Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7766351

OVERCOME LOAD BALANCING PROBLEM BY WEIGHT BASED SCHEME ON CLOUD

Nisha Kumari Uparosiya[1], Mr. Ajay Kumar[2]
[1]M.tech., Research Scholar, Department of Computer Science& Engineering,JECRCUniversity Jaipur, Rajasthan
[2]Assistant Professor, Department of Computer Science& Engineering, JECRC University Jaipur, Rajasthan

Abstract *The use of cloud computing technology may allow hosts on the internet to have their needs met. Cloud computing may be used to share or consume the resource. Instead of creating or maintaining additional organisational infrastructure, a resource like a virtual machine might be utilised for storage or as an application. Via a pool of shared resources, including servers, storage applications, and other resources, the cloud computing architecture offers on-demand access to resources. As these resources are readily available and distributable, many individuals may get services. In order to provide these users with complete access to resources, the businesses ensure that they have access to the cloud computing capabilities linked to the storage applications. The movement of virtual computers, which happens anytime there is network uncertainty, is the major issue with cloud computing. Heavy resource utilisation causes the virtual machine to become overloaded, which increases how long cloudlets take to execute. The threshold strategy, which distributes work to the most competent machine and hosts maintain checkpoints on the virtual machines, is shown on the base page. The burden must be distributed to another virtual machine when one is overworked.In this research attempt, a weight-based method of moving cloudlets from one virtual machine to another will be recommended.*

Keywords: ***Cloud Computing, Load Balancing,Genetic Algorithm, virtual machine***

I INTRODUCTION

Cloud computing may make it easier for users to access services and share hardware resources, which can prove to be highly advantageous for consumers financially. Mobile cloud technology has become more common due to the increased usage of mobile devices. Mobile users want a variety of applications on their smartphones, yet they need cloud services to perform storage and computation due to the constrained mobile resources (battery life, storage, and processing). Mobile data is offloaded to the cloud for remote processing due to resource limitations, which is then finished and transmitted back to the mobile device.

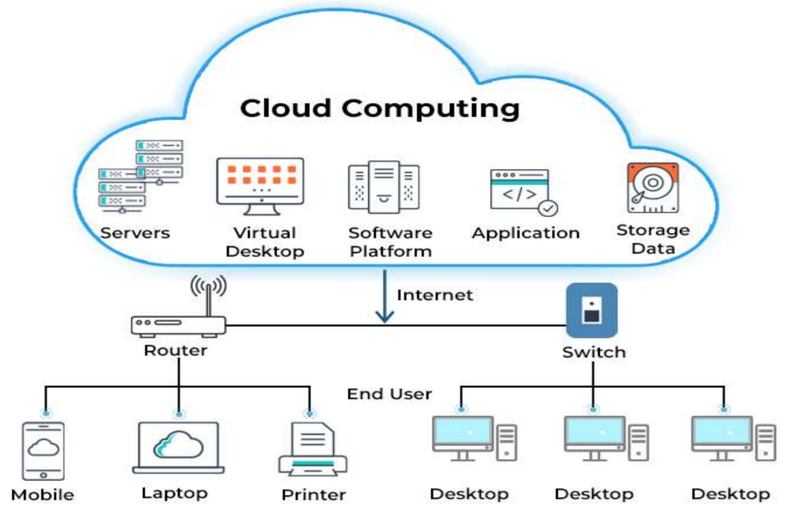

Figure 1 Cloud Computing Environment

1.1 Load balancing

A process called load balancing splits the total network load and distributes it across the system's different resources. As a consequence, any job's reaction time and resource usage efficiency both increase. This approach also prevents the situation when a small number of hubs are overloaded and a small number are under loaded. By spreading the load among several components, redundancy makes it feasible to improve dependability and data availability. Depending on memory CPU load, and network load, the system gauges load.

