RESEARCH ARTICLE OPEN ACCESS

A Survey of Load Balancing Algorithms in Cloud Computing

Harmandeep Singh Brar[1], Vivek Thapar[2], Kunal Kishor[3],
Research Scholar1&3, Assistant Professor2,
Department of Computer Science,
Guru Nanak Dev Engineering College, Ludhiana,
Punjab-India

ABSTRACT
Cloud computing is overtaking the existing conventional methods of computation and communication over the network. The entire Internet community is often lured by a new paradigm that provides a great level of availability and security with nominal usage charges. Cloud computing, in this perspective, is an important way of disseminating information and providing computational capabilities over the network. The amount of data being stored and the services being provided are increasing at a very fast rate which, in turn, demands greater storage and computing hardware. With a huge number of requests in the form of load to the servers, load balancing becomes an important issue in cloud computing. The aim here is to distribute the load amongst the available nodes in such a way that no single node is flooded with requests, while some other node is lightly loaded. The prevalent scheduling algorithms have been addressing this issue by making use of job scheduling and resource provisioning strategies efficiently. This paper discusses the popular load balancing algorithms, along with the challenges faced.

Keywords:- Cloud computing, Virtualization, Software as a Service, Public cloud, Utility computing, Private cloud, Virtual machine, Starvation.

I. INTRODUCTION

Cloud computing has emerged as a new computing paradigm that employs a real time communication network for large scale distributed computing, with extensive use of virtualization. Cloud computing not just includes the applications that are delivered over the internet, but also the underlying hardware and the software systems that are present in the datacenters. The services being provided are termed under *Software as a Service (SaaS)*, while the system software and the hardware make up the *Cloud*. If the Cloud is made available through a pay-per-use basis, it is called a *Public Cloud*; the service being sold is *Utility Computing.* On the other hand, there are some internal datacenters of some organizations that are not made available to the public. These are referred to as the *Private Cloud* (1).

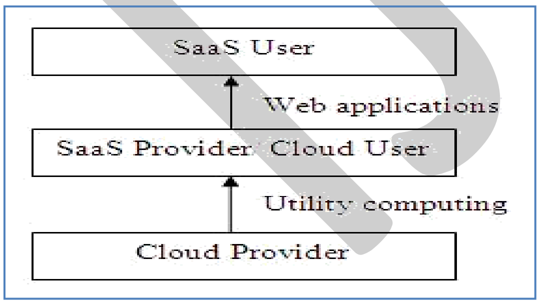

Figure 1: Cloud computing users and providers

The use of such a paradigm is advantageous to both the service provider and the end user. The service provider benefits from simplified installation and maintenance at its end as well as centralized control over versioning. On the other hand, the end user is able to access and collaborate data more easily and with more safety, anytime and anywhere over the internet (1).

The National Institute of Standards and Technology (NIST) definition describes Cloud Computing as a model that enables convenient and on-demand network access in a ubiquitous manner, to a shared pool of configurable computing resources (e.g., networks, storage, servers, services and applications) that can be rapidly provisioned and released with minimal management effort or service provider interaction (2).

NIST (2), with its responsibility for developing standards and guidelines, states the following characteristics that a cloud computing solution must exhibit:

A. ***On-demand self-service-*** Provisioning of computing capabilities like server time and storage can be performed by the consumer unilaterally, without human interaction with each of the service providers.

B. ***Broad network access-*** Network connects heterogeneous thin or thick clients like mobiles, laptops and workstations to provide capabilities through standard mechanisms.

C. ***Resource pooling-*** The provider makes use of a multi-tenant model to provide a pool of physical and virtual resources that are assigned and reassigned to serve the consumers. It also provides a sense of location independence.

D.

E. ***Rapid elasticity-*** Capabilities can be rapidly provisioned or released as per demand with complete elasticity, in some cases even automatically.

F. ***Measured service***- Cloud systems leverage a metering capability to automatically control and optimize resource use. This can be monitored, controlled and reported, thereby providing a certain level of transparency.

By satisfying the above characteristics, a cloud computing solution can aim at providing better capabilities to the users. But there are certain issues and considerations that need to taken into account. These include access control, data security and availability, network migration, data lineage and legal quagmire and transitive trust issues (3). Along with a proper provisioning technique, the internal processes of cloud computing need to be paid attention to. Load balancing is one of the most essential considerations in this regard.

