Effective load balancing in cloud computing

Zeinab Goudarzi*, Ahmad Faraahi

Department of Computer Engineering and Information Technology, Payame Noor University, PO BOX 19395-3697 Tehran, Iran

Email address:
z1990_good@yahoo.com (Z. Goudarzi), afaraahi@pnu.ac.ir (A. Faraahi)

To cite this article:

Abstract: Internet, from its beginning so far, has undergone a lot of changes which some of them has changed human’s lifestyle in recent decades. One of the latest changes in the functionality of the Internet has been the introduction of Cloud Computing. Cloud Computing is a new internet service, which involves virtualization, distributed computing, networking, software etc. This technology is becoming popular to provide various services to users. Naturally, any changes and new concepts in the world of technology have its own problems and complexities. Using Cloud Computing is no exception and has many challenges facing the authorities in this area such as load balancing, security, reliability, ownership, data backup and data portability. Load balancing is one of the essential factors to enhance the working performance of the Cloud service provider by shifting of workload among the processors. Proper load balancing aids in minimizing resource consumption, implementing fail-over, enabling scalability, avoiding bottlenecks and over-provisioning etc. Given the importance of the process of load balancing in Cloud Computing, the aim of this paper is to review the process and to compare techniques in this field.

Keywords: Cloud Computing, Load Balancing, Metrics

1. Introduction

Cloud Computing is a new technology, which provides on-demand network access to a shared pool of computing resources. Cloud is a pay-go model where the consumers pay for the resources utilized instantly, which necessitates having highly available resources to service the requests on demand [1]. It was intended to enable computing across widespread and diverse resources, rather than on local machines or at remote server farms [2].

Services in a Cloud are of 3 types as given in [3]: (1) Software as a Service (SaaS): with SaaS, the users don’t need to install the software on their machines. SaaS provider maintains and manages software, supplies the hardware facilities and the users can use software directly from the cloud. For small business, SaaS is the best way to use advanced technology. (2) Platform as a Service (PaaS): PaaS provides a platform on which users can directly develop and deploy their own applications and transfer to other customers through their server and internet. Microsoft Azure is one example of PaaS. (3) Infrastructure as a Service (IaaS): IaaS offers the hardware as a service to the customers and provides an environment for deploying, running and managing virtual machines and storage.

Cloud Computing by using the resources, information, software and shared equipment provides a client’s service within a specific time. Of course, regarding the time which is spent on the Internet, the whole Internet can be considered as a Cloud. Operating and capital costs can be reduced by using Cloud Computing [4]. Due to the spread of Cloud Computing in recent years, from the perspective of market presence, understanding the effect of load balancing in the Cloud is important. Cloud Computing Platform, is a fully automated service platform that allows users to buy, create distance, dynamic scalability, and management of the system [5].

The load measuring is mechanized so as to avoid disruption in delivery of a service when one or more components of the system are in trouble. In this case, the system components are constantly monitored, and when a component fails to respond, load balancing comes up and do not sent traffic on it. Often, problems can be minimized with proper load balancing that not only reduce costs and create green computing, but keeps the pressure minimum on the unique circuits that makes them potentially longer life [6].

Load balancing in Cloud Computing systems is really a challenge now. It is a mechanism that distributes the dynamic local workload evenly across all the nodes in the whole Cloud to avoid a condition in which some of the nodes are over
loaded while some others are idle or under loaded. It helps to achieve a high user satisfaction and resource utility of the system. It also ensures that every computing resource is distributed efficiently and fairly [7].

The load considered here can be in terms of CPU load, amount of memory used, delay or network load [8]. 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 [9]. Load balancing algorithms are placed in three categories on the basis of who starts the process of load balancing: sender initiated, receiver initiated, symmetric (combination of sender initiated, receiver initiated types) and are placed in two categories based on the current state of the system: static and dynamic [6].

