ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 8(S9), 230–234 , May 2015* DOI: 10.17485/ijst/2015/v8iS9/65583

Load Balancing in Virtualized Environment - A Survey

N. Balaji[1*] and A. Umamakeshwari[2]

[1]Computer Science, School of Computing, SASTRA University, Tirumalaisamudram, Thanjavur-613401, India; jimbalaji@rocketmail.com

[2]School of Computing, SASTRA University, Tirumalaisamudram, Thanjavur-613401, India; aum@cse.sastra.edu

Abstract

Cloud computing renders umpteen number of services such as offering computing resources like processing power, storage of data and user applications. Load balancing in cloud computing is of paramount importance. As the request floods in, certain nodes might become heavily loaded. So it is essential to spread the load uniformly among the nodes. On the whole load balancing enhances the overall performance of the system and provides proper utilization of resources. In this paper, a review of prevailing load balancing approaches in cloud environment is presented and evaluated against various metrics such as throughput, fault tolerance, scalability, migration time and resource utilization.

Keywords: Cloud Computing, Dynamic Load Algorithms, Load Balancing Algorithms, Static Load Algorithms, Virtualization

1. Introduction

Cloud computing[1] is a framework that provides access to a variety of computing resources. In cloud computing, distinct number of applications and files are accessible to the users via internet. It is possible for the consumers to use the required services on demand rather than purchasing the entire software. The consumers can pay for the duration and the type of resources they are in need of. Some of the services provided by the cloud include[2]. Software as a Service (SaaS): The software is placed on the distant servers which is made accessible to the users across the network. In this case the consumer has to pay for the use of the software. One significant advantage of SaaS is that the service provider renders authentic applications at a cheaper price. Platform as a Service (PaaS):PaaS does not require the customers to maintain the hardware and software required to manage the applications at their end. PaaS offers services such as storage, application design and testing. But PaaS does not support portability amidst service providers. Infrastructure as a service (IaaS):Unlike SaaS and PasS, IaaS does not offer applications to the users. Instead it provides hardware so that the users can customize according to their requirements. IaaS renders virtualization as a service. It provides servers that can be purchased by the user and the user is charged depending upon the amount of usage. It is also referred to as hardware as a service. Cloud can be deployed in various forms such as:

- **Private cloud**: It is designed for use by a particular organization. Responsible for storing the data of a particular organization. Access is provided only to the members of the organization. Highly secure and maintenance is easier.
- **Public cloud:** Cloud service provider offers service to the public on demand. The service provider allows access to the users through the network. Since the cloud is made publicly available, security is a major threat. Works on 'Pay and Use' basis.
- **Hybrid cloud:** It includes the characteristics of both public cloud and private cloud.

*Author for correspondence

Load balancing is a strategy in which the complete load of the system is spread among the multiple nodes such that no node remains idle[3]. Load balancing improves the throughput of the system, enables effective usage of resources and reduces the response time. It is executed either by means of hardware or software. The core objectives of load balancing algorithms are to enhance the performance, supports system stability and backup facilities during system crash.

The rest of the article is organized as section 2 that covers the static algorithms, section 3 that explores dynamic load balancing algorithms, section 4 that explains dynamic load balancing strategies and a final tabular column that shows the summary of all methodologies on the basis of evaluation metrics.

2. Static Algorithms

The assignment of jobs to processors is done before the program execution. It does not take into account the current state of the system. Preliminary knowledge of the system is essential. Static load balancing algorithms4 are not suited for large distributed systems like cloud environments. Such algorithms are less reliable because it is not feasible to transfer load from one machine to another in case of system failure at run time. Preemption of tasks is not possible. Some of the static load balancing algorithms:

2.1 Round Robin Algorithm (RR)

This algorithm[5] is uncomplicated of all and is designed on the concept of time slices or quantum. Time is divided into many slices and each node is given to a particular time interval and the operations will be carried out in that particular slice. Resources are given to the client on the basis of time quantum. Time slices play an important role. But the number of context switches is high, leads to load imbalance (certain nodes are heavily loaded whereas the rest are lightly loaded) and deciding the quantum size might be difficult.