II. LOAD BALANCING

Load balancing, in general, refers to the method of distribution and allocation of certain tasks amongst the available resources in an efficient way that promotes even and wise utilization. In computing, load balancing is a networking method that distributes the workload across multiple computing resources such as computers and their cluster, network links, central processing units or disk drives. The aims of load balancing remain to optimize resource use, minimize the response time, maximize throughput and avoid a single resource from being overloaded. Load balancing is often implemented in software, though it can also be performed using hardware or even the combination of software and hardware. A load balancer, as a software program, listens on the port where external clients connect to access services. Requests are forwarded by the load balancer to the backend servers, which respond to the load balancer in return. There is also a privilege of a backup load balancer in case all the servers are busy. In order to prevent a load balancer itself becoming a single point of failure, the implementation is done to provide for higher availability and replication of sessions is done.

III. LOAD BALANCING ALGORITHMS

While aiming for better load balancing and fulfillment of requests, the following goals need to be achieved:

A. ***Scalability***: The load balancing algorithm must provide scalability in terms of addition of new resources to address the everyday increasing demand of services with a greatly increasing number of users. This also demands flexibility in accommodating the change.

B. ***Better response time***: The load balancing must be implemented and executed well enough to provide the best possible response to a user.

C. ***Cost effectiveness***: A good load balancing algorithm must aim for better overall system performance with the cost being quite reasonable.

D. ***Prioritization***: The tasks must be prioritized so that the critical tasks do not have to face the problem of starvation, or if addressed, the problem of late response.

E. ***Fair node utilization***: The serving nodes must be utilized efficiently so that no single node is overwhelmed, leaving certain others totally free, or lightly loaded.

The load balancing algorithms that are currently being employed in cloud computing is described below, along with certain considerations:

A. ***Random***: The random load balancing algorithm is static (4) in nature, it being generally defined in the design or implementation of the system. It selects the node randomly by making use of a random number generator (5). The processes are then said to be handled by node n with a particular probability p (4). The allocation order of the processes is maintained for each processor. Though the algorithm works well with equally loaded processes, there may be some problem in case the loads are of different computational complexities. The issue is also that the algorithm follows no deterministic approach.

B. ***FCFS:*** The First Come First Serve algorithm (6) is a simple load balancing technique wherein each load balancer maintains a job queue in which job waits for its turn to get executed. The advantages of FCFS are a result of its being fast and simple. But it results in a poorer overall response time in case smaller tasks have to wait for longer time because of being at a later place in the queue.

C. ***Round Robin:*** The Round Robin algorithm (4) allocates the nodes to fulfill the requests in a round robin manner for a definite time slice, i.e. according to the process allocation order that is maintained locally. This serves the advantage of fast response in case of equal workload distribution amongst the processes. However, the job processing time for different processes is not the same. So, some nodes may be heavily loaded while some others may remain idle.

D. ***Weighted RR***: In this method, a ration weight is defined for each machine. According to this algorithm (5), the number of connections that each machine receives over time is proportionate to the defined ratio weight. This algorithm improved the Round Robin because whenever we define weighted assignments like “Machine 1 is able to serve 3x the load that machines 2 and 3 are able to handle”; three requests are sent by the load balancer to machine 1 for each request to the others. This algorithm works smoothly but has a problem because of the static definition of the weights in the beginning

E. ***Dynamic Round Robin:*** This algorithm (5) is quite similar to Weighted Round Robin; the difference being that the servers are continuously monitored and the weights keep on changing. This is a dynamic load balancing method. It makes use of various aspects of real-time server performance analysis, like the current number of connections per node or the fastest node response time to distribute the connections. The only issue with this algorithm is that it is rarely available in a simple load balancer, because of its dynamic nature.

F. ***Least connections:*** In this dynamic load balancing method, the load balancer records the connection number of each server, increasing the number when a new connection is dispatched to it, and decreasing the count when connection finishes or timeout happens (4). Using this algorithm, the system passes a new connection to the server that has the least number of current connections. As this method is dynamic in nature, it distributes connections based on various aspects of real-time server performance analysis, like the current number of connections per node or the fastest node response time. Least Connections method (5) works best in the environments where the servers or other equipment that are being load balanced have similar capabilities. This Application Delivery Controller method is rarely available in a simple load balancer.

G. ***Observed****:* This algorithm (5) makes use of a combination of the logic used in the Least Connections and Fastest algorithms. Using this method, servers are ranked based on a combination of the number of current connections and the response time. Servers possessing a better balance of fewest connections and fastest response time receive a greater proportion of the connections. This load balancing method is generally not available in a simple load balancer.

H. ***Equally spread current execution load:*** This algorithm (7) requires continuous monitoring of jobs which are present for execution in order to queue up the jobs and hand them over to different virtual machines. The load is distributed randomly by checking the size and thereby transferring the load to that virtual machine which is lightly loaded or can handle that task easily and takes less time, while giving maximum throughput.

I. ***Throttled load balancing algorithm:*** In this algorithm (8), whenever a client request is received, the load balancer tries to find a suitable Virtual Machine to perform the required operation. The algorithmic process starts by maintaining a list of the entire available VMs. Indexing is performed in order to speed up the lookup process. The request from a client is accepted if a match is found on the basis of size and availability of the machine. The VM is then allocated to the client. If, however VM that matches the criteria is available, then the load balancer queues up the request.

