Load Balancing Algorithms in Cloud Computing: A Study

Rimjhim Singh
Department of Computer Science
Shiv Nadar University
Greator Noida, India
rs134@snu.edu.in

***Abstract*— With the growth in computing technology, Cloud computing is emerging as a rising technology offering various services to clients on the basis of 'pay per use' independent of time and location. The flexible and easy-to-use nature of cloud computing has encouraged more and more organizations, companies, and businesses to shift to cloud service providers. As cloud computing is a highly popular technology, it is becoming increasingly important to face and resolve the challenges associated with it. One such challenge is Load Balancing. Load balancing is a significant component of cloud computing infrastructure. With load balancing, we can ensure that the workload is distributed equally among all nodes and that no nodes are overwhelmed with the incoming load. By redistributing the workload fairly among all nodes, load balancing ensures efficiency and optimal performance of the services offered to the clients. To achieve this, various algorithms are implemented. This paper aims to study and provide an overview of these algorithms. Moreover, this study will also discuss the challenges associated with these algorithms and compare their performances.**

Keywords—cloud computing, load balancing, load balancing algorithms

I. Introduction

In recent years, cloud computing has been rapidly replacing traditional computing technologies, offering numerous services like storage and network resources, application development frameworks, application management, and user interface etc. These services are provided to the clients remotely, i.e., independent of the location, in a 'pay as per use' basis. Cloud computing is prevalent and valuable as it aims to provide maximum services at minimum cost. Although it is such a prominent technology, some various issues and challenges need to be tackled for its optimal utilization. Load balancing, which is the equal distribution of incoming workload among all nodes to avoid overload, is a central and significant issue when it comes to cloud computing. Load balancing makes sure that no node is too heavily or lightly loaded. In case of a component failure, load balancing ensures that the services run uninterrupted and the resources are utilized efficiently. The primary challenge is to schedule the incoming request in such a manner that there is a low response time, effective resource management, and at the same resources such as time should not be wasted.

Consider, as an example, an application on a cloud network that receives thousands of users daily. The cloud infrastructure has to ensure that no matter what the traffic is, the clients should be able to access the resources they are paying for. With high number of incoming requests, it is possible to have a low response time, which ultimately leads to user dissatisfaction. Load balancing ensures that the response time to all these users is quick and efficient. This is done by distributing the work load fairly among different nodes and ensuring high performance and better user satisfaction. There are several different algorithms that are used to achieve and implement load balancing in cloud computing, each with its respective challenges and problems that they address.

A. Cloud Computing Services

There are three major types of cloud computing models-Iaas, SaaS, and PaaS.

<u>IaaS: Infrastructure as a service</u>

The IaaS cloud service provider hosts the infrastructure like server space, storage, and networking hardware, CPU cycles, as well as the hypervisor or virtualization layer. The client has a lot of flexibility in using this service. Iaas is used by companies who do not want to store and maintain their own data centers.

<u>SaaS: Software as a Service</u>

Saas provides services such as application hosting and management and makes these available to users via the internet. The clients do not need to install or download any additional software to their infrastructure. All computations, management, and support are handled by the cloud services provider in the backend. The client uses the services on a 'pay per-use' basis.

<u>PaaS: Platform as a Service</u>

Paas offers services such as application development frameworks, operating systems, and deployment frameworks. Clients can utilize these services to create customized applications.

B. Types of Cloud Computing

There are four main types of cloud computing: private clouds, public clouds, hybrid clouds, and multi clouds

<u>Private Clouds</u>

A private cloud is a cloud environment that is wholly dedicated to one end user or group, typically running behind that user's or group's firewall. When the underlying IT infrastructure is devoted to a single client with totally segregated access, all clouds become private clouds. In this type of cloud computing, a client's data can only be accessed by that client.

Public Clouds

Here the IT infrastructure is not owned by any single end user. Anyone from anywhere can access public clouds. Examples are Amazon Web Services (AWS), Google Cloud, IBM Cloud, and Microsoft Azure.

Hybrid Clouds

Simply put, it is a combination of public and private clouds. Hybrid clouds comprise multiple separate yet connected environments. All environments are managed as a single environment using integrated management.

Multi cloud

This cloud approach comprises more than one cloud service from more than one cloud vendor which can be public or private. Different clients use these clouds but have a similar purpose, like hosting similar applications.

II. Problem Statement

Load balancing is a crucial part of cloud computing technology; hence, there is a need to study and understand various techniques used to implement it. This paper aims to study and provide an overview of various existing load balancing algorithms in use and to study the challenges that come with them. Different algorithms have different challenges associated with them. This study aims to review and compare the performance and utility of various algorithms. On the basis on available work, this paper reviews existing literature on various load balancing algorithms and aims to use metrics such as resource allocation, reaction time, and efficiency to classify them.

III. Literature review

There exists a variety of literature and research on cloud computing topics. Lately, there has been a rise in research in load balancing algorithms because of their eminent significance in cloud computing. This section reviews some of the existing work on load balancing algorithms in cloud computing. We observe the different ways in which load balancing algorithms are classified by researchers. Most existing literature on load balancing algorithms has been classified into two main categories: static and dynamic [15].