2.2 Opportunistic Load Balancing Algorithm (OLB)

As the job arrives, it gets scheduled in a random manner. It does not take into account the execution time of the jobs on those resources but results in effective load balancing[6]. The drawback is that it leads to poor make-span.

2.3 Min-Min Algorithm

Initially, minimum execution time for all jobs will be computed. Job with least execution time is chosen and assigned to the node that produces minimum completion time for all the jobs. Until all the jobs are scheduled, the process is repeated. Since small jobs are scheduled first only they can access the node that performs faster.

2.4 Max-Min Algorithm

It is analogous to min-min algorithm but the job with highest completion time is chosen and assigned to the machine that produces minimum completion time for all the jobs. Since larger jobs are executed first, there will be an increase in make span and will cause the smaller jobs to wait for long.

2.5 Randomized Algorithm

In this algorithm let consider a node n has a probability p. The assignment of process to processor is performed locally. This technique works efficiently when the processors are equally loaded. This does not work well if the loads are of varying complexities.

2.6 Load Balanced Min-Min (LBMM)

This algorithm has two phases[7]: In the first phase Min-Min algorithm is used and in the second phase the resources that are overloaded are selected and are rescheduled to the resources that are under loaded. The overloaded resources are identified by selecting the resource that is having the highest make span as per the schedule produced by Min-Min. Then from the overloaded resource R_j it selects the job with minimum execution time. Find the maximum completion time for the job with respect to all the available resources. If make span produced by the Min-Min is greater than maximum completion time of that job then reschedule the job in that resource R_k and update the ready time of both resources R_j and R_k, else find the job with the next maximum completion time and resource R_k producing it. Repeat the process again and again until all the resources are considered and all the jobs given to it are rescheduled. Thus all the resources that are idle and under loaded are utilized. This algorithm ensures efficient load balancing.

2.7 Drawbacks of Static Algorithms

The distribution of workload for certain systems cannot be estimated before the execution of the program. Attributes of computing resources are all known beforehand and remain unaltered. This may not relate to distributed environments like cloud. It causes load imbalance in some nodes thus increasing the load balancing time. It does not work for systems in which there is constant load variation, Poor resource utilization.

3. Dynamic Algorithms

In dynamic algorithms[8] the workload is spread among the nodes at runtime. The current state of the system is taken into account which improves the system performance by the migration of jobs from over-loaded to under-loaded nodes. Preemption of tasks is possible. Dynamic algorithms are adaptive in nature. Some of the dynamic algorithms:

3.1 Honey Bee Inspired Load Balancing Algorithm (HBB-LB)

This algorithm mocks the food foraging behavior of honey bees. There is a set of bees known as scout bees; they help the forager bees to discover the origin of the food. Once the scout bees discover the origin, they get back to the beehive and perform a tremble dance to specify the quality or amount of the food and it also tells about the distance between the beehive and the origin. Then the forager bees go to that specified place to gather honey. These forager bees then return to beehive and perform a tremble dance to specify the remaining amount of food in the origin. Likewise when a virtual machine is overloaded the tasks that are removed are considered as bees. When the task arrives at the under loaded virtual machine it updates the number of different priority tasks and the load of it to all the waiting tasks in that virtual machine. When a higher priority task comes to a virtual machine it should check whether that virtual machine has minimum amount of high priority tasks so that it gets a chance to execute earlier.

Fundamentally, the honeybees act as tasks and the origin of the food act as virtual machines. Assigning tasks to a virtual machine is same as the bees searching for their food. When the honey gets exhausted at the origin, the bees try to discover a new origin. This phenomenon is followed by the virtual machines i.e., when the virtual machines are overloaded the tasks are moved to under loaded virtual machines. These tasks check the status of the virtual machines and give a notification to all the waiting tasks similar to the honeybees which perform a tremble dance to notify the other bees. By doing so, there is a clear plan to determine which task can be loaded into the virtual machine. The major disadvantage is that the migration time is high.