J. ***Min-Min Algorithm:*** This algorithm (4) begins by finding the minimum completion time for all tasks. Then amongst these minimum times, the minimum value among all the tasks on any resource is selected and accordingly the task is scheduled on the corresponding machine. Thereafter, the execution time of the assigned task is added to the execution times of other tasks on that machine to update the execution time on that machine. The assigned task is then removed from the list of tasks that are yet to be assigned to the machines. This procedure is followed until all the tasks are assigned on the resources. The major drawback of this method is that it can lead to starvation (9).

K. ***Max-Min Algorithm:*** Max-Min algorithm (9) is similar to the min-min algorithm except that after finding out minimum execution times, the maximum value is selected. This is the maximum time among all the tasks on any resources, according to which the task is scheduled on the corresponding machine. Thereafter, the execution time of the assigned task is added to the execution times of other tasks on that machine to update the execution time on that machine. The assigned task is then removed from the list of tasks that are yet to be assigned to the machines. This procedure is followed until all the tasks are assigned on the resources.

L. ***Token Routing:*** The algorithm (4) was designed with an aim of minimizing the system cost by moving tokens around the system. Due to communication bottleneck, agents can not possess enough information of distributing workload. The drawback of this algorithm can be removed with the help of heuristic approach of token based load balancing, thereby providing fast and efficient routing decisions. Here, agents do not need to have complete knowledge of their global state and neighbour’s working load, but make their own decisions on where to pass the token by actually building their own knowledge base. In this approach no communication overhead is generated as the knowledge base is actually derived from the previously received tokens.

IV. CONCLUSION

Cloud Computing is widely being adopted by the industry, while addressing the issues like Load Balancing, Server Consolidation, Virtual Machine Migration, Energy Management, etc. Load balancing, as a central issue, is required to evenly distribute the dynamic workload to all the available nodes in such a way that greater user satisfaction and resource utilization ratio can be achieved, with a surety of efficient and fair distribution of the computing resources. This paper discusses the concept of Cloud Computing along with the issue of load balancing. It also states some considerations for improvement in the existing load balancing algorithms. Therefore, by enhancing the existing load balancing algorithms, there is a great scope for increasing the efficiency provided by a cloud computing solution, leading to better resource utilization and greater cost-effectiveness.

REFERENCES

1. M, Armbrust, et al. ***Above the Clouds: A Berkeley View of Cloud Computing.*** **EECS Department, University of California. Berkeley : University of California, 2009. p. 23. UCB/EECS.**

2. **Mell, Peter and Grance, Timothy.** *The NIST Definition of Cloud Computing.* Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology. Gaithersburg : National Institute of Standards and Technology, 2011. Special Publication 800.

3. **Jansen, Wayne and Grance, Timothy.** *Guidelines on Security and Privacy in Public Cloud Computing.* Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology. Gaithersburg : National Institute of Standards and Technology, 2011. p. 80. Special Publication 800-144.

4. *Analysis of Load Balancing Techniques in Cloud Computing.* **K, Sidhu A and Kinger, Supriya.** 2, Fatehgarh Sahib : International Journal of Computers & Technology, April 2013, International Journal of Computers & Technology, Vol. 4. ISSN 2277-3061.

5. **MacVittie, Don.** Intro to Load Balancing for Developers – The Algorithms. *DevCentral.* [Online] 31 March 2009. https://devcentral.f5.com/articles/intro-to-load-balancing-for-developers-ndash-the-algorithms#.U5k6Xnbm4oE.

6. *A Review of the Load Balancing Techniques at Cloud Server.* **Bala, Kiran, et al.** 1, Chandigarh : International Journal of Advances in Computer Science and Communication Engineering, March 2014, International Journal of Advances in Computer Science and Communication Engineering, Vol. 2. ISSN 2347-6788.

7. *Load Balancing On Cloud Datacenters.* **S, Mahalle H, R, Kaveri P and Chavan, Vinay.** 1, s.l. : International Journal of Advanced Research in Computer Science and Software Engineering, January 2013, Vol. 3. ISSN: 2277 128X.

8. *Analytic Study Of Load Balancing Techniques Using Tool Cloud Analyst.* **Ahmed, Tanveer and Singh, Yogendra.** 2, New Delhi : International Journal of Engineering Research and Applications , 2012, Vol. 2. ISSN: 2248.

9. *Load Balanced Min-Min Algorithm or Static Meta-Task Scheduling in Grid Computing.* **T, Kokilavani and G, Amalarethinam D I.** 2, 2011, International Journal of Computer Applications, Vol. 20. 0975-8887.