Ghomi et al [1] have proposed a load balancing algorithm survey. The paper covers the classification of these algorithms in seven different categories. The two broad categories are based on the system state and based on who initiated the process. The paper compares these algorithms from different categories and discusses their advantages and shortcomings.

Mesbahi and Rahmani [2], like other papers, broadly classified load balancing algorithms into two broad categories: static and dynamic. Further, the classification has also been made based on the general algorithm, architecture, and artificial intelligence. The paper discusses key challenges in designing and implementing load balancing algorithms. Furthermore, based on the study, the authors concluded that load balancing algorithms that satisfy dynamic, non-cooperative and distributive properties are superior. It was also concluded that more focus is being given to energy saving in recent load balancing algorithms.

Kalra and Singh [3] conducted a comprehensive study of load balancing algorithms in cloud computing and used optimization metrics such as consumer-desired and provider-desired. The categorization of the nature-inspired computing algorithms are done in five categories, namely Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), League Championship Algorithm (LCA) and BAT algorithm. In depth comparison of algorithms in each category has been provided. The paper goes on to discuss open issues which the techniques studied. The survey lacks broader classification as the concentration is on evolutionary algorithms.

A classification paper proposed by Kanakala et al [4] also groups the load balancing algorithms into static and dynamic. This paper identifies and discusses challenges and issues of load balancing in cloud computing. Geographical distribution of nodes, migration time, system performance, energy management and security are some of the challenges that are identified. The algorithms were compared on the basis of QoS (Quality of Service) metrics like throughput, speed, response time, migration time, concluding that a tradeoff exists.

Milani and Jafari [5] reviewed various existing load-balancing schemes and classified them into dynamic and hybrid subdomains. The algorithms were described on the basis of various metrics and also discussed their challenges and shortcomings. Issues were analysed to develop more efficient and effective load balancing algorithms. Energy consumption issues were not discussed in this paper.

Various algorithms can handle load balancing. To summarise the existing literature on load balancing algorithms, this paper surveyed and analyzed the existing load balancing algorithms. This study surveys the existing techniques, describes their properties and outlines their advantages and disadvantages.

IV. Need for load Balancing

Load balancing is a measure to distribute the incoming traffic and workload to all available resources equally. Performance of the cloud is directly influenced by the strategy used for load balancing. Any imbalance in the balancing of the load between resources leads to poor performance of the cloud service. Response time for tasks is reduced by the use of load balancing techniques and resource utilization is also improved. This is a low cost method to enhance the performance of the cloud.

Load balancing also aims to provide scalability and flexibility for those applications whose size may increase in future and requires more resources as well as to provide priority to jobs that need instant execution as compared with other jobs.

Load balancing also plays a crucial role in green cloud computing. Green computing [6] aims to implement procedures and policies that improve the efficiency of cloud computing resources in a way that reduces the energy consumption and environmental impact. Load balancing helps in green cloud computing in the following ways:

Reducing energy consumption: Load balancing techniques help in reducing the amount of energy that is consumed by the cloud platforms. This is done as balancing the workload among the nodes of a cloud avoids overheating, which in turn reduces the overall energy consumption.

Reducing Carbon emission: Reducing the carbon footprint left by cloud platforms is a big issue in green cloud computing. As carbon emission is highly influenced by the energy consumption by cloud, reducing the energy consumption reduces the carbon footprint. As load balancing techniques help in reducing the energy consumption, it also helps in the reduction of carbon footprint of cloud platforms.

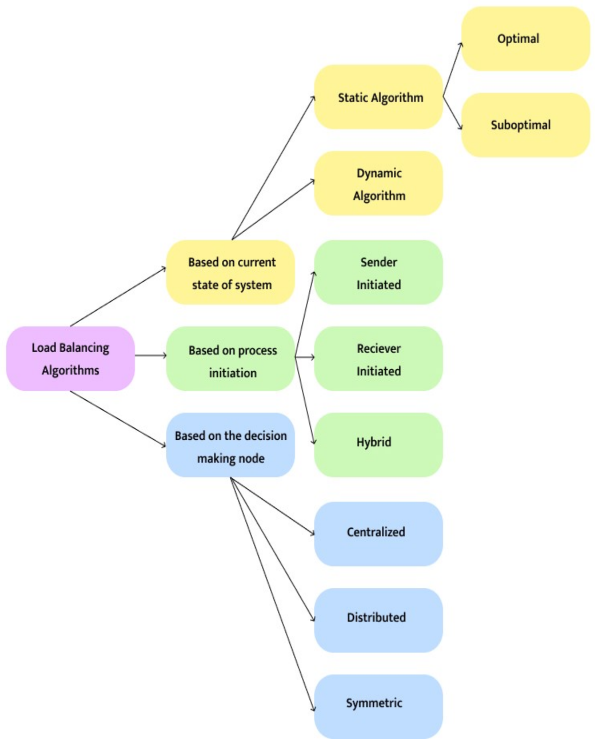


Figure 1: Classification of Load Balancing Techniques

IV. CHALLENGES FACED

Cloud contains enormous resources, and managing them requires careful layout and advanced planning. Prior to creating an algorithm, the availability of resources must be considered in light of the overall situation and it is necessary to identify the key problems that could have an impact on the algorithm's performance. In this section, the difficulties that must be overcome in order to suggest the best technique for resolving load balancing problems in cloud computing have been presented.

