Load Balancing by Min-conflicts scheduling with Enhanced Eagle Aquila Optimization for Fog Computing Environment

V.Gowri

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai,

gowrivageesan87@ gmail.com

ABSTRACT

Fog Computing" (FC) is a collection of abstracted computer resources. An Internet-based growth where resources are offered as a service through the Internet which is rapidly scalable and abstracted has grown to be a serious issue. The Fog server optimization routine becomes a difficult problem in fog computing. It primarily focuses on Load Balancing (LB) in fog data centers to increase the performance of host and reduce the number of active host machines. It should use migration strategies to move the Virtual Machines(VM) from the overloaded host to the lighter host to equalize the load throughout the whole information center. Min-conflicts scheduling with Enhanced Eagle Aquila Optimization (MCS-EEAO) approach is proposed to handle a Constraint Satisfaction Problem (CSP) in the fog sever. Dynamic Compare and Balance Algorithm (DCABA) also reduces the number of host machines that need to be maintained and the cost of fog services. In contrast to conventional server optimization schemes that only take into account LB and the allocation of resources based on the consumption of CPU, RAM, and Bandwidth in physical servers. High latency and safety issues are important challenges for fog computing The proposed system is based on fog technology which offers safety control, data management, fast response and processing time. It also reduces the service costs in the fog business while making efficient use of the resources that are already accessible.

Keywords: Fog Computing; Load balancing; Enhanced Eagle Aquila Optimization; Migration Techniques; LB; Resource Management

1. Introduction

FC a developing novel computing mechanism has become a hot topic in industry and research as a result of the rapid growth and widespread usage of Internet technology. It is intended to offer computing as a service to meet the basic needs of the community. Businesses and individuals access software services anywhere in the world on request using its facilities [1]. As a result, it provides a novel approach for the dynamic delivery of computing services, which is frequently supported by ensembles of networked Virtual Machine (VM) located in cutting-edge data centers. It is a method of spreading computing that transfers tasks from individual PCs to distant computer clusters for data analysis at fast internet speeds.

LB traffic monitoring control system that tracks user requests and server traffic. It will send incoming requests to the alternative available servers whenever a server goes offline. The LB will automatically redirect requests to a new server [2-3]. Both static and dynamic LB approaches are possible. The static LB technique is successful by gathering fundamental information about each server. Since there is less communication when using a static LB technique, execution time is reduced [4]. Unsupervised learning, supervised learning, and Reinforcement learning (RL) are the three categories used to categorize ML methods [5,6]. Unsupervised learning has issues with similarity-based grouping and labeling. An agent develops to engage with the surroundings to obtain a reward in RL [7].

FC is viewed as an infrastructure that brings cloud computing services very closer to the end people. Surprisingly, FC utilizes this idea when the virtual fog infrastructure is situated closer to the end individuals, only among end-user devices and the cloud [8]. A fog is closer to the earth and the clouds are higher in the sky. Similar definitions claim that fog computing will enable computation at the network edge and would enable the delivery of new services and applications designed expressly for the Internet's future. For the first time, researchers can define fog computing more accurately by stating that it is not only used at the network edge [9] and also offers storage, networking, and computation services among the endpoints and data centers of traditional cloud computing [10].

2. Related works

Applications for FC are executed in a multi-layer framework that interconnects software and hardware activities, enabling dynamic reconfigurations for a variety of applications while operating transmission services and intelligent systems [11–12]. On the other side, the edge server develops a direct transmission service and controls unique applications in a set logic location. Edge computing is restricted to a few auxiliary devices, whereas FC is hierarchical [13]. In addition to network and computing, FC also addresses the speed, storage, and management of information [14]. The following features must be used by an IoT consumer or smart end device when using a FC service to distinguish fog computing from other computing standards [15].

FC uses a layered paradigm that allows for unrestricted access to a large pool of adaptable computer resources. This paradigm, which is made up of fog nodes that are situated between centralized services and intelligent endpoints, aids in the arrangement of dispersed, delay-aware services and applications [16]. The context-aware fog nodes support a regular data management and communication mechanism. These nodes would be grouped vertically, horizontally, or according to the fog node delay distance from the intelligent user devices [17]. The focal point of the fog structure is the fog node. Fog nodes can be actual components including switches or gateways or virtual components including VM or virtualized switches. They could be tightly paired with connectivity networks or smart-end devices and have the ability to provide computational resources to the aforementioned devices [18].

A mathematical model that concurrently investigates and addresses response time to the end user is proposed [19]. By using max-min cloud technique, the load is distributed and server routine is maximized through LB of VMs based on their capabilities [20]. In a virtual structure, heterogeneous VMs are used, each of which has a varied number of virtual CPUs allotted to various subtasks. The cloud computing services are implemented by the proposed probability architecture [21]. An approach built on architecture identifies key characteristics and allays worries in cloud environments, including multi-tenancy [22].

Heterogeneous VMs and stochastic response times that fulfill requests according to broad likelihood distributions are features of multi-tenant models.

The method is put into practice via LB and each sub-tasks execution time is predetermined [23]. In Mobile Cloud Computing (MCC), the authors of [24] introduce cloudlets. Smart mobile devices and the cloud interact more fluidly when they are close to one another. Clients receive high-quality services via cloudlets. The Auction method is a new Incentive-based method that is used to encourage the cloudlet to share resources [25]. In this paradigm, buyers place bids on resources, sellers make a resource pool available for a specific task, and auctioneers act as middlemen in the resource auctioning process. The Incentive-Compatible-Auctions-Mechanism (ICAM) method, developed by the researchers depends on the auction method and is intended to service neighboring smartphones. The workload on the centralized cloud is balanced, latency is decreased, and resources are used effectively [26]. Truthfulness, budgetary harmony, effective resource management, computational effectiveness, and payment and clearance prices in polynomial time are all provided by ICAM.

3. Problem Description

In theory, Fog computing is a dispersed system where resources are disseminated across the network. The entire system resources must work together to fulfill a client request, which requires interaction between different system components to create an element or subset of elements that can handle the request. This may result in network bottlenecks and an uneven charge in a distributed system, where certain elements receive an excessive amount of power while others receive little to no power. LB is one of the difficult problems that fog computing systems must deal with and is crucial to the success of the technology. LB scheduling method called the MCS-EEAO approach is proposed to handle a CSP.

The primary factors to be taken into account while constructing an LB method are load calculation, load comparison, system stability, system performance, the connection among nodes, the type of work to be transferred, and node selection [27]. An efficient solution needs to be created to address the issues with LB

in the fog network. The proposed method's main goal is to use the DCABA methodology to create a LB framework used to achieve a proper balance of load across all the resources of fog servers while maximizing resource utilization.

Once the load indexes for all the resources have been calculated, an LB procedure can be started to utilize the resources efficiently while allocating resources to the appropriate node to lower the load value. Therefore, allocating resources to appropriate nodes is an optimal distribution problem, or LB relies on a variety of optimization techniques, including the genetic algorithm and enhanced genetic algorithm. Although these approaches do not address the exploration and exploration issue, they are very effective in supplying neighbor solutions. Therefore, while the genetic algorithm is a conventional and established method, using the efficient optimization process instead can result in improved LB. To carry out the LB operation in our proposed study, therefore intended to use a recent optimization method called DCABA. First, the order of requests and the accessibility of virtual servers would be taken into account when maintaining the index table.

