Review of some load balancing algorithms in fog computing

Zeinab Haghbayan[1], Shiva Razaghzadeh[1]
Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran
Email: haghbayan.iauard@gamil.com (Corresponding author)
Email: Shiva.razzaghzadeh@gmail.com

Received: 10 Apr 2023 ***Revised***: 5 May 2023 ***Accepted***: 30 May 2023

ABSTRACT:
Fog computing is a new model in which computing resources are transferred from data centers to network edges. In fact, it is the same as cloud computing, but with the difference that due to the higher speed of information transfer, it is also used in technologies such as the Internet of Things. Load balancing is an important issue in fog computing. It should be noted that one of the most important challenges in achieving load balancing is resource management and proper scheduling. In fact, due to the existence of a large number of resources and the heterogeneous nature of these resources and environment, how to manage these resources and assigning task to each resource so that the number of tasks in the resources to be equal, is one of the most important reasons for examining load balancing algorithms. With extensive research, more effective solutions will be provided. In this paper, we introduce, compare, and evaluate some load balancing algorithms. Also, we will examine solutions for achieving load balancing using load balancing algorithms.

KEYWORDS: Fog computing, load balancing techniques, load balancing algorithms.

1. INTRODUCTION

Fog computing is a decentralized computing infrastructure in which all components, including storage, computation, data, and applications are efficiently and logically located between the cloud and the data source. In fact, by using the fog computing model, users can purchase their required services from the provider and also benefit from upgrading and maintenance systems. In the fog environment, the computational core is located in a local network or LAN, and the data is sent from end points or users to the fog gateway. Load balancing algorithms in fog computing offer benefits such as reducing traffic congestion on network nodes, creating an optimized task allocation schedule, faster access speed, better bandwidth utilization, cost reduction, high operational capacity, and improved computational reliability [1,2].

Load balancing plays a crucial role in fog computing, since due to the numerous resources, tasks, and large volume of requests, load balancing is necessary. This is ensured by using load balancing algorithms, which increase load balancing, efficiency optimization, reliability and network capacity. Load balancing also performs various actions such as sending distributed client requests to different networks, monitoring the load passing through multiple active servers, ensuring high availability, and providing flexibility for added or removed servers. Load balancing is used to enhance responsiveness and the usability of applications. A load balancer exists between the server farm and the client that accepts incoming application traffic. The application distributes the traffic among the support group and services using different techniques [3].

In the following, we will examine and compare the performance of load balancing algorithms, including fog computing and edge computing. We also examine the features and structure of fog computing, the reason for its need, the classification of architectural load balancers, fog calculation and task scheduling methods based on heuristic algorithms and evolutionary algorithms. Heuristic algorithms and evolutionary algorithms that have been studied in the background of the studies include: round robin, ant colony optimization, bee colony optimization, particle swarm optimization, greedy algorithm.

In the following, we will examine and compare the performance of load balancing algorithms, including fog computing and edge computing. We will also investigate the features and structure of fog computing, the reasons for its necessity, the classification of architectural load balancers, fog computation, and task scheduling methods based on heuristic algorithms and evolutionary algorithms. The heuristic algorithms and evolutionary algorithms studied in literature review include round-robin, ant colony optimization, bee colony optimization, particle swarm optimization, and greedy algorithm.

2. LOAD BALANCING IN FOG COMPUTING

2.1. PERFORMANCE OF FOG COMPUTING

The devices present in the fog are known as nodes. Any device with network connection, computational and storage capabilities can be a node that can be placed anywhere with a network connection. Various devices, from controllers to switches, routers, and video cameras, can act as a fog node. These nodes can be deployed in target areas such as offices or in a vehicle. When an Internet of Things (IoT) device generates data, these data can be received through one of these nodes, processed within the network, and then transferred to cloud data centers [4]. It is important to note that fog networks complement cloud computing and do not replace it. Fog has the ability to perform short-term analysis at the edge, while the cloud, due to its greater resources, is responsible for long-term analysis.

Edge devices and sensors generate and collect data, but sometimes they do not require the necessary resources for computation, storage, and advanced analysis. In such cases, fog computing is utilized. Although cloud servers have the capability to perform these tasks, they are often geographically located at a significant distance, leading to latency issues. Moreover, sending data from endpoints to the cloud requires an internet connection, which can result in issues such as decreased security, the risk of privacy breaches, and legal problems. This is especially concerning when sensitive data, subject to the regulations of different countries, is being transmitted. Common fog computing applications include intelligent networks, smart cities, smart buildings, and vehicular networks and software-defined networks (SDN) [1].