A. Geographical Distance Between Nodes

Data centers in the cloud are spatially distributed, and for efficient execution of user requirements, are treated as a single location system. The load balancing algorithms designed for small areas do not consider the geographical factors that can affect the computing process. Factors such as delay in communication, computing nodes distance, distance between client and resources etc are often overlooked. Thus the challenge here is to develop load balancing algorithms and techniques that take into account the mentioned factors that come into play in the case of distantly located nodes.

B. Single Point of Failure

Some load balancing algorithms are centralized, that means decisions for load balancing and work distribution are handled by a central node. This leads to the possibility of system collapse if the central node crashes. The challenge here is to design decentralized or distributed algorithms in which no single node has the control of the whole computing system.

C. Virtual Machine Migration

Virtualization in cloud computing allows several virtual machines to run on a single physical machine. In this case the client gets a feeling of running one physical machine. Due to the service on demand nature of cloud computing, if there is an incoming service request from a user, cloud resources have to be provided. Sometimes a physical machine gets overloaded and there arises a need to transfer VMs to another physical server to balance the load. While designing load balancing algorithms it is important to consider the time of migration and attack possibilities that are associated with VM migration.

D. Energy Management

Due to increases in cloud services demands, more and more computing resources are being used for processing, requiring large amounts of energy. Load balancing algorithms in cloud computing should be designed in a way that minimizes energy consumption and therefore minimizes carbon emissions of virtual machines in data centers. Resource utilization by load balancing algorithms should adhere to green computing standards as well as maintain Quality of Service.

E. Algorithm Complexity

In cloud computing, load balancing algorithms should be less complex, both in terms of implementation and operation. Performance and efficiency of a cloud system are influenced by the complexity of load balancing algorithms used.

F. Establishing Small Data Centeres

Small data centers are more cost efficient and energy efficient in comparison to large data centers. The underlying challenge is to design load balancing algorithms for adequate response time.

V. LOAD BALANCING METRICS

In this section we discuss some of the metrics which are used to evaluate load balancing algorithms in cloud computing. These metrics are also known as Quality of Service performance metrics in the cloud environment. These metrics are described below

A. Throughput

Total number of processes computed or completed per unit amount of time. Higher the throughput, higher the performance of the algorithm.

B. Internal Failure Adaption

In case of a system component failure the algorithm should be able to distribute the tasks effectively so that there is no perceivable downtime in the cloud. The goal is to have maximum system availability and throughput.

C. Response Time

Response time is the time taken by the system to respond to the client's requests and complete them. Load balancing algorithms should have minimum response time so that the cloud framework has high performance.

D. Scalability

Cloud frameworks often have a distributed architecture. By adding more nodes, distributed systems can be scaled horizontally to handle greater data processing and storage needs. In order to achieve high scalability, any effective load balancing algorithm must have optimal resource usage. Frameworks cannot continue to scale horizontally by adding nodes if only a small number of the nodes are assigned to handle requests by load balancing algorithms. As a result, any load balancing algorithm must follow certain rules that will optimise the allocation of resources to the request and thus achieve the highest level of horizontal scalability.

E. Utilization of Resources

All resources in a cloud system should be properly utilized for optimal performance of the cloud. A good load balancing algorithm has a high resource usage.

F. Overhead

Communication between two nodes requires data to be transferred from one node to another. This sharing of data requires user bandwidth. Such overhead should be taken into consideration when evaluating a load balancing algorithm.

G. Performance

Evaluation of system effectiveness should be done after complete execution of load balancing calculations.

VI. LOAD BALANCING ALGORITHMS

This section describes the existing load balancing algorithms. Before we discuss the algorithms. We have defined the different categories use to classify the load balancing techniques and algorithms.

Existing load balancing techniques can be classified into three categories.

1. Based on current state of system
2. Based on process initiation
3. Based on the decision making node

These categories are described in detail below:

<u>Based on current state of system</u>

1. Static

These techniques are independent of the current state of the system, and follow a predetermined set of rules. These algorithms are not flexible [7] and require access to previous knowledge of resources such as communication time, memory and storage capacity of nodes, processing power of nodes and more. Although the techniques being simple, it fails to detect the attached serves which often results in unequal distribution of resources and workload. The distributed systems that change their state dynamically cannot use static algorithms because system state is not taken into account.

Properties of Static algorithms:

- Their decision is based on a fixed rule for instance input load
- Static algorithms are not flexible
- Prior knowledge about the system is required

Static load balancing techniques can be further classified into the following categories:

<u>Optimal</u>: With optimal techniques, the DCN gathers resource information and transmits tasks to the load balancer, which allocates resources optimally in the shortest amount of time.

<u>Suboptimal</u>: The load balancer will calculate a suboptimal solution if it is unable to determine the optimal choice.

Examples of a few static techniques are Round Robin, Min-Min, Max-Min, Two-phase Opportunistic Load

Balancing. Some of these are discussed in the following sections.