With load balancing, the load can be balanced by active transferring of local workload from a machine to the machine in the remote node or machine that is used less [10]. Load balancing is also required in the Clouds to meet the Green Computing. In this way, it can reduce the amount of energy by avoiding excessive interaction or virtual machines considering the workload, and by reducing the energy consumption, carbon emissions are reduced and, we thus achieve the Green Computing. As given in [7], [11] and [12] the goals of load balancing are:

i. Have a backup plan to build a fault tolerant system
ii. Substantial improvement in performance
iii. Stability maintain the system
iv. Accommodate future modification in the system

When a given workload is applied on any cluster’s node, this given load can be efficiently executed if the available resources are efficiently used. So that, there must be a mechanism for choosing the nodes that have these resources. Scheduling is a component or a mechanism, which is responsible for the selection of a cluster node, to which a particular process will be placed. This mechanism will investigate the load balancing state. Hence, scheduling needs algorithms to solve such problems [9].

2. Important Resources and Metrics of Load Balancing

Important resources in load balancing as discussed by Hamo and Saeed [13] include:

i. Computer processor time: it is the most important resource in operating system. When distributed system is used this resource need to be balanced.

ii. Computer memory: another important resource in the computer is memory. When these computers are connected across the grid, memory resources, need to be balanced.

iii. Computer I/O: I/O resources, which is depends on the effective usage of storage, in addition to that of CPU and memory, need to be balanced.

In Cloud Computing, load balancing is a necessary mechanism to increase the service level agreement (SLA) and better uses of the resources [8].

Various metrics in load balancing Techniques in Cloud Computing are discussed in Table 1.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>It is used to calculate the no. of tasks whose execution has been completed. This metric should be high to achieve good load balancing and improve the performance of the system.</td>
</tr>
<tr>
<td>Overhead</td>
<td>It determines the amount of overhead involved while implementing a load balancing algorithm. It is composed of overhead due to movement of tasks, inter-processor and inter-process communication. This metric should be minimized so that a load balancing technique can work efficiently.</td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>Is the ability of an algorithm to perform uniform load balancing in case of link failure. The load balancing should be a good fault-tolerant technique, in order to achieve a high user satisfaction and improve the performance of the system.</td>
</tr>
<tr>
<td>Response Time</td>
<td>It is the amount of time taken to respond by a particular load balancing algorithm in a distributed system. It should be minimized.</td>
</tr>
<tr>
<td>Resource Utilization</td>
<td>It is used to check the utilization of resources. It should be optimized for an efficient load balancing.</td>
</tr>
<tr>
<td>Scalability</td>
<td>It is the ability of an algorithm to perform load balancing for a system with any finite number of nodes. This metric should be improved for load balancing.</td>
</tr>
<tr>
<td>Performance</td>
<td>It is used to check the efficiency of the system. This has to be improved at a reasonable cost, e.g., reduce task response time while keeping acceptable delays.</td>
</tr>
<tr>
<td>Migration Time</td>
<td>Is the time to migrate the jobs or resources from one node to other. It should be minimized in order to enhance the performance of the system.</td>
</tr>
<tr>
<td>Energy Consumption</td>
<td>Determines the energy consumption of all the resources in the system. Load balancing helps in avoiding overheating by balancing the workload across all the nodes of a Cloud, hence reducing energy consumption.</td>
</tr>
</tbody>
</table>
3. Load Balancing Algorithms

3.1. Static Load Balancing

Static load balancing algorithms allocate the tasks to a node only based on the node’s ability for processing the new request. This process is based solely on prior knowledge of the properties of nodes, which can include the node processing power, memory, and storage capacity and so on. Static load balancing algorithms usually do not consider the changes that occur at run time for these attributes and cannot adapt to changes in runtime. Their goal is to minimize execution time and limit the communication overhead and delay [10].