4. Proposed System

This section introduces a three-layer design for the Fog system architecture. The three layers of the proposed model are the user-end layer, Fog layer, and core-cloud layer, which are shown in Figure 1. Core fog is a centralized network that is made up of distant computers for information processing, management, and storage. At the network's edge, FC is an extension of core fog. To minimize the delay, fog and end users communicate within one hop of each other. Fog has a finite number of VM that process, store, and handle information for end users.

The two regions in this model are Europe and Asia. Due to the high population density in these two regions, Europe and Asia are selected as region 1 and region 2, correspondingly. Each region features three groups of buildings and one fog. On these fogs, there are anything between 30 and 45 VMs. During off-peak hours,

each cluster's typical user count can reach 50, but during peak hours, it can exceed 200. Clusters are linked to the macro grid and the fog. Fog responds to the client's request by allocating resources to the end user.

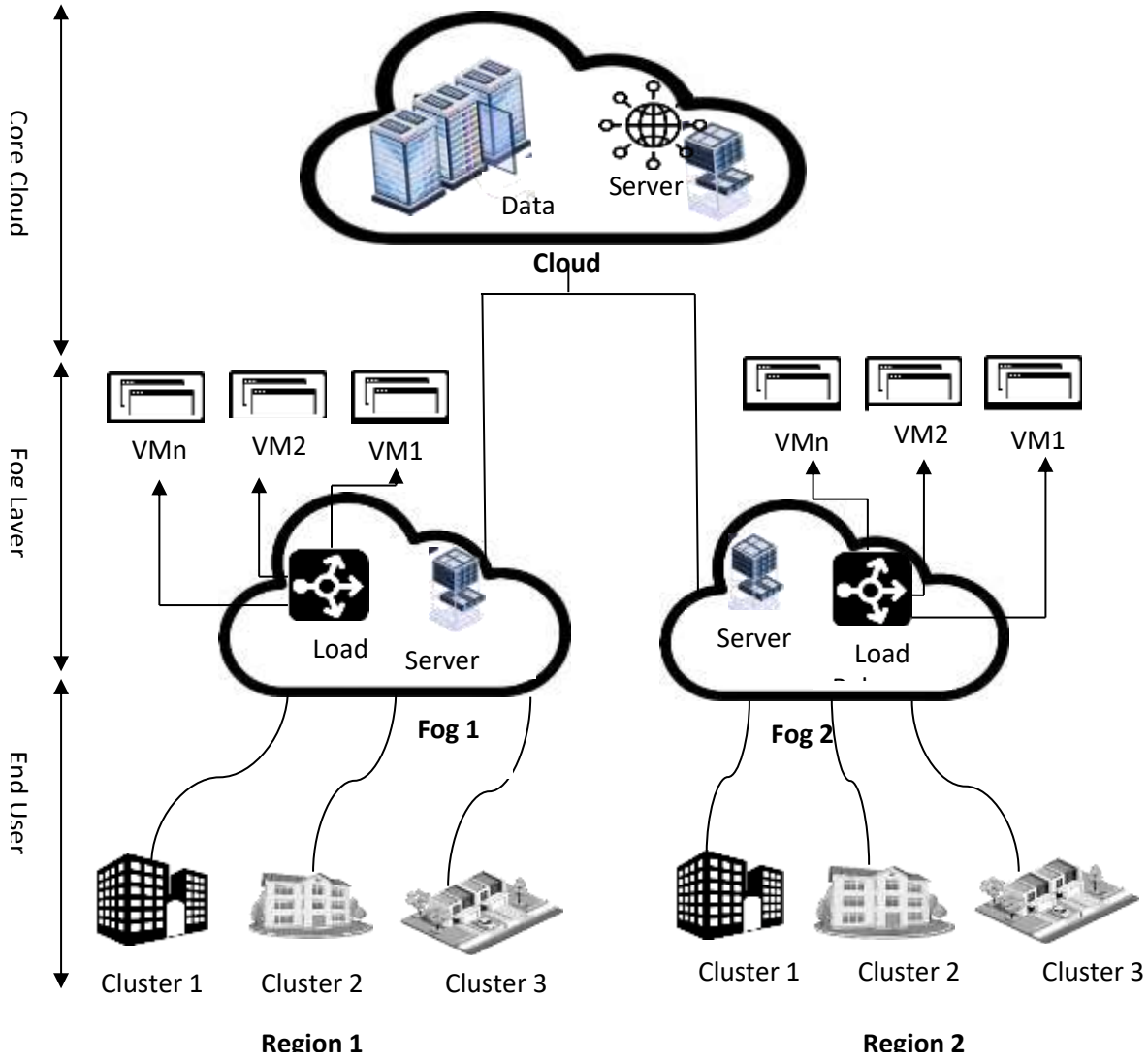

Figure 1: System Architecture

3.1 DCABA algorithm for LB

To solve an optimization issue, there are two primary families of techniques. Complete procedures that promise to either discover a legitimate allocation of values to parameters or demonstrate the absence of such an assignment. These techniques typically function well and ensure a precise and advantageous response for all inputs. However, in the worst scenario, they need exponential time, which is unacceptable in the fog computing field. The other incomplete approaches might not ensure accurate results for every input. Instead, these strategies find assignments that are satisfying for problems that are likely to be solved. Due to their efficiency in tackling specific sorts of issues, simplicity, and speed, these methods have grown in prominence in recent years. DCABA, a Hill Climbing method version, is one of the unfinished solutions to such optimization issues. Simply said, a stochastic and local optimization method is a loop that iteratively travels uphill in the direction of increasing quality. When it hits a "peak" where no neighbor's score is greater, it pauses.

This variation picks an uphill move at random, and the likelihood of selection varies with the uphill move's steepness. Therefore, by making slight adjustments to the original assignment, it maps responsibilities to a collection of assignments. Each component of the set is assessed in accordance with a set of standards intended to bring the state's assessment score nearer to that of a valid allocation. The next assignment is given to the group's top-performing component. Until a solution is discovered or a stopping criterion is met, this fundamental procedure was continued. Therefore, it consists of two key parts: a candidate creator that links a candidate solution to a set of potential replacements, and an assessment procedure that evaluates each viable solution so that improving the evaluation results in better solutions. The following is an algorithmic description of the proposed technique:

Algorithm 1: DCABA

Step 1: Keep track of the busy/available status of the VM and its index table. All VMs are accessible right away.

Step 2: In the mist, a new job appears.

Step 3: Produce a query for the following assignment.

Step 4: arbitrarily create a VM id.

Step 5: Evaluation the allocation table to determine the current status of the specific VM. If the VM is discovered to be vacant:

1. Provide the VM id.
2. Deliver the message to the VM indicated by that id.
3. Adjust the allocation table as necessary. After discovering that the VM is allocated:
4. Create an arbitrary VM using an arbitrary method.

1. A VM should be chosen for the job with an increased likelihood that it will be able to do it successfully.
2. Keep track of the VM's effectiveness; if it doesn't perform as expected, reduce the likelihood that it will be assigned in the following round.

3. Adjust the choice table as necessary.

Step 6: When the reply foglet is received and the VM has finished evaluating the request. Create a VM de-allocation alert.

Step 7: Proceed to Step 2 for the following allotment.