2. Dynamic

Current state of the system is considered while making a decision in dynamic algorithms. Tasks can be transferred from overloaded machines to under loaded machines. Unlike static load balancing techniques, dynamic load balancing techniques are flexible, which results in enhanced system performance.

Steps followed by dynamic load balancing algorithms'

- Monitoring the load: This step involves the monitoring of the load and the state of resources.
- Synchronization: There is an exchange between load and state information in this step.
- Rebalancing Criteria: At this stage it is important to compute new workload distribution and base load balancing decisions on this calculation.
- Task Migration: Movement of data occurs in this step. This step takes place when a process has to be transferred.

Properties of dynamic algorithms:

- Their decision is made on the current state of the system.
- Dynamic algorithms are flexible
- Performance of the system is improved

Examples of some dynamic load balancing techniques are Agent-based Load Balancing (Singh et al. 2015), Honey Bee Behavior Inspired Load Balancing (Babu and Krishna 2013), Ant Colony Optimization (Nishant et al. 2012), and Throttled (Domanal and Reddy 2013). Some of these will be discussed in later sections.

Based on process initiation

Based on process initiation, load balancing techniques are classified as sender initiated, receiver initiated, and symmetric, as described below:

1. Sender initiated: If a node is overloaded, the search for an underloaded node is started for the distribution of workload. The request to identify the underloaded nodes is initialized by the sender.
2. Receiver initiated: Underloaded or lightly loaded nodes initialise the process of finding overloaded nodes for workload distribution.
3. Symmetric: This is a combination of sender initiated and receiver initiated processes.

Based on decision making node

Cloud systems are distributed systems and often decisions about what techniques are to be used have to be taken. Based on which node has the responsibility of taking the decision the load balancing techniques can be classified into:

1. Centralised
   A singular node acts as a central node and controls all information of the cloud system. This technique is time efficient because the central node takes all decisions rather than all nodes together, which saves time. Due to the centralized nature of the system, the cloud network is prone to single point of failure which makes it difficult for the system to recover.

   Issues:

   There are several issues associated with centralized techniques [7] which are discussed here. Centralized systems are non-scalable and following issues arise when we take large distributed systems into consideration.

   - Global information collection becomes an expensive process
   - Storing the global state information on the central node requires extremely high memory usage
   - The presence of a single central node can become a cause for communication bottleneck with a large number of processors

2. Distributed

In distributed load balancing, multiple nodes are responsible for decision making. Unlike centralized cloud networks, all nodes in a distributed cloud network have local information about the system. This kind of system ensures efficient task distribution as well as fault tolerance. If a node fails, the system is not brought down and can continue to function seamlessly.

3. Hierarchical

This technique follows the concept of master-slave mode. A tree structure is formed where different levels correspond to the level of cloud in which the nodes work. Parent node is responsible for the decision making, based on information received by slave node. [12]

Now we discuss some of the prevalent load balancing algorithms.

A. Round Robin

Round robin algorithm (Dave and Maheta (2014) is one of the simplest ways to distribute traffic among different servers. It is a static load balancing algorithm as the state of the servers is not modified while distributing the incoming traffic. The concept of time quantum is used in round robin algorithm. A time quantum is fixed and the load is distributed such that no process gets more time than one time quantum in one go. The resources are provided to a requesting client on the basis of the time quantum. First load is selected on a random basis and the job is allocated to other nodes in a circular manner. The loads are distributed equally to all virtual machines. The process of scheduling is described as follows. The scheduler selects a random node and assigns a VM to it. Then it moves onto the next node and assigns a VM to it. This process is carried out until all the nodes have been assigned at least one VM. This process is continuous. This allows the scheduler the flexibility to distribute the tasks to different nodes without waiting for the exhaustion of the resources of a particular node. The allocation order of the requests is maintained locally.

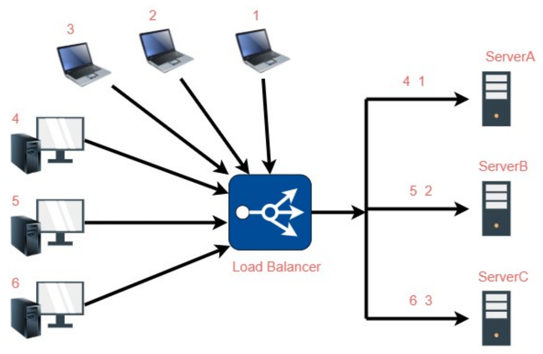


Figure 2: Round Robin Load balancing algorithm Source: Link

This concept is illustrated with a simple example. Suppose a cloud network has three servers namely server A, server B and server C. When an incoming request is received by the load balancer, it will assign the request (task) any of the servers, for the sake of this example let's say its server A. Now another request received in the cloud system can be forwarded to other server, let's say server B. Following incoming request will be assigned to the remaining server, server C. Now all servers have been assigned tasks. In this case if a request is received, it is forwarded to server A. Following requests will be forwarded first to server B and then to server C. This process carries on.

The algorithm is described as follows:

1. Begin
2. Assign the current request to the i mod Tth server.
3. Increment variable i value by 1.
4. Repeat steps 2 and 3 until there are no more requests.
5. End.