3.2 Heterogeneous Earliest Finish Time Algorithm (HEFT)

In this algorithm the problem is illustrated in form of a Directed Acyclic Graph (DAG) in which the nodes indicate the required computation and the edges indicate the communication between the tasks. For every single node in the DAG, weight is set proportionate to computation cost and weights for edges are set corresponding to the communication cost between the nodes. To start with, the subtasks will be scheduled depending on a value computed using the ranking function which is dependent on the weights allotted to the subtasks and the communication between them. Initially, a weight is assigned to each node and the edge of the graph based on the average computation time. Then the graph is spanned upwards and a rank value is assigned to each node. The ranking function is calculated by the summation of the maximum weight of the successor nodes, weight of the edge and the rank of the successor node. Finally, the tasks are arranged in the descending order depending on the rank value. It increases the total finish time and minimizes the performance.

3.3 Biased Random Sampling

In this algorithm[9], a virtual graph is created. Each vertex in the graph represents a machine and the connection between the machines constitutes the load. Free resource in every node is represented by an in-degree. Incoming edge gets deleted when a job is executed which indicates depletion in free resources. Once the job is completed, an incoming edge will be created that represents the availability of resources. Suitable node for load allocation is chosen based on computing power or machine that is lightly loaded. Each task is given a threshold value. A walk is described as the traversal from one machine to another until it reaches the destination

When a request is received, a node is selected at random. Threshold value is then compared with current walk length. The task gets executed only if the walk length is

greater than equal to the threshold value. There will be a drop in the performance if there is an increase in the number of servers.

3.4 Carton

This technique combines the services of Load Balancing (LB) and Distributed Rate Limiting (DRL)[4]. Load Balancing ensures that all the servers are equally loaded such that the cost factor is minimized. Distributed Rate Limiting ensures the proper distribution of available resources in order to maintain a reasonable resource allocation. CARTON mechanism is well suited for cloud environments.

3.5 Join-Idle Queue

This algorithm is generally designed for large-scale environments. It makes use of dispersed dispatchers by first balancing the processors that are idle across dispatchers. Then jobs are assigned to processors in order to reduce the queue length. The major drawback is that the algorithm is not scalable. Y. Lua et al proposed this algorithm for effectively scalable web services. This algorithm reduces the load of the system, experiences no overhead during job arrivals. Reliability and scalability is of utmost importance in today's dynamic web-content services and therefore this algorithm is not recommended.

3.6 Throttled Load Balancer

This algorithm totally depends on the virtual machine. In this technique, the client initially requests the load balancer to discover an appropriate virtual machine that can easily access the load and process the operations that are specified by the client. The cloud environment may have many virtual machine instances. Depending on the type of requests they can manage, the machines get grouped. Whenever a request is sent by the user, the load balancer checks for that group and allots the task to the under loaded machine of that group.

3.7 Equally Spread Current Execution Algorithm

An index table is maintained by the load balancer for all virtual machines and the number of requests currently assigned to the VM. If the request from the data centre needs to be allotted to a VM it initially checks the index table for a lightly loaded VM. It then returns the VM id to the data centre controller. The data centre then communicates the request to the VM that has been identified. The allocation count is then increased by the data centre for that particular VM. Finally, when the task is completed by the VM, a request in then sent to the data centre which is then notified by the load balancer. Then the load balancer decreases the allocation count in the index table for that VM. Because of scanning the queue again and again the algorithm has high computational overhead.

3.8 Load Balancing Virtual Strategy (LBVS)

It yields a huge data storage model and service model for storage depending on cloud storage[4]. A three-layered structure is used to achieve storage virtualization and two load balancing modules are used to achieve load balancing. By means of replica balancing, it enhances the effectiveness of concurrent access and it also reduces the response time and helps to recover data in case of failures. The advantage of this technique is it is flexible, robust and improves the usage rate of the storage resources.