2.2. PERFORMANCE OF FOG COMPUTING MODEL COMPARED TO EDGE COMPUTING MODEL

The main idea behind fog computing compared to edge computing is to increase the speed of processing information. In general, data generated from the Internet of Things (IoT) can be processed in three locations: cloud data centers, network, or devices. In the field of Internet of Things (IoT) technology, where there is a vast amount of data that requires high processing speed, performing data analysis at the network or device level often results in superior performance. Both fog computing and edge computing technologies are closely linked to the cloud computing environment, and their primary objective is to minimize latency and maximize processing speed. Fog computing and edge computing are both efficient in enhancing the processing capability of data and information in a local network. However, the main difference between the two lies in the location where data processing takes place. In edge computing, data is processed directly on sensors that are physically close to the source device, without the need to transfer the data anywhere else. In fog computing, data is processed in the network space before being sent to cloud data centers [1]. Ultimately, both fog computing and edge computing are dependent on cloud computing, and their primary objective is to complement cloud computing technology. The aim of implementing fog computing is to bring fundamental analytical services to the network edge. By bringing computing resources closer to the required location, the distance that data needs to be transferred is reduced. As a result, the system's efficiency and performance is improved. Both fog computing and cloud computing provide storage space, applications, and data for users. However, fog computing has a closer proximity to the end user and has a wider geographical distribution. The foundation of fog computing is identical to that of cloud computing, with both relying on data, storage, and applications without the limitations of a specific physical location. By utilizing the fog computing model, users have the ability to acquire the services they require from providers, while also taking advantage of upgrade and maintenance systems. Fog computing differs from edge computing, with the difference lying in where intelligence and computational power are located. In the fog environment, the computational core is located in the local network (LAN), and data is sent from endpoints to the fog gateway. Then, it is transferred to the desired resources for processing. Edge computing is a subset of fog computing, where the computational core and power can be located at endpoints or the gateway. The use of edge computing reduces the risk of failure points. Because each device operates independently and determines which data should be sent to the cloud for analysis. The scalability of fog computing is greater than that of edge computing, and it offers a wider network perspective in comparison [1,2].

2.3. LOAD BALANCING AND THE REASONS FOR ITS NECESSITY

The fog computing layer includes devices that have lower storage and computational capabilities compared to cloud data centers. The fog layer is responsible for processing only the tasks that require immediate processing, while other tasks are sent to the cloud data centers. Sometimes, fog layers cannot handle a large volume of user requests, which causes load imbalance. Therefore, load balancing is required in order to distribute the workload evenly across all resources in the fog layer. The load balancer in the fog layer receives user requests and analyzes virtual machines in terms of their capacity and performance. If some virtual machines are underloaded, tasks are taken from overloaded virtual machines and assigned to underloaded ones [5].

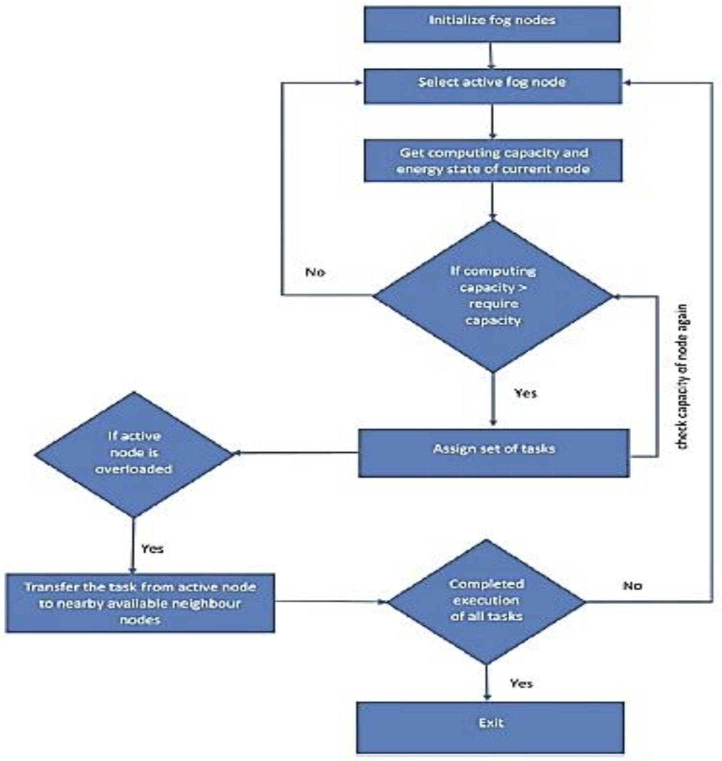
Initialize fog nodes
Select active fog node
Get computing capacity and energy state of current node
If computing capacity > require capacity
No
Yes
Assign set of tasks
check capacity of node again
If active node is overloaded
Yes
Transfer the task from active node to nearby available neighbour nodes
Completed execution of all tasks
No
Yes
Exit

Fig1. Flowchart of Execution Flow in Energy-Aware Algorithm

2.4. LOAD BALANCING TECHNIQUES

Various techniques for traffic load balancing are listed below:

- Centralized load balancing

In load balancing, there is a central node that manages the distribution of workload across the fog nodes in a distributed environment. Central nodes that store information and status of each node can be easily managed and repaired in case of failure.

- Distributed load balancing

This type of load balancing enables users to manage network traffic in a fog computing environment. In distributed load balancing, there is no central node that manages the calculation of each node in a fog environment. Instead, each node participates in the load balancing mechanism and transfers the load to the adjacent fog node. Decision making in such environments is based on the node's own observations and information about the system.

- Agent-based adaptive load balancing

Agent-based adaptive load balancing can be used for load balancing in fog environments. This technique enables maximum utilization of load-balanced server clusters (farms). Moreover, by utilizing the information obtained from the main server in the farm, it leads to an improvement in the decision-making process for all load balancing operations. All servers in the farm report on the current load of the (load-balanced) server. Then, the information is used to make decisions about which server is suitable to handle the requests.

- Fixed weighting

In Fixed weighting load balancing algorithm, fog nodes can be efficiently managed based on priority. In this technique, each server receives weights, and requests are routed to the server with the highest priority weight. If the server is rejected with the highest priority, then the server with the second priority will take over the responsibility for providing the services.