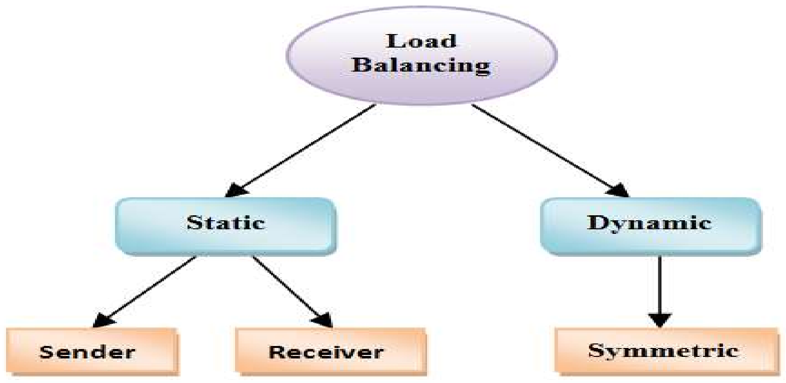

Figure 2 Static and Dynamic Load balancing

1.2 Existing Load Balancing Algorithms

Many load balancing techniques have been suggested by different scholars throughout the years. Here are some of such algorithms, explained:

- **Round Robin**: This algorithm helps in allocating the time to individual process periodically with similarity to the processor's context switching. Therefore, for all the processors, equal processor time is provided. Cloud computing applies round-robin to allocate equal amount of load across different modes. In all nodes that have similar efficiency equal amount of load does not exist since the capacities of nodes vary. The effectiveness of this algorithm is high when all nodes have equal capability. For HTTP request, this method is used more often since all these requests are very alike.
- **Weighted Round Robin:** To improve round robin algorithm, this algorithm is introduced. And it is in charge of eliminating the issues that existed with earlier conventional algorithms. Each node is given a weight, and based on the weights allocated, the appropriate amount of requests are made. To each instance the load is assigned depending upon the processing capability. Based on the behaviour of instance, it is important to consider this factor. Each server existing in the surroundings is allocated through weight. A weight is assigned to the notifying server when any server is killed. IP balancer is attached above two powerful servers based on two requests. It is possible to assign the other two servers with all the requests. When identical weights are given to each node, each is in charge of receiving traffic.
- **Central Queuing:** The dynamic load balancing maintains central algorithm to manage all of the activities queued on VM at the central manager. When an activity is discovered to be in the ready state, the manager utilises that activity to process. If not, the activity is ignored until a suitable resource in the VM is found for it.
- **Min-Min Algorithm:** To perform load balancing, this popular scheduling algorithm is applied effectively. To all the incomplete tasks, a completion time is assigned and these tasks are ordered based it. Here, the job that takes the least amount of time to complete is picked and processed. Further, it is possible to recalculate the completion time. Until no tasks are The procedure continues repeating if you are left in the line.
- **Honey Bee Foraging Algorithm:** This algorithm was created in accordance with honey bee behaviour. Here, both the jobs and the honey bees are compared. The reliable sources are regarded as VM in this case. Bees, which are sometimes depicted in this context as tasks, are prioritised by the beehive. All of the virtual machines have been sorted, and it is considered that the priority corresponds to the capacity of the jobs in these resources. The VM with the highest capability is given the task. This approach allows for bidirectional communication. As a job is finished, data is transmitted back to the beehive, where the algorithm selects the subsequent priority task. The process continues until every task is finished. [14].
- **Genetic Algorithm:** In the soft computing mechanism, GA is implemented within dynamic scenarios. When applying GA better outcomes are achieved in comparison to other algorithms like Round Robin. GA helps in performing complex objective function by applying improved search space. This algorithm avoids getting trapped in local optimal solution.