In fact, static load balancing allocates a work that enters the system to a processor or a fixed node, and each time the system is restarted, the same processor will be responsible for executing the work. It is possible that the allocated task to the same processor does not work, but allocating the new coming tasks is in a fixed order or model. The static load balancing is either Deterministic or Probabilistic. In the Deterministic load balancing, tasks are connected to the smallest queue of work stations since routing decisions are based on the state of the system. In Probabilistic load balancing, tasks are randomly sent to the stations with equal probability [13].

3.2. Dynamic Load Balancing

Dynamic load balancing algorithms take into account the various features of nodes, capabilities and network bandwidth. Most of these algorithms rely on a combination of information about the nodes in the cloud which have already been collected and properties of collection time. The algorithm may actively re-allocate the tasks to the nodes based on the information gathered just after allocating the tasks to the nodes. So they need constant monitoring over the nodes and tasks stream, and are usually more difficult to implement. But on the other hand, they enjoy a higher accuracy and can produce a more efficient load balancing results [15].

An important advantage to dynamic load balancing is that load balancing decisions are based on the current state of the system which contribute to better overall performance of system with dynamic migration of loads [10]. A dynamic strategy is usually performed several times and may re-allocate a scheduled task to a new node based on the dynamic state of the system environment [13].

Dynamic load balancing algorithms fall into two categories: distributed and Non-distributed. In distributed type, the task of balancing the load is for all the nodes in the system. Interaction between nodes to achieve load balancing can be in two forms: cooperation and non-cooperative. In the first form, the nodes cooperate with each other to reach a common goal such as improving the time of the total response. And in the second form, each node works independently toward a local target like improving the response time of a local work [9].

However, in Non-distributed form, one node or a group of nodes are doing load balancing task. The Non-distributed dynamic load balancing algorithms can be classified into two types of Centralized and Semi-Distributed. In Centralized type, the load balancing algorithm is performed only by one node of system, so called central node. This node solely has the duty of load balancing of the entire system. The other nodes only have interactions with the central node. In the type of Semi-Distributed, the nodes of system have been divided into clusters in which load balancing of each cluster is centralized. By using appropriate techniques for each cluster, a node can be selected to look after the load balancing in cluster. Hence, the total load balancing of system is done by the central nodes of each cluster [4].

The distributed dynamic load balancing algorithms produce more messages than their Non-Distributed counterparts since each node within the system needs to communicate with any other node. The advantage of this method is lack of bottleneck in the system, which in turn affect the system performance partly. The distributed dynamic load balancing can put an enormous pressure on a system in which each node needs to exchange situational information with any other node. This method is more effective when most of the nodes work independently with little interaction with other nodes. Since the centralized dynamic load balancing receive fewer messages, the total number of interactions within the system is diminished as compared to the semi distributed type. However, the centralized algorithms can create bottleneck on the central node, and also the load balancing pattern become useless when the central node crashes. So this algorithm is more suitable for small-sized networks [10].

4. Load Balancing Techniques

4.1. Honeybee Foraging Algorithm

The main idea behind the algorithm is derived from the behavior of honey bees for finding and reaping food. There is a class of bees called the forager bees which forage for food sources, upon finding one, they come back to the beehive to advertise this using a dance called waggle dance. The display of this dance, gives the idea of the quality or quantity of food and also its distance from the beehive. Scout bees then follow the foragers to the location of food and then began to reap it. Then they return to the beehive and do a waggle dance, which gives an idea of how much food is left and hence results in more exploitation or abandonment of the food source.

In case of load balancing, as the webservers demand increases or decreases, the services are assigned dynamically to regulate the changing demands of the user. The servers are grouped under virtual servers (VS), each VS having its own virtual service queues. Each server processing a request from its queue calculates a profit or reward, which is analogous to the quality that the bees show in their waggle dance. One measure of this reward can be the amount of time that the CPU spends on the processing of a request. The dance floor in case of honey bees is analogous to an advert board here. This board is also used to advertise the profit of the entire colony. Each of the servers takes the role of either a forager or a scout. The server after processing a request can post their profit on the advert boards with a probability of pr. A server can choose a
queue of a VS by a probability of px showing forage/explore behavior, or it can check for advertisements (see dance) and serve it, thus showing scout behavior. A server serving a request, calculates its profit and compare it with the colony profit and then sets its px. If this profit was high, then the server stays at the current virtual server; posting an advertisement for it by probability pr. If it was low, then the server returns to the forage or scout behavior [4].