As round robin is a static algorithm, it is not adaptive or flexible to its environment. Consider the case in which server A is handling a large request. The load balancer will nevertheless forward incoming requests to server A if it is its turn in the circle. Other servers may sit idly during this time, under-loaded. Resource allocation and task distribution is not efficient in this algorithm. This problem has been tackled by the weighted round robin algorithm, which assigns some weights to each node and the weights govern the number of requests a node is allowed to receive.

B. *Min-Min*

This algorithm was introduced by Kokilavani et al. (2011). Minimum completion time and minimum execution time are both considered in this algorithm. The minimum execution time of the unassigned task is computed. Minimum completion time on all the resources is also computed. As it is a static algorithm, prior knowledge of the system is essential. The task is assigned to the node which has the minimum execution time. The problem of starvation is prevalent in this algorithm. The tasks with the highest execution time have to wait for an unspecified amount of time, so that all other tasks are assigned and updated.

C. *Max-Min*

Max min algorithm [10] is used in distributed which begins with a set of unscheduled tasks. This algorithm is same as the min min algorithm; the only difference is that the algorithms select the task with the maximum execution time and assigns it to a node with minimum execution time. The recently assigned tasks are removed from the tasks set and the process continues till the task set is empty.

Algorithm:

```
Step 1: For all submitted tasks in meta-task Ti
Step 2: For all resource Rj
Step 3: Compute Cij = Eij + rj
Step 4: While meta-task is not empty
Step 5: Find the task Tm consumes maximum completion time.
Step 6: Assign task Tm to the resource Rj with minimum execution time.
Step 7: Remove the task Tm from meta-tasks set
Step 8: Update rj for selected Rj
Step 9:Update Cij for all Ti
```

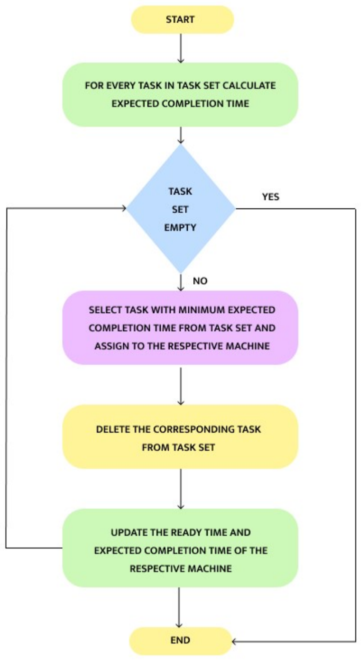

START
FOR EVERY TASK IN TASK SET CALCULATE EXPECTED COMPLETION TIME
TASK SET EMPTY
YES
NO
SELECT TASK WITH MINIMUM EXPECTED COMPLETION TIME FROM TASK SET AND ASSIGN TO THE RESPECTIVE MACHINE
DELETE THE CORRESPONDING TASK FROM TASK SET
UPDATE THE READY TIME AND EXPECTED COMPLETION TIME OF THE RESPECTIVE MACHINE
END

Figure 3: Min-Min Load Balancing Algorithm

D. *OLB Opportunistic load balancing*

OLB was proposed to achieve better resource utilization and response time. It is a static algorithm hence it does not consider the current state of the system. The aim of the algorithm is to keep all nodes busy and involved in the execution of tasks. As the current workload is not taken into account, the tasks are distributed randomly amongst the nodes. Request processing is slow because execution time of the tasks are not calculated, leading to a bottleneck situation even if some nodes are free. Hence the performance of the algorithm is poor.

E. *LBMM*

This technique is a combination of opportunistic load balancing and load balancing min min algorithm. [9] This algorithm improves the performance of the system as resources are utilized more efficiently and task proficiency is increased. The LBMM algorithm works in three levels. In the three layer structure the service node that used to carry out subtasks is on the third level. The service manager, who used to break the task down into some logical, independent subtasks, is the second level. The first level is the request manager, which is responsible for assigning the task to the appropriate service manager. The incoming task is divided into sub tasks and distributed among the nodes. The task assigning is based on the availability of nodes, memory availability and CPU capacity.