II Literature Review

Several load balancing and scheduling approaches have been suggested by many academics [12]. Due to the large number of processors and the many similar or unique instances of virtual machines that each one of them contains, scheduling tasks in a mobile cloud environment is challenging (VM).A fundamental challenge in the mobile cloud environment is allocating workloads to the appropriate VM while enhancing performance and cost. To accomplish load balance and reduce the makespan, job scheduling should take this into account as well [8]. A few heritage scheduling and load balancing techniques have been utilised extensively in several works. For instance, the Min-Min algorithm chose the task with short tasks [5] and was better since it reduced the makespan while balancing the load.Scheduling is often done depending on the size of the jobs. While load balancing and an improved min-min scheduling technique [6] had been suggested, execution cost was not taken into account in this research. It demonstrated a rise in resource use and a fall in makespan. After picking the job, the max-min algorithm selected the task with the largest size [9]. Priority-based scheduling has been the subject of several studies [7] that aim to execute equitable scheduling and prioritise the tasks. Nevertheless, the complicated, unpredictable, and real-time consumer demand was too much for these outdated algorithms to handle. These studies didn't take utility-based prioritising into account.

Poonam and Suman Sangwan (2023) In this study, a smart, fuzzy firefly-based method for mobile cloud computing load balancing with shorter makespan is introduced. The recommended approach is brilliant because of its fuzzy and firefly properties. It automatically converges on the target function and search results. The three-tier architecture of public clouds and cloudlets functions. Fuzzy logic chooses cloudlets with limited resources based on capacity and waiting time. Without using arithmetic, fuzzy makes decisions that resemble those of a person. Firefly is optimised for speed and diversity. It lessens the load on the cloud and execution time. Locally ideal traps may be managed by levy flying. Fuzzy fire fly-levy flight hybridization balances load by reducing makespan, execution time, and imbalance. NASA and Clarknet datasets were simulated using Cloud Analyst. When it comes to makespan, degree of imbalance, and figure of merit, the proposed technique surpasses ACOQDM, DSOA, and UFA.

Kanellopoulos, D.; Sharma, V.K. (2022) The Internet of Things enables smart devices to observe, interact, and perceive the real world as well as to communicate with one another (IoT). Sensors, actuators, smart devices, cameras, protocols, and cloud services enable intelligent IoT applications such as environmental monitoring, traffic monitoring, remote patient monitoring, security surveillance, and smart home automation. For the IoT network to be utilized to its full potential, issues with energy constraints, scalability, dependability, heterogeneity, security, privacy, routing, QoS, and congestion must be fixed.Effective load balancing (LB) spreads traffic loads among pathways to reduce traffic congestion in the IoT.In this paper, edge-fog-cloud networking principles are discussed in relation to IoT designs. It then contrasts IoT LB surveys. It distinguishes between edge/fog and IoT cloud dynamic LB techniques. Lessons and research questions are provided at the end.

Li, Y.; Li, J.; Sun, Y.; Li, H. (2022) The broad use of computational fluid dynamics in many fields, as well as the growing complexity and size of the computational grid, need the employment of large-scale parallel computing to meet this challenge. In multi-block grid numerical simulations, load balancing and communication overhead are impacted by the grid block-to-processor mapping technique. Load-balancing issues are resolved utilising graph network dynamic programming and a multi-level graph partitioning strategy. In order to weighted combine the firefly and ant colony optimization processes, The firefly-ant compound optimization (FaCO) method was suggested in this study. Graph results become a travelling salesman problem after multi-level graph partitioning (TSP). The solution load distribution is optimized prior to the projection of a preliminary graph segmentation in order to get the most cutting-edge segmentation optimization results utilizing this technique. TSP issues are often solved using the firefly algorithm (FA) and ant colony optimization (ACO), despite the fact that both methods are susceptible to local optimization and provide inconsistent search results.This is made better by the FaCO approach by updating the location and changing the weight of iterative site selection. Public datasets like Oliver30 and eil51 were compatible with the FaCO approach. It outperforms the firefly approach and others in search results, efficiency, and parallel computing load-balancing optimization.