4.2. Based Random Sampling

Biased Random Sampling is a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each in degree directed to the free resources of the server each server is symbolized as a node in the graph, with each in degree directed to the free resources of the server. Regarding job execution and completion,

i. Whenever a node does or executes a job, it deletes an incoming edge, which indicates reduction in the availability of free resource.

ii. After completion of a job, the node creates an incoming edge, which indicates an increase in the availability of free resource.

The addition and deletion of processes is done by the process of random sampling. The walk starts at any one node and at every step a neighbor is chosen randomly. The last node is selected for allocation for load. Alternatively, another method can be used for selection of a node for load allocation, that being selecting a node based on certain criteria like computing efficiency, etc. Yet another method can be selecting that node for load allocation which is under loaded i.e. having highest in degree. If b is the walk length, then, as b increases, the efficiency of load allocation increases. We define a threshold value of b, which is generally equal to log n experimentally. A node upon receiving a job, will execute it only if its current walk length is equal to or greater than the threshold value. Else, the walk length of the job under consideration is incremented and another neighbor node is selected randomly. When, a job is executed by a node then in the graph, an incoming edge of that node is deleted. After completion of the job, an edge is created from the node initiating the load allocation process to the node which was executing the job. Finally what we get is a directed graph. The load balancing scheme used here is fully decentralized, thus making it apt for large network systems like that in a Cloud [4].

4.3. Active Clustering

Active Clustering works on the principle of grouping similar nodes together and working on these groups. The process involved is:

i. A node initiates the process and selects another node called the matchmaker node from its neighbors satisfying the criteria that it should be of a different type than the former one.

ii. The so called matchmaker node then forms a connection between neighbors of it which is of the same type as the initial node.

iii. The matchmaker node then detaches the connection between itself and the initial node.

The above set of processes is followed iteratively [4].

The performance of the system is enhanced with high resources thereby in-creasing the throughput by using these resources effectively. It is degraded with an increase in system diversity. This algorithm optimizes job assignment by connecting similar services by local re-wiring and performs better with high resources.

4.4. Carton

This technique is used to equally distribute the load balancing tasks among different servers. Therefore, the corresponding costs can be reduced and limited distribution rates are used to ensure a fair allocation of resources [15].

4.5. Event-Driven

A load balancing algorithm has suggested an event-driven method for the real time Massively Multiple Player Online Games. This algorithm after receiving the incoming capacity as input analyzes its own components in the context of resources and the overall state of play settlement. As a result, the activities of load balancing produce the game session [16].

4.6. Server-Based Load Balancing for Distributed Internet Services

A new service-based load balancing policy for Web servers that are distributed throughout the world has been suggested. This policy helps to reduce the time of service through limiting the number of deviances of a request to the closest server of remote path without making them the overhead. A middleware has been described for the implementation of this protocol which uses a discovery method for helping the Web servers to bear the load [17].

4.7. Black-Box and Gray-Box Strategies

Wood et. al, [18] adopt Black-Box and Gray-Box strategies for Virtual Machine (VM) migration in large data centers. Black-Box technique has been designed to monitor the system resource usage, detect hotspots and initiate the necessary migrations, that can make these decisions by simply observing each Virtual Machine from the outside and without any knowledge of the application resident within each Virtual Machine, and Gray-Box technique has been designed to access a small amount of OS level statistics to better inform the migration algorithm. The authors have designed the Sandpiper system to support techniques. Sandpiper imposes negligible overheads and that Gray-Box statistics enable Sandpiper to make better migration decisions when alleviating memory hotspots.
4.8. Min-Min

This algorithm works by considering the execution and completion time of each task and begins with a set of all unassigned tasks. First, minimum execution time for all tasks is found. Then, the task with the least execution time among all the tasks on any resources is selected. The algorithm assigns the task to the resource that produces the minimum completion time. Until all tasks are scheduled the same procedure is repeated [19].