3.2 Problem Formulation

Our method is for a mobile-edge-fog system with multiple users, multiple tasks, and multiple tiers, to reduce the system's total power consumption and computation time. Consequently, the following is the formulation of our optimization issue:

$$\min_{\alpha}\left[\sum_{x=1}^{N}\sum_{y=1}^{M}H_{x,y}\right] \quad (1)$$

$$s.t \begin{matrix} [e_{x,y} - e^{l}_{x,y}] \leq 0 & \forall_k \in [1 \ldots k]\ c1 \\ [t_{x,y} - t^{l}_{x,y}] \leq 0 & \forall_k \in [1 \ldots k]\ c2 \end{matrix} \quad (2)$$

$$\sum_{x=1}^{N} \sum_{y=1}^{M} \sum_{\varphi=1}^{k+1} \alpha_{x,y,\varphi} f_x^e \leq F_k \qquad \forall_k \in [1 \ldots k]\ c3 \quad (3)$$

$$\sum_{\varphi=1}^{k+1} \alpha_{x,y,\varphi} = 1 \qquad \forall_{x,y} \qquad c4 \quad (4)$$

$$\alpha_{x,y,\varphi} \in \{0,1\} \qquad \forall_{x,y} \qquad c5 \quad (5)$$

An issue could be viewed as a single optimization task that attempts to use offloading and safety implementation to reduce the cost of the system in terms of energy and time. The first two limitations correspond to the maximum amount of energy and time that could be consumed.

The disparity in the arrangement of stations on stations could be attributed to the fact that certain platforms are overloaded while others are underloaded. As a result, lengthy delays and poor service quality due to network congestion reassign the load among the stations by obliging them to transfer to the greatest available locations. Every station first uploads a summary of its locations, which contains the number of attached stations, the transmitted information rate, the number of CPU cycles allocated to each MUD job, or the number of stations that could be reallocated to other locations nearby. The central control manager then cycles through the stations and forces them to pass off to the best accessible stations depending on the number of users who are currently available and the calculation time and information rate.

VMs make up fog resources. Requests are distributed VMs to handle. RT, cost, and processing time are the three primary performance metrics for FC. In allocating VMs appropriately, load scheduling methods are employed to optimize the workload of fog. Let N be the number of customers per group N = $\{n_1, n_2, \ldots, n_n\}$ and C be the number of groups in the framework C = $\{c_1, c_2, \ldots, c_c\}$. A client makes R number of requests every hour, where R = $\{r_1, r_2, \ldots, r_m\}$. VM = $\{vm_1, vm_2, \ldots, vm_y\}$ was a representation of a set of VMs. Requests are directed to fog, which uses LB to allocate VMs to them. Following are the total requests made by an N-user group:

$R_{Cluster} = \sum_{x=1}^{M}(r_x)$ (6)

And a total request:

$R_{Total} = \sum_{cluster=1}^{c}(R_{cluster})$ (7)

Performance metrics are impactedR_{Total} is mapped to the number of VMs j, where a high volume of requests causes delays.

Processing Time Start time less the final duration equals the processing period Tp.

$T_p = T_{start} - T_{end}$ (8)

R request's processing time at vm_j is the time allocated to assigning R request to j

$P_{rj} = \frac{Length\ of\ r}{vm_j}$ (9)

In this case, the position of the request r at vm_j is either provided or not, according to λ_{yx}. When the total number of allotted VMs is known, the conflict for constraint implies that no VMs will be issued. Conflict decrease increases our processing speed. The algorithm's goal is to reduce processing time:

$Min_{T_p} = \sum_{y=1}^{j}\sum_{x=1}^{r}(\lambda_{yx} * P_{yx})$ (10)

Response Time (RT) was the total latency and demand processing time.

RT = $T_p + T_d$ (11)

T_d represents the delay experienced when the request makes it to the fog.

3.3 MCS-EEAO algorithm for LB

The proposed MCS-EEAO algorithm is discussed in this section. Aquila is one of the maximum desired birds in the world because of its hunting bravery. When the male Aquilas hunted alone, they captured a great deal more target. With their quickness and energetic greenhouses, Aquila chases squirrels, rabbits, and a variety of other things. It is even identified as a threat to the adult deer. Ground squirrels are reportedly the second most important species in Aquila's diet. For bird hunting in flight, the first technique, strong flight with a horizontal person was used in which the Aquila rises above the ground. The Aquila begins a long flight at a shallow angle after finding prey with speed increasing as the wings close. For that approach to work, Aquila needs to have a vertical advantage over her target. To simulate thunder, the wings and tail were deployed just before the confrontation, and the feet were propelled forward to capture the prey. The second approach, bypass flight with a brief gliding attack, was considered the most common technique used by Aquila, where the Aquila climbs from the floor at a reduced rate. Whether the prey travels or flies, the prey is pursued with care. For shooting ground squirrels, breeding grouse, or seagulls, this strategy was ideal. The procedures for carrying out our proposed LB method are shown in Algorithm 2.

First Method Second Method

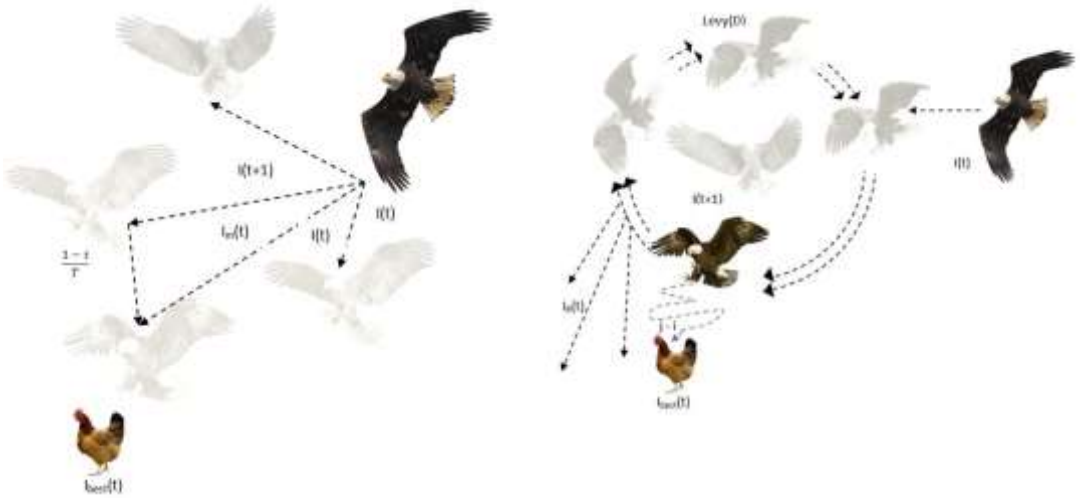

Figure 2: Eagle Aquila hunting methods

MCS-EEAO the optimization method begins with the generation of a random collection of possible alternatives, known as the community. Using a recurrence trajectory, the MCS-EEAO search algorithms analyze eligible investments for an almost optimal response/the best response. The MCS-EEAO optimization techniques update each optimizer location with the best option. Four separate research tactics for exploration and exploitation have been proposed to illustrate the balance between MCS-EEAO research strategies. Once the final criterion is met, the MCS-EEAO search process ends.