➢ Weighted Response Time

The aim of this technique is to minimize the maximum response time of servers by distributing the load among them based on their response time and weight. In this technique, each server's weight is calculated based on its response time, and tasks are assigned to servers with lower weights to reduce their response time and increase system responsiveness.

➢ Compatible Networking

This technique is often managed by a software-defined networking (SDN) controller, which uses software programs to send and receive data on a network. The main objective of this technique is to create flexible SDN network control that can improve traffic management for network nodes. The ultimate goal is to enable efficient responses to changing network needs.

➢ Minimum response time

The minimum response time technique is a commonly used algorithm in load balancing that aims to distribute incoming requests across multiple servers in a way that minimizes the overall response time. This technique works by identifying the server with the lowest processing time and routing incoming requests to this server. By doing so, the workload is evenly distributed and requests are processed as quickly and efficiently as possible, resulting in a seamless user experience. The minimum response time technique is often used in conjunction with other load balancing techniques to optimize system performance and ensure that applications can handle large volumes of traffic without experiencing downtime or performance issues [3].

2.5. CLASSIFICATION OF LOAD BALANCER BASED ON THEIR TYPE, ENVIRONMENTAL QUALITY, AND POLICIES

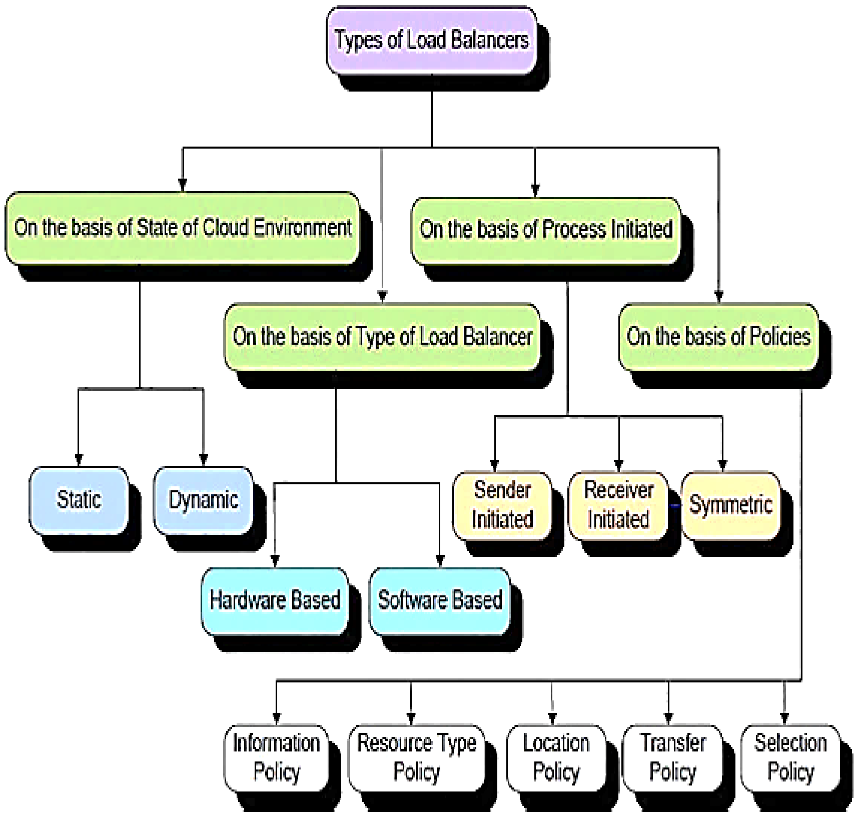

Fig2. Classification of types of load balancers

1. cloud environment quality

- Static Environment**:**
  This environment includes homogeneous resources. The Round Robin algorithm is an example of load balancing in a static environment.
- Dynamic Environment:
  This environment includes heterogeneous resources. The Load Balancing Min-Min (LBMM) algorithm is an example of load balancing in a dynamic environment.

2. Load Balancer
   Load balancers can be based on hardware or software. However, they are defined based on the initial sender and receiver.

- Initial Sender:
  A load balancer is started by heavily congested centers. To this end, information related to the bulk of data between different repositories is collected. Then, simultaneously with displaying the remaining

ongoing tasks, the bulk of data is distributed among less congested repositories.

- Initial receiver:
  Load balancing begins with low-accumulation centers that gather information about heavily-accumulated centers and take over the task from them.
- Symmetric:
  Load balancing is accomplished through coordination between the sender and receiver depending on the conditions.

3. policies
   Load Balancer are based on the following five policies:

- Information Policy
  Information policy specifies which data should be collected from various centers and locations.
- Resource type policy
  In this policy, resources are defined as either a server or a recipient of a process, and they will be displayed based on their accessibility.
- Location-based policy
  This policy specifies which target centers need to be selected for task transfer.
- Transfer policy
  Transfer policy specifies that the task needs to be selected by the neighboring center to be transferred to a farther center.
- Selection policy
  Selection policy defines what all processors need to participate in the load balancing mechanism [3].