4.9. Max-Min

This algorithm is almost same as the Min-Min algorithm but, it schedules larger tasks first of all i.e. After finding out minimum execution times, the maximum value is selected which is the maximum time among all the tasks on any resources. Until all tasks are scheduled the same procedure is repeated [19].

4.10. The Message Oriented Model

Clusters offer the opportunity of using the distributed applied programs by different computers on the networks. This is associated with the introduced clusters in the network performance. If the total load in distribution network is loaded by a computer, it makes the network slower. To prevent this situation, the resource management can be used as software metrics for traffic distribution between stations to keep the network performance in a high level. The Web services are mainly used in the "quick online message programs". This technology is for the real time communication between different sides; however, the availability of functional program is important. One model has been presented that uses XMPP for load balancing. The clients of XMPP send the prepared information to the prepared server of XMPP and XML flows contain the details of the prepared information of the customers generated by these servers. Using a load balancing on top of a XMPP server allows incoming requests from public services to be prioritized and applied [20].

4.11. OLB + LBMM

This method uses a combination of two algorithms for a better implementation and maintaining the load balancing of system. The OLB scheduling algorithm, keeps each node in the mode of work to reach the goal of load balancing, and LBMM scheduling algorithm is used to reduce the time each task is run on a node which leads to reduction in total run time. The usage environment of this algorithm is three dimensional cloud computing networks [21]. This combination algorithm makes better use of the resources, increases productivity and provides better results compared to honeybees exploration, random sampling and active clustering.

4.12. Fuzzy Logic

The paper in [22] proposes the novel load equalization technique using Fuzzy Logic in Cloud Computing, within which load equalization could be a core and difficult issue in Cloud Computing. In this work, the authors have designed a new load balancing algorithm based on Round Robin in Virtual Machine environment of Cloud Computing in order to achieve better response time and processing time. The load balancing algorithm is done before it reaches the processing servers the job is scheduled based on various parameters like processor speed and assigned load of Virtual Machine and etc. It maintains the information in each VM and numbers of request currently allocated to VM of the system. It identify the least loaded machine, when a request come to allocate and it identified the first one if there are more than one least loaded machine. In this architecture, the fuzzifier performs the fuzzification process that converts two types of input data like processor speed and assigned load of Virtual Machine and one output like balanced load which are needed in the inference system. The design also considers the processor speed and load in Virtual Machine as two input parameters to make the better value to balance the load in Cloud using Fuzzy Logic. These parameters are taking as inputs to the fuzzifier, which are used to measure the balanced load as the output. Two parameters named as the processor speed and assigned load of virtual Machine of the system are jointly used to evaluate the balanced load on data centers of Cloud Computing environment through Fuzzy Logic.

4.13. Join-Idle-Queue Algorithm

This algorithm suggests a load balancing algorithm for active scalability of Web services. This algorithm provides load balancing in large-scale with distributors of the dispatchers. First, the load balancing of idle processors is done across the dispatchers for access of any idle processor to each dispatcher and then allocation of tasks to processors is performed to reduce the average queue length of each processor [23]. By removing the load balancing work of the vital routes of processing requests, this algorithm effectively reduces the system load in which no communicative overhead is occurred while entering tasks, and real response time is not raised as well.


This policy balances the load evenly in a computing cloud or distributed virtual machine and increases the overall performance of the system by about 20%, but does not take into account those systems that have a fault tolerance. This method uses information of general mode for load balancing decisions [24].