Algorithm 2: MCS-EEAO

Input: BS is allocated with MDU

Output: Best available MDU with LB

Step 1: Population P initialized and parameters α, δ

Step 2: do-while

Determine fitness function values $P_{best}(r)$

Identify the best solution achieved

Step 3: for x ranges from 1 to N

The Average value of the present solution $P_m(r)$ is updated

Modify the value x, y, H_1, H_2, Levy(R)

if $r \leq \left(\frac{2}{3}\right)$ then

if random ≤0.5 then

Expanding (P_1)

The Current solution is updated by using Equation 4.

if Fitness (P_1 (r+1)) < Fitness (P(r)) then

P(r) = (P_1(r+1))

if Fitness (P_1 (r+1)) < Fitness (P_{best}(r)) then

P_{best} (r) = P_1(r+1)

end

end

else

Step 4: Narrow (P_2))

The Current solution is updated by using Equation 5

if Fitness ($P_2(r+1)$) < Fitness ($P(r)$) then

$P(r) = (P_2(r+1))$

if Fitness ($P_2(r+1)$) < Fitness ($P_{best}(r)$) then

$P_{best}(r) = P_2(r+1)$

end

end

end

end

Step 5: Start the EEOA **Proceduer** and set the input (a,P(r), $P_{best}(r)$)

Step 6: Set X =0 and Y=0

Step 7: for (a,t) ϵ $P_{best}(r)$ union do

set the VECTORIZE (a,t) to X append u(a,b) to Y

Set the gp= Gaussian Process (X,Y)

Є*, δ* = POSTERIOR(gp,l*)

end for

Step 8: for all BS at time t do

Assign a =Number of MDU to BS.

Assign b = data rate and consumptions uplinked for each MDU at BS

Assign c = reallocated all the MDUs to nearby BS

Step 9: Hand over the best available BS to LB based on a,b,c

Step 10: return the value

4. Experimental Results

The proposed planning technique is depending on DCABA and employs it to optimize how requests are handled in the fog network. Another element of the suggested approach is to balance the load in the fog system. The proposed methodology is described in the sections above. By using FogSim tool, map the experimental evaluation of the proposed planning approach.

The time required to run an efficient schedule queue serves as the primary principle for the evaluation of the planned scheduling approach over a fog network. The CPU utility rate and storage consumption rate would be the primary deciding criteria that would be taken into account. The performance of the proposed planning approaches was tested using the simulated database depending on various factors. The amount of time was determined using the amount of time needed to create an efficient planning method.

4.1 Performance Evaluation

The CPU usage rate and storage utilization rate are the two parameter variables in the recommended assessment section. Therefore, the aforementioned decision characteristics are regarded as the nodes' load. As a result, when creating a schedule by balancing the provided loads, we must take performance into account. They provide two sorts of assessments in performance evaluation. Here, they adjusted the CPU's maximum utilization to be between 60 and 100, and the analysis's findings are shown in Figure 3. The line time and the memory utilized for the same in the storage rate line serve as a representation of the time and memory required for planning for various CPU rates. The use of memory per 100 KB is used to compute the memory rate. According to the evaluation, scheduling times are consistent throughout a range of CPU speeds, and memory rates.

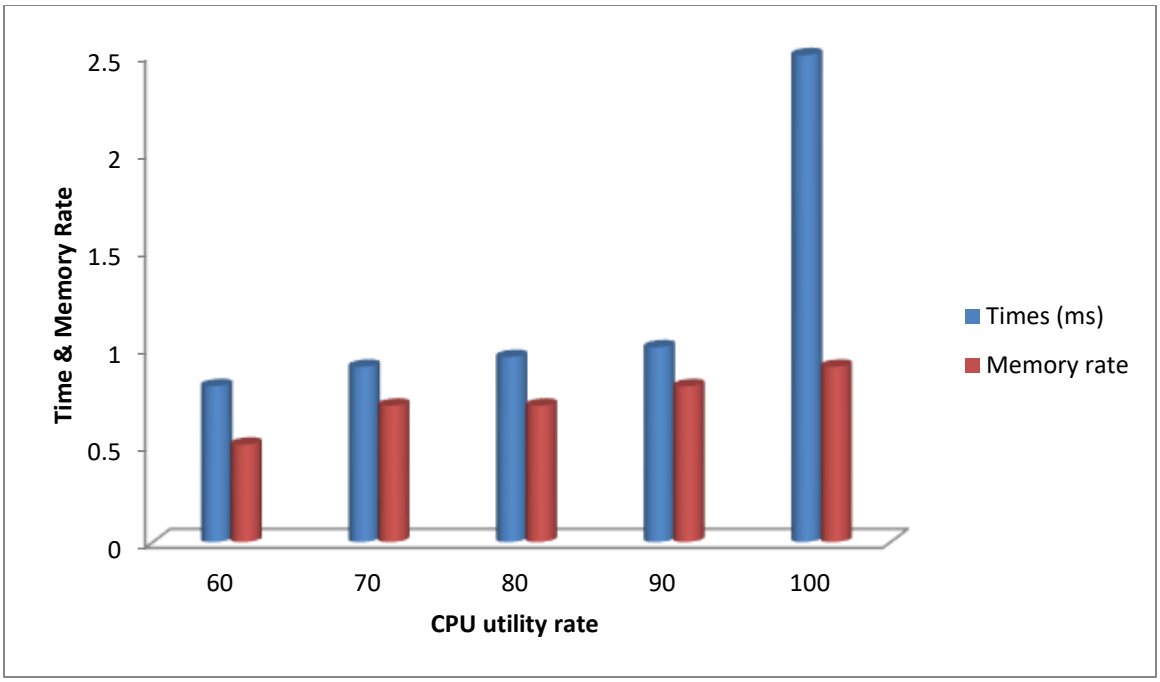

Figure 3: CPU rate-based assessment

Researchers chose a memory rate range of 60 to 100, and the evaluation's findings are displayed in Figure 4. The time and CPU rate, correspondingly indicate the time and CPU consumption for planning for various memory rates. The utilization of the CPU divided by the CPU's 100% utilization yields the CPU utilization rate. An investigation demonstrated that the planning period is consistent for varying memory and CPU rates.