2.6. STRUCTURE AND CHARACTERISTICS OF FOG COMPUTING

The load balancing framework in cloud computing, presented in this section based on the cloud computing environment, consists of three parts:

- The end-user layer
  End users are directly in contact with the fog layer and indirectly connected to the cloud layer through the fog layer. These end users generate requests, and these requests are directly transmitted to the fog layer. After processing in this layer, the fog layer immediately responds to the end users.
- Fog layer
  In this method, all tasks reach their respective processors on time and are executed in less time. Additionally, the need for energy consumption of idle virtual machines will be reduced. Therefore, load balancing in the fog layer will help in diminishing execution time, energy consumption, and implementation costs. This layer includes a manager who receives tasks from end-users. These tasks are then assigned to the task scheduler based on the principle of 'first-come, first-served'.
  The scheduler assists in prioritizing task execution, and load balancers subsequently take on the tasks and allocate them to available virtual machines. If there is an imbalance in the load on the virtual machines, the load balancer retrieves tasks from overloaded virtual machines and assigns them to less-loaded ones.
- Cloud Layer
  This layer provides security for data and includes large data centers with high storage and computing capabilities. This layer offers a range of services, including SaaS, IaaS, and PaaS [5].

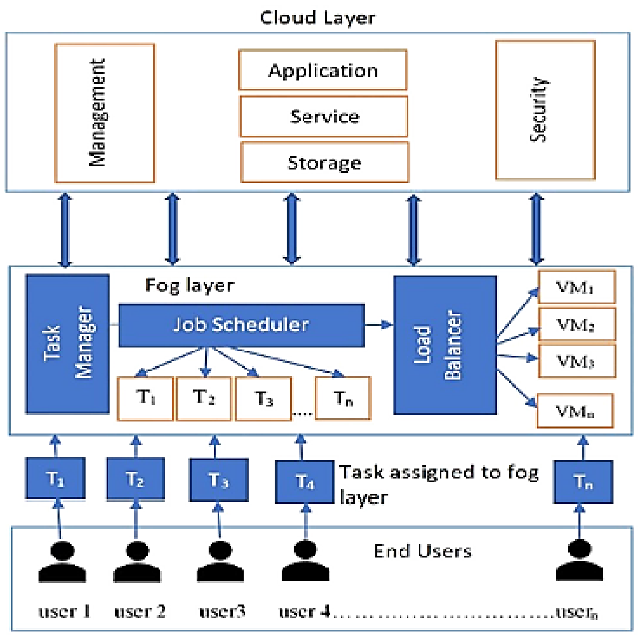

Fig3. Load balancing framework for fog computing [5].

2.1. LOAD BALANCING APPROACHES UTILIZING LOAD BALANCING ALGOTITHMS IN FOG COMPUTING

A lot of research has been done on how to balance the workload in fog computing using different algorithms. However, due to limited resources and information, researchers may have difficulty understanding how these techniques work and their underlying concepts.

- Scheduling Algorithm of minimum execution time - minimum completion time

In this algorithm, tasks are assigned to resources that have the minimum execution time. In other words, tasks are assigned to resources that can provide the minimum completion time. After a task is completed, the availability time of the resource is updated. This process is repeated sequentially until all desired tasks are scheduled. The workflow of this algorithm starts by selecting the smallest task and scheduling it, then proceeds to schedule the other tasks. The algorithm operates based on considering the execution time, completion time, and resource availability. The major issue with this scheduling algorithm, is the uneven distribution of workload on resources. The algorithm, prioritizes the minimum completion time of a process, thus scheduling tasks with the minimum completion time first, which benefits short tasks. The main advantage of this algorithm is its low response time [2].

- Scheduling Algorithm of Maximum execution time - minimum completion time

This algorithm is similar to the Scheduling Algorithm of minimum execution time - minimum completion time, with the difference that among the unscheduled tasks, the task with the longest execution time is assigned to the resource with the minimum completion time. In other words, longer tasks are scheduled first, followed by shorter ones. This process continues until all tasks are scheduled. This algorithm is used when there are more short tasks than long tasks. The main advantage of this algorithm is its simplicity in implementation and its main drawback is creating starvation for short tasks [2].

- Minimum Execution Time Scheduling Algorithm

This algorithm assigns the task to a resource that has the minimum completion time. First, the task is added to the list of unscheduled tasks, then a search operation is performed to find the resource with the minimum completion time. Finally, the task is assigned to the resource that has the minimum completion time. This algorithm significantly improves the maximum completion time of tasks overall. However, maintaining task and resource information calculated by the scheduling system has a high cost. This algorithm tries to ignore communication overheads in scheduling and minimize the average completion time of tasks on resources as much as possible. This approach has all the advantages of minimum execution time algorithms and fair load distribution, such as speed and creating a balanced load on resources [6].

- Round-Robin Scheduling Algorithm

The round-robin scheduling algorithm defines a loop as a queue and also defines a constant time quantum. Each task can only be executed with this quantum in turn. If a task is not completed in one quantum, it will return to the queue and wait for the next turn. The main advantage of this algorithm is that tasks are executed in their own turn and there is no need to complete previous tasks. Therefore, there is no hunger for other tasks in this algorithm. However, if the queue is completely filled or the workload is very heavy, a lot of time is required to complete all tasks. In addition, choosing a suitable time quantum for scheduling in this algorithm is difficult. This round-robin algorithm primarily focuses on the issue of fairness and justice [7].

- Task scheduling algorithms inspired by evolutionary algorithms

Typically, task scheduling algorithms do not consider all user-specified parameters and constraints. This is because considering all parameters and constraints would be very time-consuming and unacceptable for obtaining the optimal solution. Therefore, heuristic and evolutionary methods are used to reach an optimal or relatively optimal solution in a desirable time and under acceptable conditions. In the following, we will discuss some of these algorithms.