4.15. Power Aware Load Balancing Strategy

The strategy proposed in [25] is an energy conscious, power aware load balancing strategy based on adaptive migration of Virtual Machines. This strategy will be applied to Virtual Machines on Cloud, considering higher and lower thresholds for migration of Virtual Machines on the servers. If the load is greater or lower then defined upper & lower thresholds, VMs will be migrated respectively, boosting resource utilization of the Cloud data center and reducing their energy consumption.
The system models that used in this paper consist of global and local manager. The local managers, which are part of VM monitor, resides on each node and are responsible for keeping continuous observation on when to migrate a VM and utilization of the node. The end-user refers its service request along with some CPU performance parameters to a global manager which in turns intimates the VM monitor for VM Allocation. The local manager reports the global manager about the utilization check of its node. And thus, global manager keeps the check of overall utilization of the resource. To reduce number of migration they integrate minimum migration time policy.

4.16. Dynamic Load Balancing Approach

The paper in [26] proposes the load balancing method based on the remaining storage capacity. The method considers the load and the node’s remaining storage capacity of the node. The method uses a hybrid structure model using two layers of load balancing strategies. The lower layer manages the local load information. Each level uses the different balance strategy. This will not only be able to timely balance local load, the scheduler can balance the global load in the upper. This model is not a bottleneck. It spends small resources to improve overall system performance. According to the different hotspots of the data, it uses two strategies that include migrating and replicating data to reduce the load.

4.17. Efficient Load Balancing Approach

Zuhori [27] presented Round Robin Algorithm for efficient load balancing in Cloud Environment. This algorithm is Round Robin to reschedule the CPUs. Here at first consumer’s request submitted into the Service Accepter and Service Accepter search for free VMs. When it finds one it starts to serve the services to those VMs using Round Robin Algorithm. In Round Robin algorithm the time is divided into multiple slices and each node is given a particular time slice or time interval. The decision of a service acceptance or rejection is taken by the service accepter. The total work of this thesis work is done for ten servers. Here, the real Cloud environment has not used and the procedure of the process scheduled is not dynamic.

4.18. Cost and Energy Optimization

The paper in [28] proposes an Optimized Load Balancing algorithm (OLB) which not only balances the load among the servers but also reduces energy consumption and SLA violation. This paper uses Local regression (LR) for deciding whether host is overloaded or not and uses Minimum Migration Time policy to select the VMs to be migrated away from that host. In this paper under loaded Host Detection has been used to move VMs to other hosts so that this under loaded host can be put to sleep thus saving energy.

4.19. GFTLBS

Yao [29] presented a novel guaranteeing fault-tolerant requirement load balancing scheme (GFTLBS) to guarantee the fault-tolerant level of all services provided by the data center while balancing the load based on VM migration among the hosts. With GFTLBS, by moving CPU state, memory content, storage content and network connections of VM, VMs can be migrated from the host with the heaviest load to the lightest one while not violating the fault-tolerant requirements of all the services. Based on VM migration, the hardware utilization, power savings, availability, security and scalability can be increased without disrupting the customer applications running in the VMs. In this article all the VMs of different services are assumed to have the same capacity and the same load level. GFTLBS can balance the load among all the hosts in the data center and also guarantee the fault-tolerant level of all services provided in the data center and works well with various groups of the number of VMs and hosts.

4.20. Avoid Deadlocks

Rashmi [1] presented a load balancing algorithm to avoid deadlocks among the Virtual Machines while processing the requests received from the users by VM migration. The proposed algorithm avoids the deadlock by providing the resources on demand resulting in increased number of job executions. In this paper, hop time and wait time may be considered. Hop time is the duration involved in migration of the job from the overloaded VM to the underutilized VM for providing the service. Wait time is the time after which the VMs become available to service the request. Various users submit their diverse applications to the Cloud service provider through a communication channel. The Cloud Manager in the Cloud service provider’s datacenter is the prime entity to distribute the execution load among all the VMs by keeping track of the status of the VM. The Cloud Manager maintains a data structure containing the VM ID, Job ID and VM Status to keep track of the load distribution. The VM Status represents the percentage of utilization. The Cloud Manager allocates the resources and distributes the load as per the data structure and analyzes the VM status routinely to distribute the execution load evenly. In processing, if any VM is overloaded then the jobs are migrated to the VM which is underutilized by tracking the data structure. If there are more than one available VM then the assignment is based on the least hop time. The Cloud Manager automatically updates the data structure on completion of the execution.