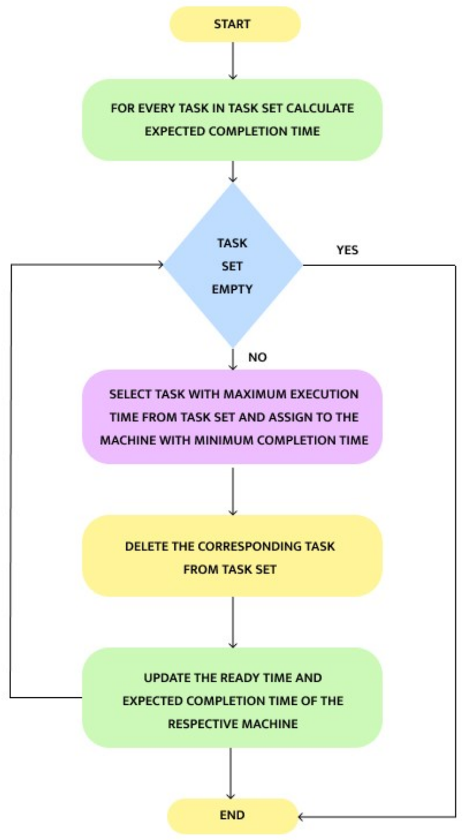

START
FOR EVERY TASK IN TASK SET CALCULATE EXPECTED COMPLETION TIME
TASK SET EMPTY
YES
NO
SELECT TASK WITH MAXIMUM EXECUTION TIME FROM TASK SET AND ASSIGN TO THE MACHINE WITH MINIMUM COMPLETION TIME
DELETE THE CORRESPONDING TASK FROM TASK SET
UPDATE THE READY TIME AND EXPECTED COMPLETION TIME OF THE RESPECTIVE MACHINE
END

Figure 4: Max-Min Load Balancing Algorithm

F. *Ant Colony Algorithm*

Inspired by ant colonies that work together efficiently in the search of food. In this technique load is balanced as well as make span is reduced. The assigned tasks hold the same weights/priority but fault tolerance is not considered. The assumption is that all tasks are mutually independent and computationally intensive.

G. *Honey Bee Behaviour*

This algorithm is inspired by the behaviour of honey bees for finding food. Upon finding the location where food is available, bees broadcast the message to their group by doing a waggle dance. This broadcast contains information about quality, quantity and location of the food. Other bees receive this information and can use it to acquire the food. This approach is utilized to design an algorithm for load balancing in cloud computing. The tasks are treated as bees and the virtual machines are treated as food resources. If any virtual machine is over loaded, it will migrate the task to an underloaded VM. After the migration happens, information about load on the machine as well as available tasks with their priorities is broadcasted. This technique helps other tasks in choosing their VM. A task with high priority chooses a VM with minimum number of priority tasks, this way it gets processed quicker. This algorithm not only achieves load balancing but also keeps track of the priority of different tasks. Throughput is maximised and response time is minimised.

H. *Throttled Load Balancing*

This algorithm depends upon the theory of suitable search of virtual machine. The task manager makes a list of virtual machines. By using the list, client request allotted to the relevant machine. If the size and capability of the machine is suitable for request, then the job is given to that machine. This algorithm is better than round robin algorithm.

A configuration table of virtual machines and their statuses is maintained and is used for balancing the load. When a request is received from the datacenter to allocate the VMs, load balancer chooses the VM in the list of available virtual machines. The request is allocated if a VM is found, if not then the load balancer return value of -1 to the datacentre [13]. Then this request is added to the queue and waits for a VM to become available. Being a dynamic load balancing algorithm, throttled load balancing has good resource utilization and performance. The drawback is that all VMs need to be searched and if it not efficient if the idle VM is at the bottom of the list. This is because the algorithm considers the first idle VM on the list and does not consider the current load of VMs.

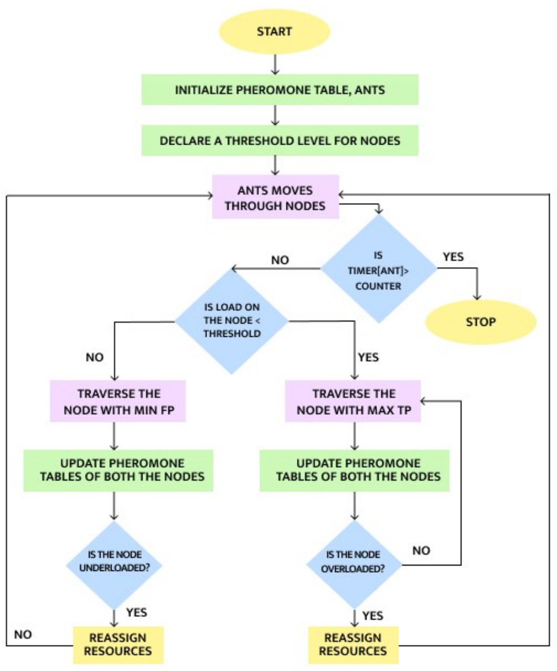

START
INITIALIZE PHEROMONE TABLE, ANTS
DECLARE A THRESHOLD LEVEL FOR NODES
ANTS MOVES THROUGH NODES
IS TIMER[ANT]> COUNTER
YES
STOP
NO
IS LOAD ON THE NODE < THRESHOLD
NO
YES
TRAVERSE THE NODE WITH MIN FP
TRAVERSE THE NODE WITH MAX TP
UPDATE PHEROMONE TABLES OF BOTH THE NODES
UPDATE PHEROMONE TABLES OF BOTH THE NODES
IS THE NODE UNDERLOADED?
IS THE NODE OVERLOADED?
NO
YES
YES
NO
REASSIGN RESOURCES
REASSIGN RESOURCES

Figure 5: Ant Colony Load Balancing Algorithm

I. Carton

Carton is a load balancing method that combines distributed rate limiting (DRL) and load balancing. [14]. Jobs are fairly distributed to the servers through LB. While DRL makes sure that resources are distributed equally. To increase performance and distribute the workload equally among all the servers, work load is dynamically assigned. This algorithm is simple to use and requires little communication.