- Particle Swarm Optimization algorithm (PSO)

In PSO, a group of particles, each representing a potential solution to the optimization problem, moves through the solution space seeking the best solution. In research conducted using this method in the field of cloud computing scheduling, it has been concluded that better results can be achieved with this method compared to similar algorithms. This method focuses on optimizing the completion time of tasks assigned to cloud computing and does not consider its cost for the user [8].

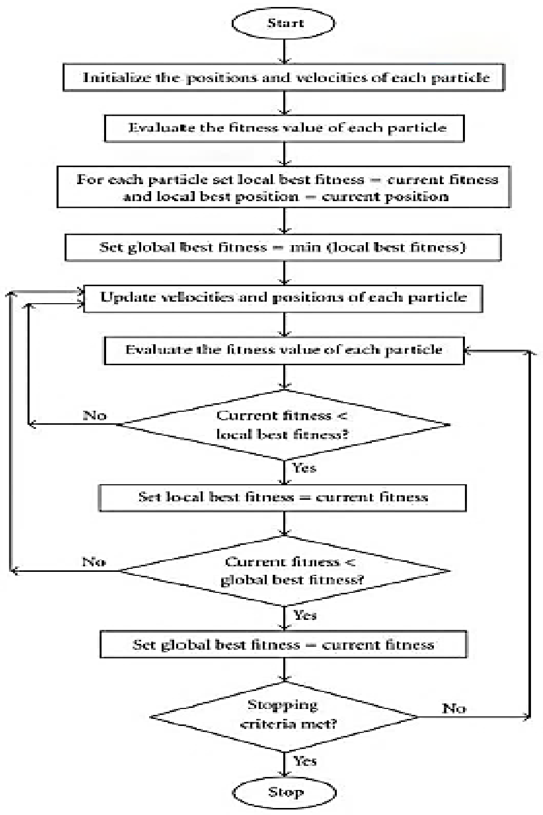
Start
Initialize the positions and velocities of each particle
Evaluate the fitness value of each particle
For each particle set local best fitness = current fitness and local best position = current position
Set global best fitness = min (local best fitness)
Update velocities and positions of each particle
Evaluate the fitness value of each particle
Current fitness < local best fitness?
No
Yes
Set local best fitness = current fitness
Current fitness < global best fitness?
No
Yes
Set global best fitness = current fitness
Stopping criteria met?
No
Yes
Stop

Fig4. Flowchart of Particle Swarm Optimization Algorithm

- Task scheduling based on the genetic algorithm

The genetic algorithm is one of the evolutionary optimization algorithm inspired by the process of natural selection and genetics. The algorithm works by maintaining a population of potential solutions, evolving the population through selection, crossover, and mutation operations, and selecting the best solutions as the algorithm progresses. In the genetic algorithm, time and cost have different weights. Each configuration in the genetic algorithm corresponds to a mapping from tasks to resources, which is a solution to the scheduling problem. The goal of the genetic algorithm is to minimize the fitted sums of time and cost in scheduling. These two fitted sums are multiplied by the average weights of time and cost for the tasks present in the schedule. The time fitting is defined as the maximum end time of activities for resources. The activity time for each resource is the sum of the completion times of the tasks in its queue, i.e., the sum of execution time and time spent on scheduling tasks for that resource. Cost fitting, is the total executing cost of the tasks currently in the queue. Overall fitting" is a combination of time fitting and cost fitting. This algorithm has generated a practical scheduling solution by adopting an evolutionary process to speed up convergence and consequently reduce search time. The main drawback of this approach is the high computational cost [9].

- Ant colony optimization algorithm-based resource scheduling strategy

In the ant colony algorithm, tasks are initially classified based on QoS because the objective is customer satisfaction. QoS has criteria such as completion time, network bandwidth, reliability, and total cost incurred to perform the task. After classification using MapReduce technology, these processes are scheduled. MapReduce has three entities: worker, user, and master. In [10], after classifying tasks based on QoS, the ant colony algorithm has been used, in which ants naturally find the optimal path for collecting food and returning to the nest. This algorithm has been applied to find the best and most optimal combination for allocating resources to processes [10].

- The Round Robin algorithm

The Round Robin algorithm is the simplest load balancing method that also provides tolerance for simple errors. In this technique, multiple servers, referred to as homogeneous identical servers, are configured to provide similar services. These servers are grouped under the same internet domain, with each server having its own unique IP address on the network.

When a user makes a request, the request is sent to a DNS server to retrieve the IP address associated with the domain name. The DNS server then selects one of the obtained IP addresses and returns it to the user. In subsequent requests, the DNS server provides the next IP address, and this sequential and cyclic process continues [11].

- Energy-Aware Load Balancing Algorithm

With the increasing demand for resources in the cloud computing layer, the number of hardware also increases, leading to an increase in energy consumption. Energy consumption can be reduced by decreasing the hardware requirements. Therefore, a framework is needed to reduce energy consumption in fog computing. In this article, an energy-efficient framework was presented to reduce energy consumption in the fog computing layer. Along with this, this section includes an energy-aware load balancing algorithm that will help reduce task execution time. By reducing the execution time, system performance can be improved, which will also help reduce energy consumption and implementation costs. The energy-aware load balancing algorithm for the environment is provided below:

The LB-EA algorithm aims to reduce energy consumption in fog computing by utilizing load balancing techniques. This is particularly important in scientific workflow applications, where there is a high volume of data transfer that requires more hardware, consequently increasing the energy demand. Proper utilization of all fog nodes through LB-EA's load balancing techniques can help optimize execution time and cost, while also reducing energy consumption in fog environments [5].