5. Discussion and Conclusion

In this section we discuss the different techniques that were discussed in Section 4. We also compare these techniques based on the Metrics discussed in Section 2.

Table 2 shows a comparison among the reviewed techniques. The comparison shows the positives and negative points of each technique. For example, The Honeybee Foraging Algorithm achieves global load balancing through local serve actions and does not increase the throughput as the system size increases. However Performs well as system diversity increases. Furthermore, Active Clustering and Based
Random Sampling perform better with high and similar population of resources, but have not good performance when system diversity increases. Based Random Sampling achieves load balancing across all system nodes using random sampling of the system domain. In addition, OLB + LBMM algorithm makes better use of the resources, increases productivity and provides better results compared to Honeybee Foraging Algorithm, Based Random Sampling and Active Clustering. As for the Carton approach, we can see that this approach with very low computation and communication overhead is simple and easy to implement. Event-driven is able to either increase or decrease the scale of the play in several resources based on the variable load of the user, while violation of the quality of service is happened every now and then. Message oriented architecture as a middleware model has been pointed out to improve load balancing in distributed networks. Based on messaging techniques XMPP allowed resources to be monitored and provide availability of cloud resources. This technology is open for real time communication between various parties.

Table 2. Advantages and Disadvantages of Load Balancing Techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Advantages and Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honeybee Foraging Algorithm</td>
<td>Performs well as system diversity increases. Does not increase the throughput as the system size increases.</td>
</tr>
<tr>
<td>Based Random Sampling</td>
<td>Performs better as the number of processing nodes is increased. Performs not well as system diversity increases.</td>
</tr>
<tr>
<td>Active Clustering</td>
<td>Performs better as the number of processing nodes is increased. Performs not well as system diversity increases.</td>
</tr>
<tr>
<td>Carton</td>
<td>With Very low computation and communication overhead is Simple and easy to implement.</td>
</tr>
<tr>
<td>Event-driven</td>
<td>Is able to either increase or decrease the scale of the play in several resources based on the variable load of the user. Quality of services (QOS) breaches.</td>
</tr>
<tr>
<td>Server-based load balancing</td>
<td>Reduces service response time.</td>
</tr>
<tr>
<td>Black-Box &amp; Gray-Box</td>
<td>Has a drawback in the migration phase.</td>
</tr>
<tr>
<td>Min-Min Algorithm</td>
<td>Has a major drawback that it chooses smaller tasks first which makes use of resource with high computational power. Thus, this scheduler when number of smaller tasks exceeds the large ones is not optimal and it can lead to starvation.</td>
</tr>
<tr>
<td>Max-Min Algorithm</td>
<td>The waiting time of smaller tasks and the make-span may increase in this scheduler.</td>
</tr>
<tr>
<td>Message Oriented Model</td>
<td>As a middleware model has been pointed out to improve load balancing in distributed networks. This technology is open for real time communication between various parties.</td>
</tr>
<tr>
<td>OLB + LBMM</td>
<td>Efficient utilization of resources. Enhances system performance.</td>
</tr>
<tr>
<td>Fuzzy Logic</td>
<td>Can balance the load with decreases the processing time as well as improvement of overall response time, which are leads to maximum use of resources.</td>
</tr>
<tr>
<td>Join-Idle-Queue algorithm</td>
<td>Reduces the system load in which no communicative overhead is occurred while entering tasks, and real response time is not raised as well.</td>
</tr>
<tr>
<td>Power Aware LB Strategy</td>
<td>Reduces number of migrations.</td>
</tr>
<tr>
<td>Efficient LB Approach</td>
<td>Tries to reduce response time. Is not dynamic.</td>
</tr>
<tr>
<td>Cost and Energy Optimization</td>
<td>Reduces energy consumption. Reduces SLA violation.</td>
</tr>
<tr>
<td>GFTLBS</td>
<td>Balance the load and guarantee the fault-tolerant level of all services.</td>
</tr>
<tr>
<td>Avoid Deadlocks</td>
<td>Enhances the number of jobs to be serviced Improving working performance.</td>
</tr>
</tbody>
</table>