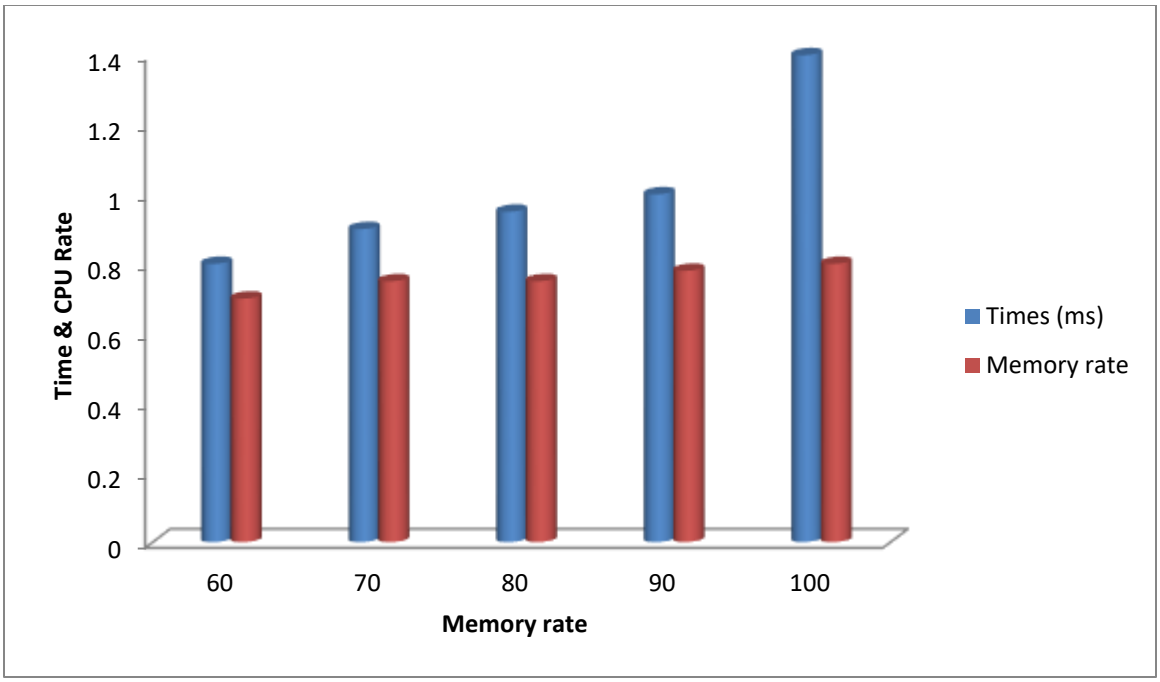

Figure 4: Memory rate based on assessment

4.2 Comparative Analysis

The contrast evaluation of the proposed method with the current methodology is shown in Figures 5 and 6. The findings of the LB planning approaches were used to determine the values of the current approach. According to the analysis of Figure 5, the proposed strategy uses the CPU rate more effectively than the current approach does during LB. The memory graph evaluation reveals that the average memory utilization for the technique is lower than the proposed strategy. Therefore, the proposed method is more efficient than the current method while taking into account a load-balanced condition and the CPU usage rate.

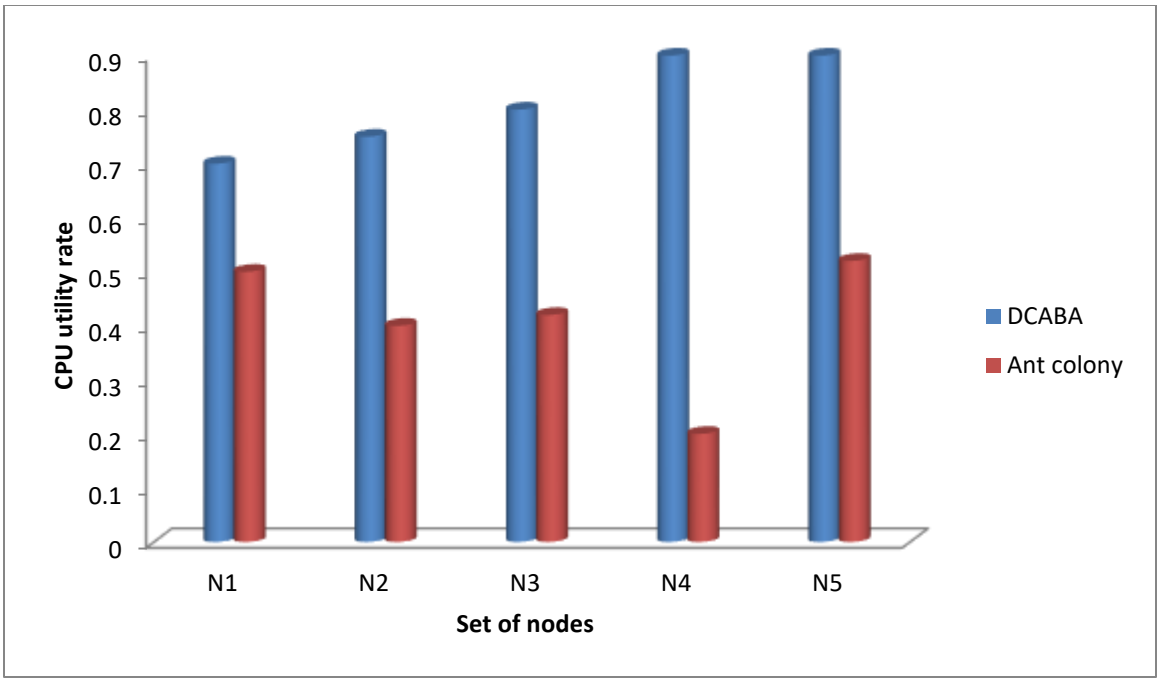

Figure 5: CPU usage rate contrast

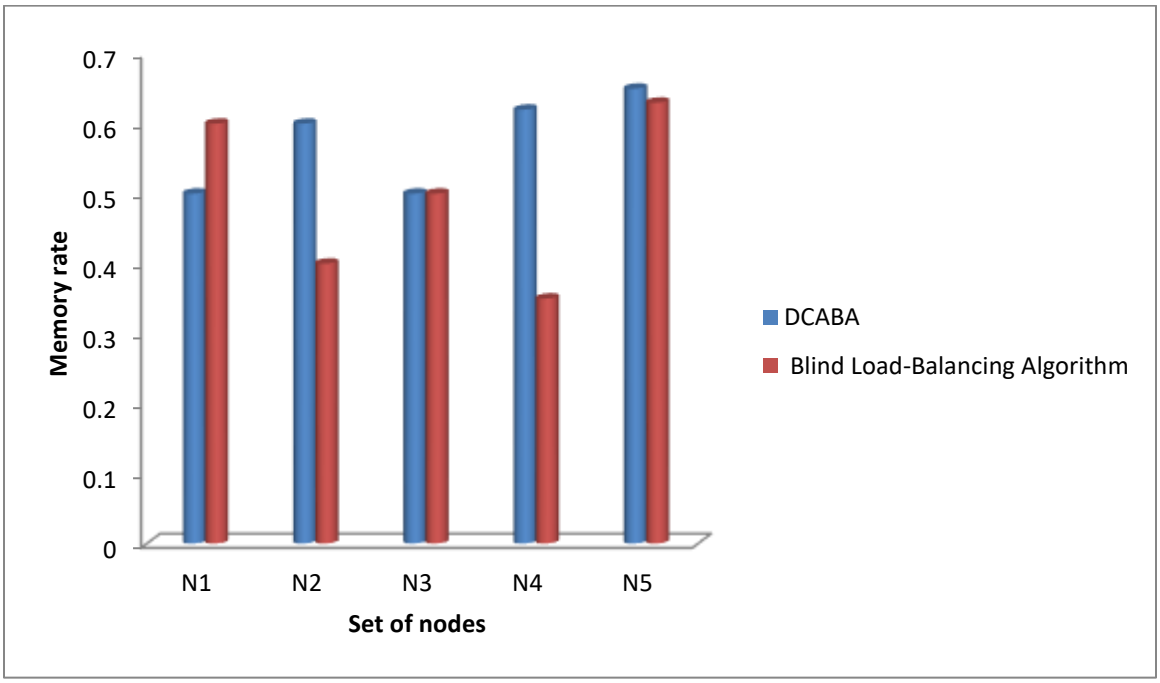

Figure 6: Memory size

4.3 Load Balance

The number of Stations assigned at each Station is shown in Figure 7 for both situations with and without the integration of LB. As can be seen from that, in the latter case, or without LB, stations 2 and 3 are overloaded while stations 1 and 4 are underloaded. As it is, our proposed method maintains a balance to the number of stations provided by each station, resulting in a more balanced load distribution and enhancing the system's efficiency.