4. Dynamic Load Balancing Strategies

1.Location Policy: The policy that is used by a heavily loaded node to transfer load is referred to as location policy.
2.Transfer Policy: The policy that is used to choose a task to be transferred to a remote node is referred to as transfer policy.
3.Selection Policy: The policy that is used for determining the processors that participate in load balancing is referred to as selection policy.
4.Information Policy: The policy that is required for collecting information with which the decision for load balancing is made.

5. Metrics of Load Balancing

In cloud environment, load balancing is essential to spread the load uniformly among the available systems. Load balancing improves the throughput of the system, enables effective usage of resources and reduces the response time. It ensures user satisfaction and allocation of resources. In this paper, we have taken into account several metrics such as:

Table 1. Comparison of prevailing load balancing algorithms

Metrics	Min-Min	Max-Min	HBB-LB	Equally Spread	Join Idle Queue	Biased Random
Throughput	Yes	Yes	Yes	No	No	No
Fault Tolerance	No	No	No	No	No	No
Scalability	No	No	No	Yes	No	No
Migration Time	No	No	No	Yes	No	No
Performance	Yes	Yes	Yes	Yes	Yes	Yes
Resource Utilization	Yes	Yes	Yes	Yes	No	Yes

1.**Throughput** determines the number of tasks executed.
2.Migration Time is the time taken to migrate the tasks from one machine to another. In order to maximize performance, migration time should be minimum.
3.Fault Tolerance is the capability of the algorithm to ensure efficient balancing even in case of failure.
4.Resource Utilization is used to determine the number of resources utilized.
5.Scalability is used to verify whether the algorithm is scalable according to the specifications.
6.Performance is used to verify the effectiveness of the system at a reasonable cost.
Based on the above metrics, a comparison on the various load balancing methods is shown below in Table 1.

6. Conclusion

Load balancing is one of the major challenges in cloud computing. Load balancing is a strategy in which the complete load of the system is spread among the multiple nodes such that no node remains idle. Load balancing improves the throughput of the system, enables effective usage of resources and reduces the response time. In this paper we have made an analysis of the existing load balancing algorithms in cloud computing. According to the user's requirements, a particular algorithm can be chosen. Analysis on each load balancing algorithm can be performed to determine the efficiency of each method as future work.

7. References

1. Shetty JP, Kumar HK. Cloud computing: an exploratory study on adoption among SME clusters in Bangalore and Mysore. Indian Journal of Science and Technology. 2015; 8(Supplementary 4):169–75.
2. Sidhu AK, Kinger S. Analysis of load balancing techniques in cloud computing. International Journal of Computers & Technology. 2013; 4(2):737–41.
3. Padhy RP. Load balancing in cloud computing systems. Rourkela: National Institute of Technology; 2011.
4. Tong R, Zhu X. A load balancing strategy based on the combination of static and dynamic. DBTA; 2010.
5. Samal P, Mishra P. Analysis of variants in round robin algorithms for load balancing in cloud computing. IJCSIT. 2013; 4(3):416–9.
6. Narang A, Laxmi V. Comparision of a new approach of balancing the load in cloud environment with the existing techniques. International Journal of Engineering Sciences & Research Technology. 2014; 3(10):451–8.
7. Moharana SS, Ramesh RD, Powar D. Analysis of load balancers in cloud computing. Int J Comput Sci Eng. 2013; 2(2):101–8.
8. Feng Y, Li D, Wu H, Zhang Y. A dynamic load balancing algorithm based on distributed database system. International Conference on High-Performance Computing in the Asia-Pacific Region; IEEE Computer Society; 2000.
9. Rahmeh OA, Johnson P, Taleb-Bendiab A. A dynamic biased random sampling scheme for scalable and reliable grid networks. INFOCOMP Journal of Computer Science. 2008; 7(4):1–10.