Based on the Discussed metrics, the reviewed techniques have been compared in Table 3.
### Table 3. Comparison of reviewed Load Balancing Techniques Based on Discussed Metrics

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Considered Metrics in each Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honeybee Foraging Algorithm</td>
<td>Throughput, Performance &amp; Scalability</td>
</tr>
<tr>
<td>Based Random Sampling</td>
<td>Throughput, Performance &amp; Scalability</td>
</tr>
<tr>
<td>Active Clustering</td>
<td>Throughput, Performance &amp; Scalability</td>
</tr>
<tr>
<td>Carton</td>
<td>Performance, Overhead &amp; Resource Utilization</td>
</tr>
<tr>
<td>Event-driven</td>
<td>Resource Utilization</td>
</tr>
<tr>
<td>Server-based load balancing</td>
<td>Performance &amp; Response Time</td>
</tr>
<tr>
<td>Black-Box &amp; Gray-Box</td>
<td>Overhead &amp; Response Time</td>
</tr>
<tr>
<td>Min-Min Algorithm</td>
<td>Throughput, Overhead, Response Time, Resource Utilization &amp; Performance</td>
</tr>
<tr>
<td>Max-Min Algorithm</td>
<td>Throughput, Overhead, Response Time, Resource Utilization &amp; Performance</td>
</tr>
<tr>
<td>Message Oriented Model</td>
<td>Response Time &amp; Performance</td>
</tr>
<tr>
<td>OLB + LBMM</td>
<td>Performance &amp; Resource Utilization</td>
</tr>
<tr>
<td>Fuzzy Logic</td>
<td>Response Time &amp; Resource Utilization</td>
</tr>
<tr>
<td>Join-Idle-Queue algorithm</td>
<td>Response Time, Performance &amp; Overhead</td>
</tr>
<tr>
<td>Central Load Balancing</td>
<td>Response Time, Performance, Throughput</td>
</tr>
<tr>
<td>Power Aware LB Strategy</td>
<td>Resource Utilization, Performance &amp; Energy Consumption</td>
</tr>
<tr>
<td>Dynamic LB Approach</td>
<td>Response Time, Performance</td>
</tr>
<tr>
<td>Efficient LB Approach</td>
<td>Response Time, Performance</td>
</tr>
<tr>
<td>Cost and Energy Optimization</td>
<td>Performance, Migration Time &amp; Energy Consumption</td>
</tr>
<tr>
<td>GFTLBS</td>
<td>Fault Tolerance, Resource Utilization &amp; Scalability</td>
</tr>
<tr>
<td>Avoid Deadlocks</td>
<td>Response Time, Performance &amp; Overhead</td>
</tr>
</tbody>
</table>

Load balancing is one of the main challenges in Cloud Computing. It is required to distribute the load evenly at every node to achieve a high user satisfaction and resource utilization ratio by making sure that every computing resource is distributed efficiently and fairly. So in this paper we compared different load balancing algorithms in Cloud Computing and concluded that we can use a special algorithm based on our needs. However Cloud Computing covers wide areas and as already stated, none of the above algorithms do not satisfy all the criteria. Therefore, the need to develop an adaptive method is necessary which is suitable for heterogeneous environments.

### References


