 1.

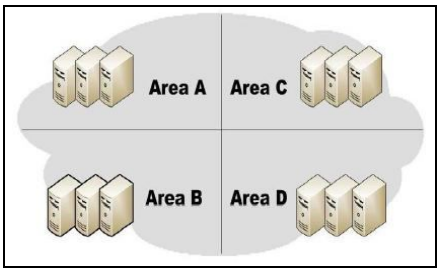

Figure 1: Typical cloud partitioning

The load balancing strategy is based on cloud partitioning concept. When job arrives at system, main controller decides which partition should receive the job. Each partition has partition balancer. When request come to partition load balancer, it decides how to assign the jobs to nodes. If cloud partition is overloaded, the job is transferred to another cloud partition. The whole process is shown in Figure 3.

7. Main Controller and Balancers

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.

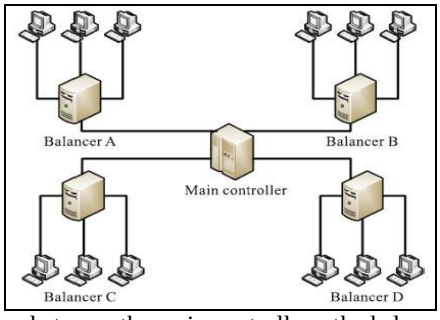

Figure 2: Relationships between the main controllers, the balancers, and the nodes[4].

8. Cloud Partition Status And Job Assignment

Cloud partition status can be divided into three types:

- Idle: When percentage of idle node is more than α, change to idle status.

- Normal: When percentage of idle node is more than β, change to normal status.
- Overload: When percentage of idle node is more than γ, change to overloaded status.

The parameters α, β, γ is set by the partition balancer.

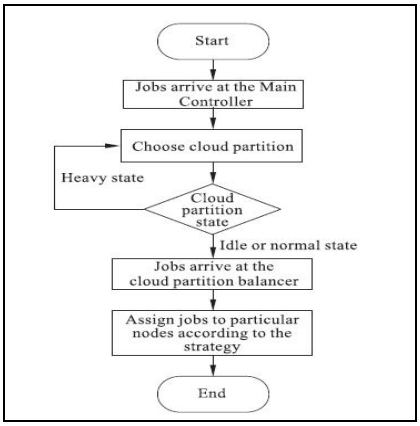

Figure 3: Job assignment strategy

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 main controller, it queries the cloud partition. If the partition's status is idle or normal, the job handled locally. If it is overloaded, the job is transferred to another cloud partition.
Following is the algorithm to search a partition based on partition status.

Algorithm 1 Best Partition Searching

```
begin
  while job do
    searchBestPartition (job);
    if partitionState == idle || partitionState == normal then
      Send Job to Partition;
    else
      search for another Partition;
    end if
  end while
end
```

Figure 4: Partition selection algorithm

9. Job Assignment To The Nodes In Cloud Partition

Job assignment is done considering cloud partition status. The partition balancer gathers load information from every node to evaluate cloud partition status.
It is required to calculate load degree of every node to determine cloud partition status. Load degree associated with the node is a combination of static and dynamic parameters. Static parameters includes number of CPU's, CPU processing speed, memory size etc. Dynamic parameters include memory utilization ratio, CPU utilization ratio, network bandwidth etc. Load degree is computed as[4]:
1. We define a load parameter set considering static and dynamic parameters. Set F consists of total number of either static or dynamic parameters.
F= {F1,F2,F3….,Fm} where m = total number of parameters. Using this set F, we calculate load degree.

$$Load\_Degree(N)=\sum_{i=1}^{m}\alpha iFi$$

Where N= Current node
α = Weight of a job. This may differ for different kinds of job.
2. Define benchmarks.
Calculate Load $degree_{avg}$ from load degree statistics.

$$Load\_degree_{avg} = \sum_{i=1}^{n} Load\_degree(N_i)\ /n$$

$Load\_degree_{high}$ is calculated using $Load\_degree_{avg}$ for different kinds of situations.
3. Get nodes load status level from Load_degree.
Idle: When Load_degree (N) = 0,
There is no job being processed by this node so the status is changed 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 load degree status of each node then stored into the Load Status table which is created by partition balancer. Each balancer maintains this table and it refreshes after a fix time period T. This table is used to calculate cloud partition table. When job arrives at the cloud partition, the balancer decides execution strategy depending upon current partition status and assign jobs to nodes accordingly. Balancer changes strategy when cloud partition status changes.

10. Load Balancing Strategy Algorithms

10.1. Round Robin Algorithm
In round robin algorithm, time quantum is allocated to each node and nodes perform operation in allocated time interval. Following figure shows working of round robin algorithm.

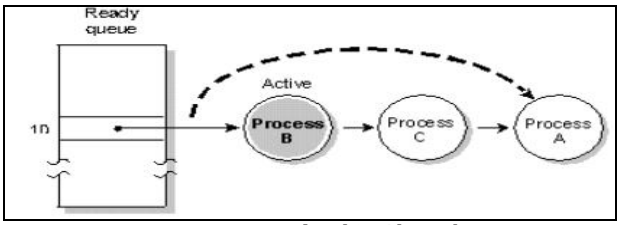

Figure 5: Round Robin Algorithm

It is a FCFS scheduling but pre-emption is added to switch between processes. It consists of a ready queue which has FIFO sequence of processes. The ready queue is treated as circular queue. New processes are added to the tail of the ready queue [3].
The CPU scheduler picks up first process from the ready queue, sets the timer interrupt and dispatches the process. If CPU burst of current process is more than time quantum allocated then after context switch, process is again added at the tail of the ready queue.
In Figure 5, processes B, C, A are in ready queue where process A just moved at the tail because of context switch and CPU executing process B.