Xiang Wang and Ying Zhang (2022) Human resource dispersion has an effect on how people are used, which has an effect on how mobile businesses can be and how productive workers are. This study presents a PSO-based model for corporate HR optimal allocation. A scale prediction, structure analysis, and implementation strategy are provided for the best HR allocation in enterprises. This issue is resolved by developing a more efficient PSO, and a model for optimizing HR configuration is developed, starting with operability, on the basis of system analysis and quantitative evaluation. Numerical simulations demonstrate how successful this approach is. Experiments show that the modified PSO has a fast convergence rate and a 5% lower average error rate than the conventional method.This strategy has a 94% accuracy rate. This strategy seeks to maximise HR settings.

Santhosh Kumar Gorva and Latha ChannagiriAnandachar(2021) More concerns concerning the security and energy requirements of cloud data centers have been brought up by the recent rapid growth of cloud computing. In this study, a hybrid paradigm for load balancing and data security is proposed.To balance tasks across heavy-loaded and light-loaded VMs, a Modified Particle Swarm Optimization (MPSO) approach is first given. Website, database, application, and other computing resources are improved via load balancing. A linear decreasing inertia weight in the MPSO optimises VM migrations and load balancing. Second, an EECC approach is recommended for data security. The EECC approach employs a fresh public key and pseudo-random key to secure cloud data centres. The SLA, execution time, energy consumption, energy SLA violation, encryption, and decryption comparison time of the MPSO-EECC model are all validated by this research. The experimental part showed that the MPSO strategy reduced energy consumption by 30% to 70% when compared to conventional optimization strategies.Comparative cryptography technique decryption time was reduced by the EECC algorithm from 10 to 30 microseconds.

Monesh Kumar and Prof. Devendra Singh Rathore (2020) This study uses SDN to create purposeful active allocators for cloud data centre virtual machines (VMs) (DC). CPU, RAM, disc, and memory make up VM requests. In private cloud architecture, virtual machine allocation is an issue. This environment assigns virtual machines to physical hosts depending on host resources. Quantifying the performance of cloud infrastructure scheduling and allocation algorithms for diverse applications and service models is difficult due to varying performance metrics and system requirements.

III Research Methodology

In this research, problems in the cloud network brought on by node failure are addressed using a BFO algorithm. The proposed method has a huge number of nodes. A candidate node is selected among the nodes based on the failure rate and quickest execution time. When utilising a master node, the threshold value is fixed in this situation. By taking into consideration the failure rate and the maximum execution time, the threshold value is calculated. The master node candidates' failure rates and quickest execution times are taken into account. The value of the N1 node is below the threshold value. As a result, it may be chosen as a candidate node here. Node N2 cannot be chosen as a candidate node since it has one parameter less and one larger. Given that Node N3's value is equal to the threshold value, it is more likely to be chosen as the candidate node. Also, since node N4's value is greater than the threshold value, it cannot be chosen as a candidate node. A candidate node starts operating as soon as it is chosen. There are a few things that need to be done in order to do this study. When a work is finished, a node could migrate, which would result in task failure. A novel approach is used to handle the failure issue that occurs during node mobility. The master node time is a new parameter that is added by the suggested fix. Master node time, which encourages node cooperation, is the period of time that ultimately connects end users.

The following are the key formulae for determining master node time:

1. E-cost = Master Node Time + Maximum Execution Time (Master Node Time) Then, we shall determine each node's profit.
2. Each node's profit is equal to its E-cost plus its failure node.
3. Each node's weight is determined by adding the quantity of tasks to the maximum execution time and profit.

The most heavily weighted node is now chosen. The weight value is calculated using the aforementioned equations.