J. Genetic Algorithm

This algorithm is based on mechanism of natural selection and genetics. The algorithm manages the load by minimizing the make span of tasks. [11]

A genetic algorithm is composed of three steps:

1. Selection
2. Genetic Operation
3. Replacement

These operations facilitate spread-out search space, to apply complex objective function and to avoid being trapped into local optimal solutions. A population of possible solutions to the problem at hand is generated and is allowed to evolve for multiple generations. This leads to better solutions.

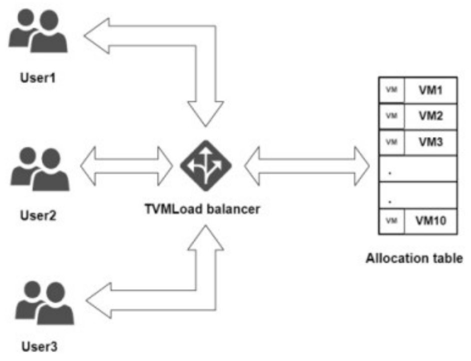
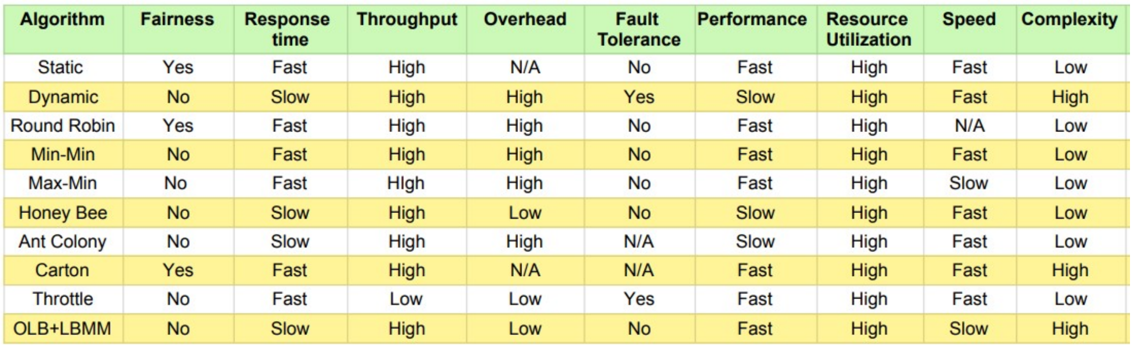


Figure 6: Throttled Model For Load Balancing

Load Balancing Algorithm	Advantages	Disadvantages
Static Load Balancing	Load Balancing decision is made during compile time Traffic is divided equally among nodes There are few complexes	Used only where load variaitons are few Cannot handle load changes throughput runtime
Dynamic Load Balancing	Load balancing takes place at run time Fault tolerance is there Only current state of system is considered	Nodes need constant check This technique is considered more complex
Round Robin	Uses fixed time quantum Easy to understand Gives better performance for short CPU burst Uses priority in running and arrival time	Longer time for larger tasks Task distribution is not efficient Not flexible to its environment
Min-Min	Completion time is short Handles smaller tasks most efficiently	Difficult to predict machine and task variations Starvation
Max-Min	Works better because requirements are known beforehand	Long time taken for task completion
Honey bee	Throughput increased, response time decreased	High priority tasks can't work without VM.
Ant Colony	Information collection is faster Tasks are independant Computationally intensive	Network overhead makes seraching take longer
Carton	Good performance Low communication required	Lower costs required
Throttled Load Balancing	Good performance Uses lists to manage tasks	Tasks have to wait
Genetic Algorithm	Better performance Better degree of load balancing	Low scalability and availability Overhead not evaluated Bottleneck

Table 1: Advantages and disadvantages of some load balancing algorithms



Algorithm	Fairness	Response time	Throughput	Overhead	Fault Tolerance	Performance	Resource Utilization	Speed	Complexity
Static	Yes	Fast	High	N/A	No	Fast	High	Fast	Low
Dynamic	No	Slow	High	High	Yes	Slow	High	Fast	High
Round Robin	Yes	Fast	High	High	No	Fast	High	N/A	Low
Min-Min	No	Fast	High	High	No	Fast	High	Fast	Low
Max-Min	No	Fast	HIgh	High	No	Fast	High	Slow	Low
Honey Bee	No	Slow	High	Low	No	Slow	High	Fast	Low
Ant Colony	No	Slow	High	High	N/A	Slow	High	Fast	Low
Carton	Yes	Fast	High	N/A	N/A	Fast	High	Fast	High
Throttle	No	Fast	Low	Low	Yes	Fast	High	Fast	Low
OLB+LBMM	No	Slow	High	Low	No	Fast	High	Slow	High

Table 2: Analysis of load balancing algorithms on the basis of evaluation metrics

VII. Future work

In this section we discuss the performance of above-mentioned algorithms.