- Simulated Annealing (SA) Algorithm

The Simulated Annealing algorithm (SA) is a local search method used to find a global optimal solution for complex problems. The approach involves initially heating an object to a high temperature and then slowly cooling it down so that the system is almost always in a thermodynamic equilibrium state. In equilibrium states, the object has many configurations, each with a specific merit. A random perturbation is applied to the current configuration to obtain the next one, and its corresponding merit is calculated. The SA procedure begins by creating an initial mapping using a uniform random distribution. This mapping is then modified in a way similar to the genetic algorithm method, and the new delay time is evaluated. If the new delay time is better, it replaces the previous one; otherwise, a random number between 0 and 1 is selected. This number is then compared to Y. If RND < Y, the new mapping is accepted and used as the starting point for the next iteration. Otherwise, the previous delay time remains in place. The system's temperature is then lowered, which makes weaker solutions less likely to be accepted. Up to this point, one iteration of the SA process is completed. This algorithm terminates when no change in the delay time is observed for a specified number of iterations or the system temperature reaches zero [12].

4. REVIEW OF LOAD BALANCING METHODS IN FOG COMPUTING

- Researchers have discussed various approaches to load balancing, where the basis of these approaches is categorized into hardware-based and software-based load balancing approaches. They have mentioned different objectives that should be considered while advancing towards load balancing techniques. In summary, they described various load balancing algorithms, including the Honey Bee Algorithm, which achieves load balancing of tasks, and the Genetic Algorithm- algorithm. Some researchers, in order to implement their proposed framework, utilized a modified version of the Honey Bee Algorithm for load balancing. Their framework resulted in faster attainment, improved utilization of bandwidth, cost reduction, increased operational efficiency, and enhanced computational requirements for the Internet of Things (IoT) [13].
- Optimal scheduling algorithm without interruption for load balancing in fog Computing.

  The researchers utilized the Cloudsim simulation tool for implementation of case in the fog environment. This proposed framework can complete the tasks without interruption within the given deadline, enhance operational efficiency, and effectively address the increasing demands of end users [14].
- Employing a Greedy Algorithm-based Work Scheduling Approach for Achieving Load Balancing and Reducing Workflow, Time, and Cost.

  Greedy-based task scheduling has been widely employed in several studies to improve the time efficiency of rotations and minimize costs associated with user-submitted tasks within specific time slots. This algorithm employs a greedy approach by selecting the most suitable resource based on its cost and rotation time, utilizing a task priority formula. Notably, this algorithm outperforms sequential scheduling, yielding superior outcomes.

  The experimental results demonstrated a statistically significant improvement in the overall execution time by up to 9% when utilizing the Min-Max algorithm compared to the Min-Min approach, while concurrently achieving an enhanced system utilization rate compared to the Max-Min algorithm. Furthermore, the total completion time and average response time exhibited statistically significant reductions of 7% and 9% respectively. Also, the study revealed that the employed scheduling algorithm incorporated specific constraints. Consequently, a greedy approach was adopted for activity selection within the algorithm. The results demonstrated substantial enhancements in workflow, time efficiency, and cost reduction for each task feature [8].
- Dynamic resource allocation algorithms that integrate available resources in the network for load balancing.

  Researchers presented a systematic framework based on the neural network of the human body as a model for the features of cloud data centers. They deliberated on the concept of an iterative game strategy as a means of incentivization and supervision for actively carrying out tasks. This game constitutes an infinitely repeated game without a terminal stage. They employed the Nase to compute the results, aiming to achieve maximum capital efficiency. The foundational Hadoop system framework was utilized for implementing their proposed scheme. They determined the SLA violation rate and the corresponding completion time for various workloads. Furthermore, they conducted a comparative analysis between their algorithm and the Min-Min algorithm, as well as the MBFD algorithm [15].

  The algorithm of virtual machine allocation for load balancing and energy consumption reduction in data centers, based on Particle Swarm Optimization (PSO) and multi-resource allocation model

  The researchers employed the total Euclidean distance as the fitness function within the PSO algorithm and subsequently compared the results with results of MBFD and MBFH algorithms. The CloudSim simulator has been utilized for executing the plan [16].
- The developed strategy for load balancing, utilizing the migration of virtual machines through an algorithm inspired by the behavior of honey bees.

  This approach guarantees that each node in the system is efficiently utilized. The nodes are characterized by two important criteria: total migration time and service accessibility**.**

  The researchers have utilized various algorithms, including load balancing, task load balancing inspired by honey bee behavior, and the concepts of migration and virtualization, in the calculation of cloud resource utilization [17].