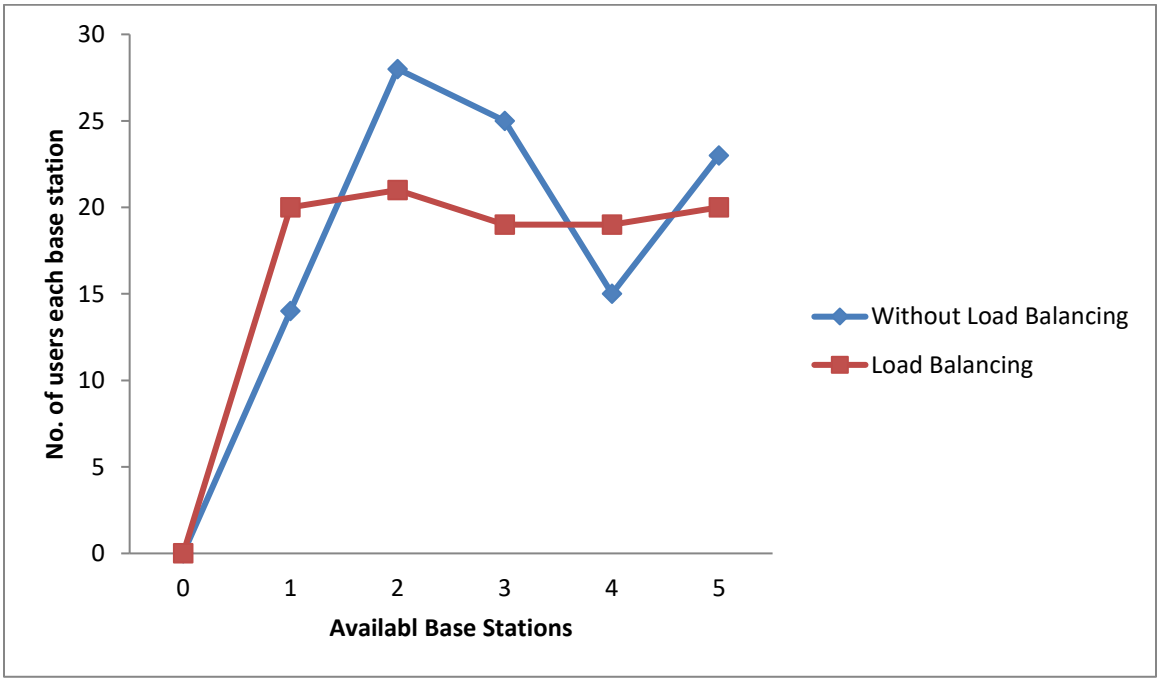

Figure 7: The locations of the users with and without LB

4.4 Simulation Results

This section presents the simulation and its outcomes. The performance variables used in the simulations are computed using formulas. Fog-analyst, which simulates the real-world environment, is used for these variable calculations. From there, they analyze a large number of tests to obtain and optimize findings for a actual-world scenario. The charts for the techniques with Min-conflicts illustrate the cost, response time, and processing time.

The total Response Time(RT) of each method is displayed in Figure 8. The overall response time in the case is 65.25 ms, round robin is 78.95 ms, and the overall RT of the proposed technique MCS-EEAO is 64.74 ms. Min-conflicts outperform the other two methods in fog computing by providing a lower RT and greater effectiveness. The performance of the MCS-EEAO technique is shown in Figure 9 for RT to each of the six fog groups.