10.2. Equally Spread Current Execution Algorithm
It is a dynamic scheduling algorithm. Here, jobs are submitted to the computing system. These submitted jobs are queued in a stack. The balancer estimates the job size and checks for the availability of the virtual machine and also the capacity of the virtual machine.

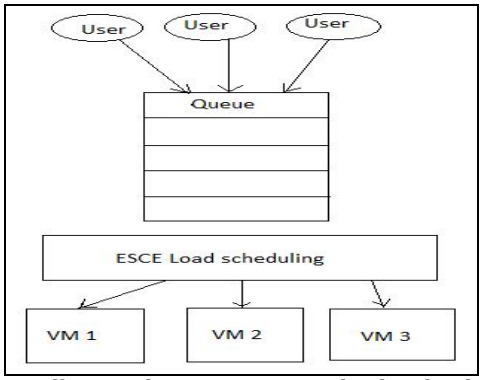

Figure 6: Equally spread current execution load to the cloud system.

Once the job size and available resource size matches balancer allocates the identified resource to the job in a queue. Because of dynamic scheduling, there is improvement in response time and processing time. The jobs are equally distributed. Hence, the computing system is load balanced and no virtual machines are underutilized. Due to this, there is reduce in the virtual machine cost and data transfer cost. Figure 6 is the representation of equally spread current execution algorithm.

11. Cloud Partition Load Balancing Strategy

11.1. 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.

There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin[5]. The Round Robin algorithm is used here for its simplicity.
The Round Robin algorithm is one of the simplest load balancing algorithms, which passes each new request to the next server in the queue. The algorithm does not record the status of each connection so it has no status information. In the regular Round Robin algorithm, every node has an equal opportunity to be chosen. However, in a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used, which called "Round Robin based on the load degree evaluation". The algorithm is still fairly simple. Before the Round Robin step, 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 T. When the balance table is refreshed, at this moment, if a job arrives at the cloud partition, it will bring the inconsistent problem. The system status will have changed but the information will still be old. This may lead to an erroneous load strategy choice and an erroneous nodes order. 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.
Thus, at each moment, one table gives the correct node locations in the queue for the improved Round Robin algorithm, while the other is being prepared with the updated information. Once the data is refreshed, the table flag is changed to "Read" and the other table's flag is changed to "Write".

11.2. Load balance strategy for normal status
Job assignment to cloud partition when it is in normal status is complex. In normal status, jobs arriving are much faster than idle status. Hence, different strategy is used for load balancing. Public cloud needs a method that can complete the jobs of all users with optimal response time.
The strategy is proposed based on a game theory for distributed systems. As an implementation to distributed system, the load balancing in cloud can be viewed as a game.
Game theory has cooperative and non cooperative games. In co-operative games, the decision is made for the teams benefit and every decision maker decides comparing notes with others. In non-cooperative game, each decision is made by decision maker for his own benefit. System then reaches to a stage where each player in the game has a chosen strategy and no player can benefit by changing his or her strategy while other players strategy remains unchanged.
Since the gird computing and cloud computing environments are also distributed system, these algorithms can also be used in grid and cloud computing environments. Cloud partition in normal load status can be viewed as non-cooperative game.

12. Conclusion
With switch mechanism in public cloud, system can achieve effective load balancing for improved performance. Load balancer with switch mechanism uses different strategies in different situations to have an optimal utilization of virtualized resources. This load balance model for public cloud ensures availability and responsiveness.

13. Future work
- Cloud division rules: Nodes in a same cluster may be far from other nodes or there will be some clusters in the same geographical area that are still apart. Thus, the framework will need different cloud division methodology.
- Setting refresh time period: Main partition and cloud balancers need to refresh status information at a fixed time period. If it is too short, the high frequency will influence the system performance. If it is long, then the information will be too old to make decisions. Hence, statistical tools and algorithms required to set a refresh time.
- Find other load balance strategy: Other load balance strategies may provide better results, so tests are needed to compare different strategies. Many tests are needed to guarantee system availability and efficiency.

14. References
1. R.Hunter, The why of cloud, http://www.garter.com/DisplayDocument?doccd=226469&ref=g_noreg,2012.
2. 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.
3. N.G.Shivaratri,P.Krueger, and M.Singhal, Load distributing for locally distributed systems,Computer,vol. 25,no.12,pp. 33-44,dec.1992.
4. Gaochao Xu, Junjie Pang, and Xiaodong Fu,A Load Balancing Model Based on Cloud Partitioning for the Public Cloud, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6449405&tag=1.
5. B.Adler, Load balancing in the cloud: Tools, tips and techniques, http://www.rightscale.com/infocenter/whitepapers/Load-Balancing-in-the-cloud.pdf,2012.