When all of the servers have comparable or identical performance and are operating with equal loads, the round robin method functions effectively. Performance suffers as the demand on the servers changes since the server with the fewest resources gets the next job even though it isn't yet ready to handle the previous one. It is necessary to create an algorithm that uses cutting-edge task allocation models to address this flaw. The Min-Min algorithm does not provide fault tolerance or fairness. Its performance is good in small tasks. The algorithm also struggles with the starvation issue and is unconcerned about energy usage. The main problem for cloud providers is load imbalance, which is the biggest drawback. Future work on this will require the creation of an algorithm that shortens the processing time while improving resource efficiency. In the Max-Min algorithm, as the requirements are known beforehand, the performance is better and throughput is high. The processing of the OLB algorithm for static environments with centralised balancing is considered to be slow because it fails to calculate the current execution time. In the future projects, we must create an algorithm that determines the current execution time. In the future, we can use different crossover and selection procedures in genetic algorithms to provide outcomes that are more precise and effective. Ant colony algorithm is simple and less complex. In order to significantly reduce the time spent looking for candidate nodes, future work on the ant colony needs to examine the process for ant generation triggering and the strategy for pheromone update. The Carton algorithm requires low communication and its working is fair. The advantages and disadvantage of the discussed algorithms are presented in Table 1. Comparisons between these algorithms on the basis of the evaluation metrics is discussed in Table 2.

VIII. Conclusion

The demand for cloud computing services is increasing day by day. Hence load balancing techniques and research on load balancing algorithms is becoming crucial. Load balancing plays an important role in providing efficient and reliable services to users. This paper aimed to understand the existing load balancing algorithms and compare them based on their performance. This paper discusses the different ways in which Load Balancing algorithms can be classified, namely, based on the current state of the system, based on process initiation and based on the decision making node. Various Load balancing algorithms were discussed such as Round Robin, Min-Min, Honey Bee, Ant Colony etc. The advantages and disadvantages of these algorithms in different scenarios was also discussed. Moreover the algorithms were evaluated based on evaluation metrics discussed in the paper such as overhead, performance, fault tolerance and so on. All the algorithms discussed in this paper are not sufficient and there is a need to develop algorithms which considered fault tolerance and scalability. Here is huge scope for future research on the basis of improvement of these algorithms to get better and more efficient results.

IX. References

[1] Ghomi EJ, Rahmani AM, Qader NN (2017) Load-balancing algorithms in cloud computing: a survey

[2] Mesbahi M, Rahmani AM (2016) Load balancing in cloud computing: a state of the art survey

[3] Kalra M, Singh S (2015) A review of metaheuristic scheduling techniques in cloud computing

[4] Kanakala VR, Reddy VK, Karthik K (2015, March) Performance analysis of load balancing techniques in cloud computing environment.

[5] Alireza Sadeghi Milani and Nima Jafari Navimipour, Load balancing mechanisms and techniques in the cloud 3environments: Systematic literature review and future trends, Journal of Network and Computer Applications, http://dx.doi.org/10.1016/j.jnca.2016.06.003

[6] Jain, Nidhi & Chana, Inderveer. (2012). Cloud Load Balancing Techniques : A Step Towards Green Computing. International Journal of Computer Science Issues. 9.

[7] Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey PAWAN KUMAR and RAKESH KUMAR, NITTTR, Chandigarh

[8] Jixiang Yang, Ling Ling, Haibin Liu, "A Hierarchical Load Balancing Strategy Considering Communication Delay Overhead for Large Distributed Computing Systems", Mathematical Problems in Engineering, vol. 2016, Article ID 5641831, 9 pages, 2016. https://doi.org/10.1155/2016/5641831

[9] Shu-Ching Wang, K. -Q. Yan, Wen-Pin Liao and Shun-Sheng Wang, "Towards a Load Balancing in a three-level cloud computing network," 2010 3rd International Conference on Computer Science and Information Technology, 2010, pp. 108-113, doi: 10.1109/ICCSIT.2010.5563889.

[10] Kanani, Bhavisha & Maniyar, Bhumi & Mohammad, Sameer. (2015). Review on Max-Min Task scheduling Algorithm for Cloud Computing. SSRN Electronic Journal. 2. 781-784.

[11] Kousik Dasgupta, Brototi Mandal, Paramartha Dutta, Jyotsna Kumar Mandal, Santanu Dam, A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing, Procedia Technology, Volume 10, 2013, Pages 340-347, ISSN 2212-0173

[12] Aslam, Sidra and Munam Ali Shah. “Load balancing algorithms in cloud computing: A survey of modern techniques.” 2015 National Software Engineering Conference (NSEC) (2015): 30-35.

[13] Le, Hieu N. and Tran, Hung C.. "ITA: The Improved Throttled Algorithm of Load Balancing on Cloud Computing." *International Journal of Computer Networks & Communications (IJCNC)* 14 , no. 01 (2022): 25-39.

[14] Nuaimi, Klaithem & Mohamed, Nader & Alnuaimi, Mariam & Al-Jaroodi, Jameela. (2012). A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms. Proceedings - IEEE 2nd Symposium on Network Cloud Computing and Applications, NCCA 2012. 137-142. 10.1109/NCCA.2012.29.

[15] Afzal, S., Kavitha, G. Load balancing in cloud computing – A hierarchical taxonomical classification. *J Cloud Comp* **8**, 22 (2019). https://doi.org/10.1186/s13677-019-0146-7