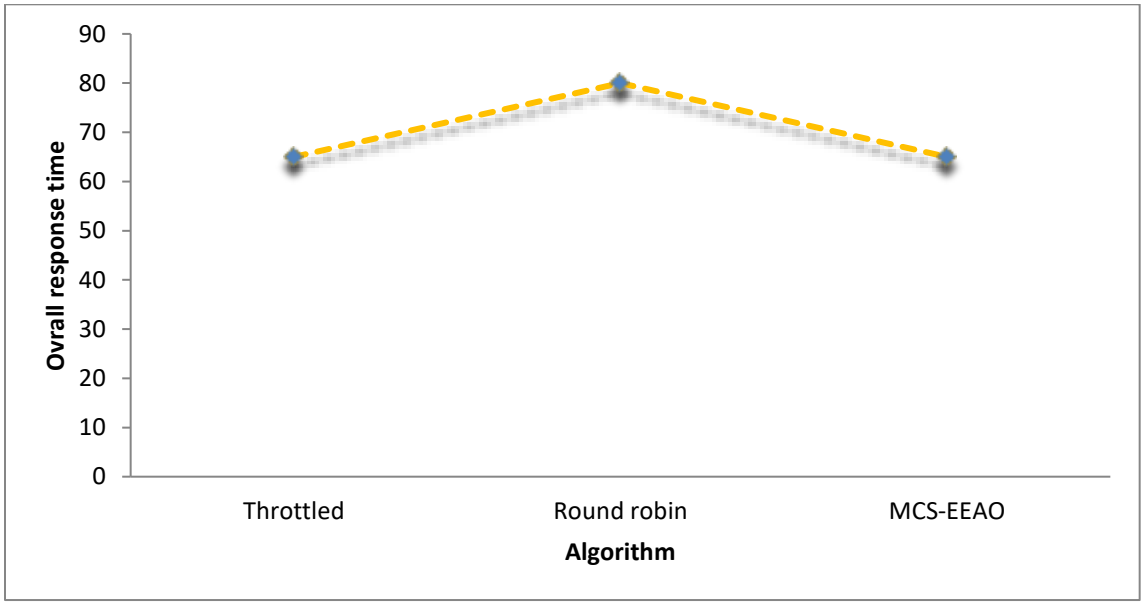

Figure 8: Overall Response Time of Fogs

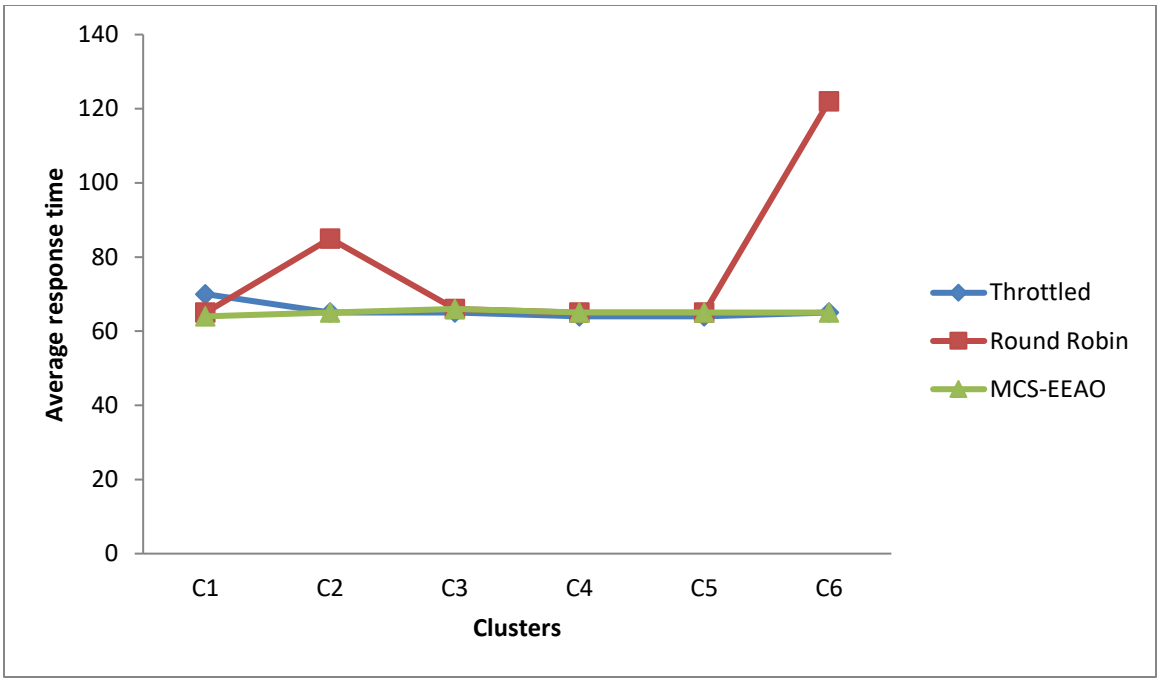

Figure 9: Average RT of Fogs

The processing times for throttled, RR, and min-conflicts are 15.60 ms, 28.24 ms, and 15.14 ms, correspondingly, in Figure 10. MCS-EEAO operates better than the other two methods since they demand less processing time. The processing time of fogs is shown in Figure 11.

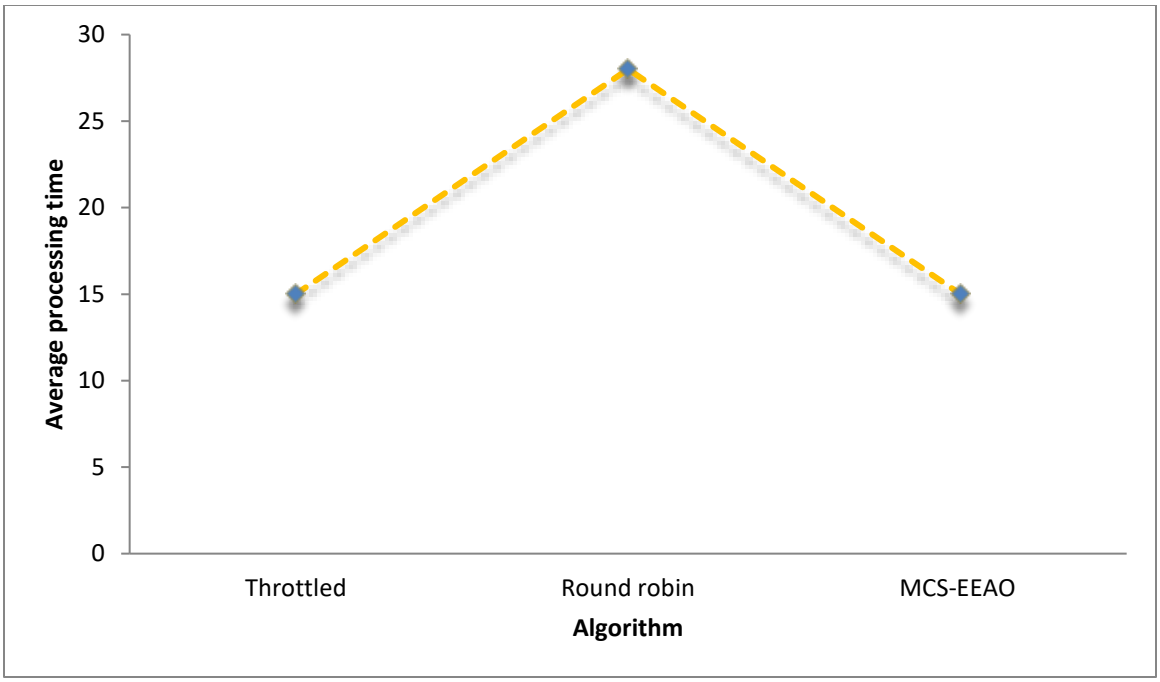

Figure 10: Overall Processing Time

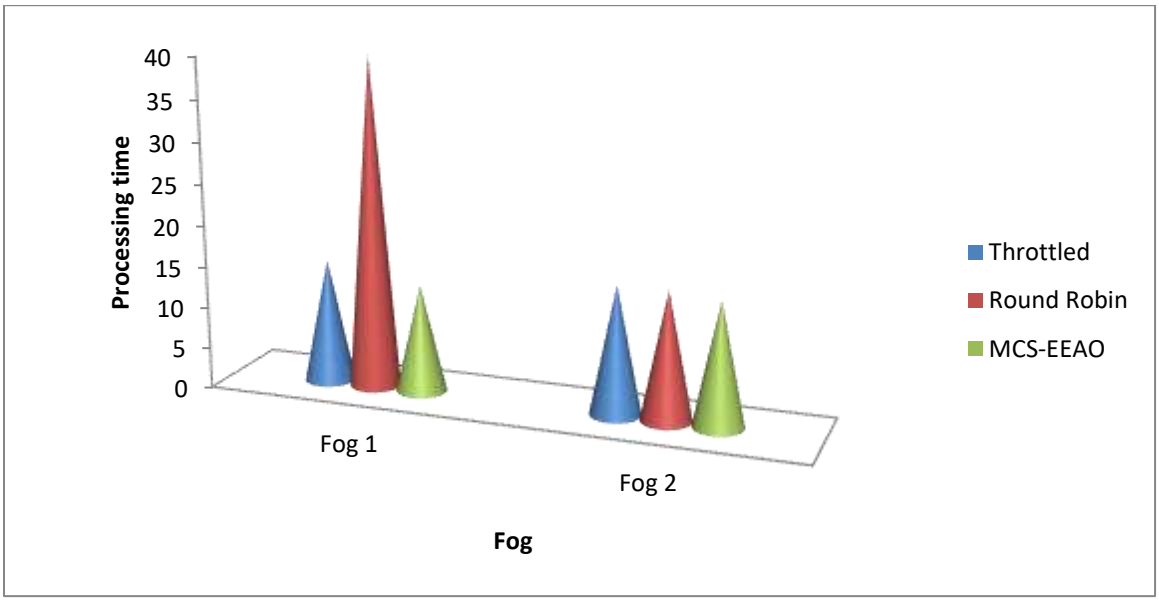

Figure 11: Fogs' processing time

In the described situation, a cloud-fog based technology can be utilized. The cloud, fog, and end-user levels make up the three tiers of the recommended framework. For the simulation, which takes into account

the two areas of Asia and Europe, the fig sim tool is used. In this case, three building groups are linked to one fog that is put in one area. With cloud and energy provider infrastructure, the fog communicates. Allocating VMs to the requests uses the MCS-EEAO LB technique. Compared to RR and Throttled, this approach offers better simulated outcomes for RT and processing time.