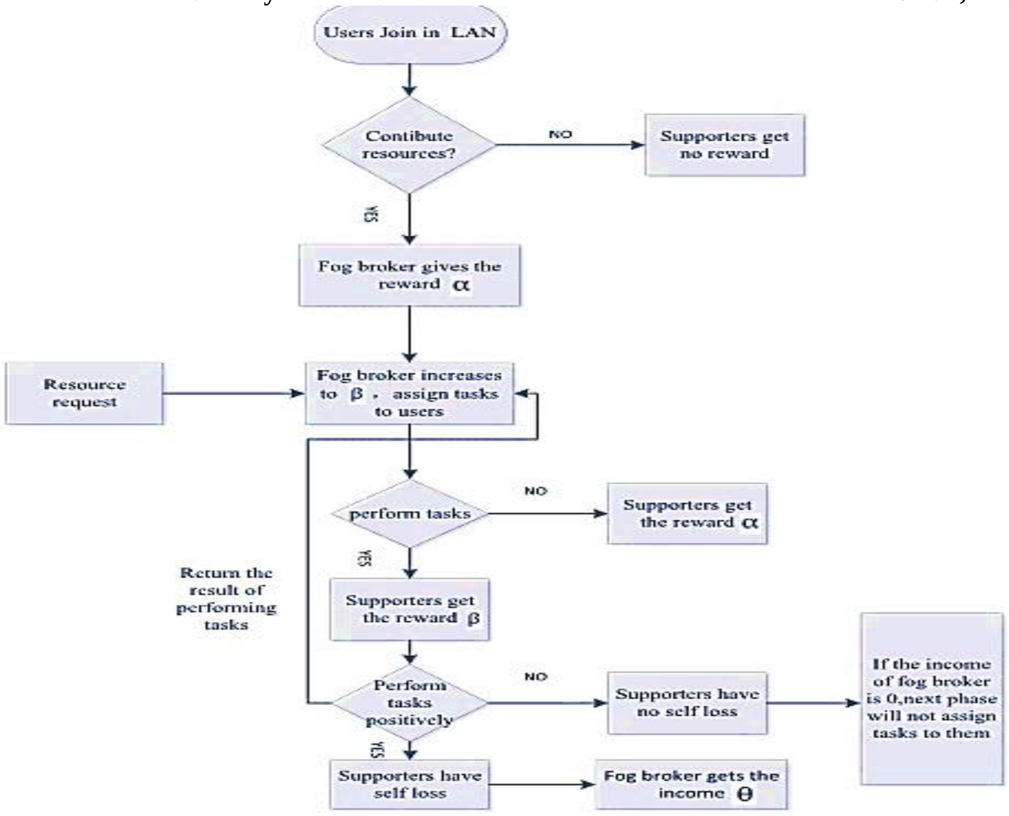
Users Join in LAN
Contibute resources?
NO
Supporters get no reward
YES
Fog broker gives the reward α
Resource request
Fog broker increases to β , assign tasks to users
perform tasks
NO
Supporters get the reward α
YES
Return the result of performing tasks
Supporters get the reward β
Perform tasks positively
NO
Supporters have no self loss
If the income of fog broker is 0,next phase will not assign tasks to them
YES
Supporters have self loss
Fog broker gets the income θ

Fig5. Flowchart of the Collective Financial Provisioning Algorithm

- Energy-Aware Virtual Machine Placement Algorithm for Load Balancing.
  The researchers have employed the task clustering technique for load balancing, which aids in reducing energy consumption in cloud data centers. The proposed technique combines various small tasks with large tasks to reduce the workload on virtual machines. The proposed load balancing technique is based on the MIN-MIN algorithm.
- Load Balancing Equilibrium Technique for Fog and Cloud Computing Calculation Using Nature-Inspired Algorithms.
  Several different load balancing methods have been proposed in previous studies for cloud computing. In some of these studies, load balancing techniques have been designed by drawing inspiration from existing natural techniques such as particle swarm optimization. The primary objective of these methods is to reduce the challenges in the load balancing process in cloud systems by leveraging the experiences gained from nature.
- Virtual Machine Allocation Algorithm to Achieve Load Balancing.

The Virtual Machine Allocation Algorithms Address the Allocation of Resources to Devices. Virtual Machine Allocation Strategies play a crucial role in load balancing algorithms. In this paper, researchers propose Virtual Machine Allocation Strategies aiming to prioritize low-priority tasks (tasks with high time constraints [18].

```
1.  Begin
2.  Arrival of New job
3.  If(New job.deadline < all jobs running in host)
4.  High priority job=New job
5.  If (VM is available)
6.  Allocate High priority job to that VM
7.  Else
8.  susend job ← selection of job for execution of high
    priority job();
9.  Suspend (Suspend job)
10. Allocate High priority job to VM form witch a job was
    suspended
11. End if
12. Execution of all jobs running in the VM
13. If (completion of a job witch is running in VM)
14. Resume (Suspend job)
15. Allocate the resumed job to that VM
16. End if
17. Execution of resumed
18. End
```

Table 1. Comparison of Load Balancing Algorithms in Cloud Computing

Algorithm	**Advantages**	**Disadvantages**
Minimum execution time - minimum completion time	Simplicity of the response time algorithm and fast execution	Creating starvation for long tasks and addressing load imbalance on resources
Maximum execution time - minimum completion time	The simplicity of the algorithm, fast execution and low response time	Creating starvation for short tasks and addressing load imbalance on resources
Minimum execution time	High execution speed and easy implementation	Load imbalance on resources
Round-robin scheduling	Reasons and fairness of avoiding starvation	Long execution time when dealing with high workload volume
Honey Bee Scheduling Algorithm	Low execution time, achieving load balance on resources	Use a coordinated number of variables
Particle Swarm Optimization	Reducing execution time	Neglecting cost considerations
Genetic Algorithm	Intelligent Allocation	Neglecting job completion time and cost
Ant Colony Algorithm	Relatively low execution time, guaranteed convergence to optimum solution	Poor performance in a large number of resources, slow convergence speed
Simulated annealing algorithm	Intelligent Allocation	Poor performance in a small number of resources

6.CONCLUSION

Due to the increasing use of cloud computing and the high volume of requests, achieving load balancing has become a significant challenge. Consequently, there is a need for appropriate and optimized algorithms to establish load balancing. Some of the reasons behind these challenges include the heterogeneity of resources and the environment, task allocation methods, and the growing importance of load balancing. Load balancing also involves issues such as resource and task management, which can be addressed by utilizing load balancing algorithms. By solving these issues using load balancing algorithms, costs and execution time can be reduced, and operational energy and capacity can be improved. Therefore, extensive research has been conducted to enhance and utilize load balancing algorithms. In this paper, we examine and evaluate various types of load balancing algorithms and classify them accordingly. This classification enables us to compare and investigate load balancing algorithms and utilize them in the paper**.** Furthermore, we have presented the applications and features of load balancing techniques. This organized classification can be beneficial for researchers and developers to expand their ideas on load balancing algorithms in fog computing. Different load balancing algorithms have been discussed in this paper based on their strategies in dynamic environments. In the future and upcoming works, these load balancing algorithms can be utilized with existing technologies, as observed in numerous applications such as online gaming, video streaming, and social applications that use load balancing algorithms for social welfare purposes. With the rapid proliferation of sensors, this technology will gain special importance. The technology is characterized by high demand in almost every field. By employing various advanced techniques, we can achieve load balancing and improve its efficiency.

REFERENCES

[1]. L. Atzori, A. Ier, G. Morabito, "**The internet of things: A survey. Computer networks**,"**,** Computer Networks, vol. 54, no. 15, 2010.

[2]. H.Chen, F.Wang, N.Helian, G.Akanmu, **"User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. "**, In 2013 National Conference on Parallel computing technologies (PARCOMPTECH);pp. 1-8, IEEE, February.2013.

[3]. S.P.Singh, R.Kumar, A.Sharma, A. Nayyar, " **Leveraging energy-efficient load balancing algorithms in fog computing. "**, Concurrency and Computation: Practice and Experience, vol. 34, no.13, 2022 .

[4]. F.Bonomi, R.Milito, J. Zh, & S.Addepalli, " **Fog computing and its role in the internet of things**. **"**, In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16, August .2012.

[5]. M.Kaur & R.Aron, **"Energy-aware load balancing in fog cloud computing**."**,** In Materials Today: Proceedings, 2021.

[6]. G.Ming & H.Li , **"An improved algorithm based on max-min for cloud task scheduling**."**,** Recent Advances in Computer Science and Information Engineering; Volume 2, p.p. 217-223, 2012.

[7]. S.Mohapatra, S.Mohanty, K.Smruti Rekha, " **Analysis of different variants in round robin algorithms for load balancing in cloud computing**. **"**; International Journal of Computer Applications,

vol. 69, no. 22, pp.17-21, 2013.

[8]. S.Pandey, Li.Wu, S,M,Guru, R,Buyya, **“A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments.”**, In 2010 24th IEEE international conference on advanced information networking and applications, pp. 400-407, ,April.2012.

[9]. Zhao, C., Zhang, S., Liu, Q., Xie, J., & Hu, **“Independent tasks scheduling based on genetic algorithm in cloud computing.”**, September., In 2009 5th international conference on wireless communications, networking and mobile computing (pp. 1-4). IEEE, 2009.

[10]. L.Zhu, Q.Li & L.He, **“Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm. ”**, International Journal of Computer Science Issues (IJCSI), 9(5), 54, 2012.

[11]. A.Choudhary, M. C. Govil, G.Singh, L. Awasthi, E.Pilli, **“Task clustering-based energy-aware workflow scheduling in cloud environment. ”** , IEEE 20thInternational Conference on High Performance Computing and Communications (HPCC)) (pp. 968-973). IEEE, June.2018.

[12]. G.Gan, T.Huang, S.Gao, **“Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. ”**, In 2010 International Conference on Intelligent Computing and Integrated Systems, pp. 60-63. IEEE, October**.** 2010.

[13]. M.Aldağ, Y.Kırsal & S.Ülker , **“An architecture for load balancing techniques for fog computing environment. ”**, , International Journal of Computer Science and Communication, vol. 8, no. 2, pp. 43-49, 2015.

[14]. M.Verma, N.Bhardwaj, & A.Yadav, **“Real time efficient scheduling algorithm for load balancing in fog computing environment. ”**, ,”Int. J. Inf. Technol. Comput. Sci, vol. 8, no. 4, pp. 1-10**,** 2016.

[15]. Y.Sun., & N.Zhang, **“ A resource-sharing model based on a repeated game in fog computing. ”**, Saudi journal of biological sciences, vol. 24, no. 3, pp. 687-694, 2017.

[16]. A.Xiong and C.Xu, **“ Energy efficient multiresource allocation of virtual machine based on PSO in cloud data center. ”**, ”Mathematical Problems in Engineering, 2014.

[17]. M.Mukhija, **“A resourceful technique for virtual machine migration in fog computing.”**, International Journal of Innovative Science and Research Technology, 6(6), 167-170, 2016.

[18]. A. Saraswathi, Y.R.A. Kalaashri, S. Padmavathi, **“Dynamic resource allocation scheme in cloud computing.”**, Procedia Computer Science, 47, pp. 30-36, 2015.