5. Conclusion

To achieve demonstrable gains in server workload control and lower the cost of fog services, hence this work demonstrated the practicality of LB and server consolidation strategies. Fog vendors can benefit from enhanced cost benefits because of the proposed dynamic criterion dependent on DCABA. Additionally, a separate portion for server consolidation and LB improves the method's scalability. A proposed strategy is implemented in three stages: first, an individual'is derived from the fog network, followed by planning index estimation, and ultimately, the planning list is optimized using the DCABA methodology. The same generated fog network is used for the experiments, and the efficiency of the proposed strategy is assessed. The results of the study delivered the projected outcomes, demonstrating the effectiveness of the MCS-EEAO for optimizing schedules by balancing the loads and reduces the overall response time.

Declaration:

- We confirm that we have read, understand, and agreed to the submission guidelines, policies, and submission declaration of the journal.

- We confirm that all authors of the manuscript have no conflict of interests to declare.

Ethical Approval :

Not applicable for this work.

Competing Interest:

We declare that have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding:

There is no funding body for this study.

Availability of data and materials:

Data sharing not applicable to this article as no datasets were generated or analysed during the current study

References

[1] Kashani, M. H., and Mahdipour, E. (2022). LB algorithms in fog computing: A systematic review. IEEE Transactions on Services Computing, (01), 1-1.

[2] Batra, S., Anand, D., and Singh, A. (2022, April). A Brief Overview of LB Techniques in Fog Computing Environment. In 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 886-891). IEEE.

[3] Albalawi, M., Alkayal, E., Barnawi, A., and Boulares, M. (2022). LB Based on Many-objective Particle Swarm Optimization Algorithm with Support Vector Regression in Fog Computing. Journal of Engineering and Applied Sciences Technology. SRC/JEAST-170. DOI: doi. org/10.47363/JEAST/2022 (4), 138.

[4] Potu, N., Bhukya, S., Jatoth, C., and Parvataneni, P. (2022). Quality-aware energy efficient scheduling model for fog computing comprised IoT network. Computers and Electrical Engineering, 97, 107603.

[5] Gupta, S., and Singh, N. (2023, January). Resource Management with LB Strategies in Fog-IoT Computing Environment: Trends, Challenges and Future directions. In 2023 International Conference on Artificial Intelligence and Smart Communication (AISC) (pp. 1358-1359). IEEE.

[6] Malik, S., Gupta, K., Gupta, D., Singh, A., Ibrahim, M., Ortega-Mansilla, A., ... and Hamam, H. (2022). Intelligent load-balancing framework for fog-enabled communication in healthcare. Electronics, 11(4), 566.

[7] Abohamama, A. S., El-Ghamry, A., and Hamouda, E. (2022). Real-time task scheduling algorithm for iot-based applications in the cloud–fog environment. Journal of Network and Systems Management, 30(4), 54.

[8] Mirtaheri, S. L., Azari, M., Greco, S., and Arianian, E. (2023). An Ant-colony Based Model for LB in Fog Environments. Supercomputing Frontiers and Innovations, 10(1), 4-20.

[9] Verma, R., and Chandra, S. (2023). HBI-LB: A Dependable Fault-Tolerant LB Approach for Fog based Internet-of-Things Environment. The Journal of Supercomputing, 79(4), 3731-3749.

[10] Malik, S., Gupta, K., Gupta, D., Singh, A., Ibrahim, M., Ortega-Mansilla, A., ... and Hamam, H. (2022). Intelligent Load-Balancing Framework for Fog-Enabled Communication in Healthcare. Electronics 2022, 11, 566.

[11] Kanellopoulos, D., and Sharma, V. K. (2022). Dynamic LB Techniques in the IoT: A Review. Symmetry, 14(12), 2554.

[12] Kesavan, R., Loganathan, V., Shankar, T., and Periasamy, J. K. (2022). Fog-computing: a novel approach for cloud-based devices using perceptual cloning manifestation-PerColNif taxonomy by energy optimization. Energy conservation solutions for fog-edge computing paradigms, 107-128.

[13] Kesavan, R., Poorani, S., Iyswarya, R., Muthunagai, S. U., Anitha, R., and Vijayaraja, L. (2023). Convergence Perceptual Model for Computing Time Series Data on Fog Environment. In Computer Vision and Machine Intelligence Paradigms for SDGs: Select Proceedings of ICRTAC-CVMIP 2021 (pp. 15-23). Singapore: Springer Nature Singapore.

[14] Liu, S., Yang, S., Zhang, H., and Wu, W. (2023). A Federated Learning and Deep Reinforcement Learning-Based Method with Two Types of Agents for Computation Offload. Sensors, 23(4), 2243.

[15] Songhorabadi, M., Rahimi, M., MoghadamFarid, A., and Kashani, M. H. (2023). Fog computing approaches in IoT-enabled smart cities. Journal of Network and Computer Applications, 211, 103557.

[16] Gowri, V., and B. Baranidharan.(2022)."Dynamic Energy Efficient Load Balancing Approach in Fog Computing Environment." In Intelligent Communication Technologies and Virtual Mobile Networks: Proceedings of ICICV pp. 145-160. Springer Nature Singapore, 2022.

[17] Wen, W., Demirbaga, U., Singh, A., Jindal, A., Batth, R. S., Zhang, P., and Aujla, G. S. (2023). Health Monitoring and Diagnosis for Geo-Distributed Edge Ecosystem in Smart City. IEEE Internet of Things Journal.

[18] Abkenar, S. B., Kashani, M. H., Akbari, M., and Mahdipour, E. (2023). Learning textual features for Twitter spam detection: A systematic literature review. Expert Systems with Applications, 120366.

[19] Tahmasebi-Pouya, N., Sarram, M. A., and Mostafavi, S. (2022). A Blind Load-Balancing Algorithm (BLBA) for Distributing Tasks in Fog Nodes. Wireless Communications and Mobile Computing, 2022.

[20] Arefian, Z., Khayyambashi, M. R., and Movahhedinia, N. (2023). Delay reduction in MTC using SDN based offloading in Fog computing. Plos one, 18(5), e0286483.

[21] Janakiraman, S., and Priya, M. D. (2023). Hybrid grey wolf and improved particle swarm optimization with adaptive intertial weight-based multi-dimensional learning strategy for LB in cloud environments. Sustainable Computing: Informatics and Systems, 38, 100875.

[22] Kanellopoulos, D., and Sharma, V. K. (2022). Dynamic LB Techniques in the IoT: A Review. Symmetry, 14(12), 2554.

[23] Fahimullah, M., Ahvar, S., and Trocan, M. (2022). A Review of Resource Management in Fog Computing: Machine Learning Perspective. arXiv preprint arXiv:2209.03066.

[24] Ezhilarasi, T. P., Dilip, G., Latchoumi, T. P., and Balamurugan, K. (2020). UIP—a smart web application to manage network environments. In Proceedings of the Third International Conference on Computational Intelligence and Informatics: ICCII 2018 (pp. 97-108). Springer Singapore.

[25] Alirezazadeh, S., and Alexandre, L. A. (2023). Ordered balancing: LB for redundant task scheduling in robotic network cloud systems. Cluster Computing, 1-16.

[26] Mattia, G. P., Pietrabissa, A., and Beraldi, R. (2023). A LB Algorithm for Equalising Latency across Fog or Edge Computing Nodes. IEEE Transactions on Services Computing.

[27] Tripathi, G., Singh, V. K., and Chaurasia, B. K. (2023). An energy-efficient heterogeneous data gathering for sensor-based internet of things. Multimedia Tools and Applications, 1-24.