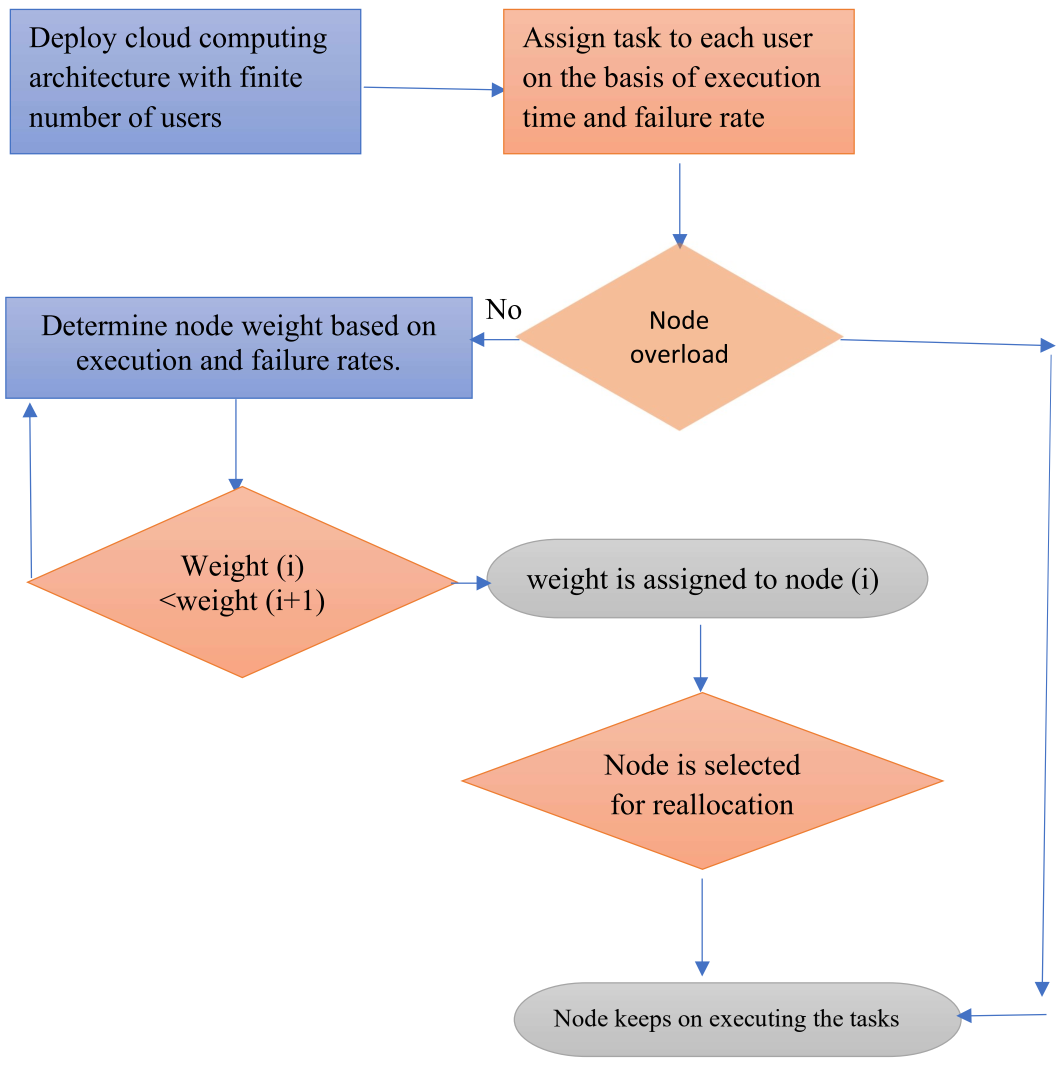
Deploy cloud computing architecture with finite number of users
Assign task to each user on the basis of execution time and failure rate
Node overload
No
Determine node weight based on execution and failure rates.
Weight (i) <weight (i+1)
weight is assigned to node (i)
Node is selected for reallocation
Node keeps on executing the tasks

Figure 3: Proposed Flowchart

To emphasize the implementation assumptions, the suggested load balancing algorithm's pseudocode, and ultimately the flowchart are described. The primary goal of this effort is to increase cloud performance by taking into account both task scheduling and load balancing. It schedules jobs well to minimize Makespan and Execution Time while maximizing resource consumption by using all of the machines' available CPUs. The CPU is first distributed equally across all VMs before being modified in accordance with the violation status. The suggested approach uses the CPU to its utmost capacity.

IV Experimental Results

The suggested study is carried out In MATLAB, the effectiveness of new and old techniques is compared using specific performance indicators.

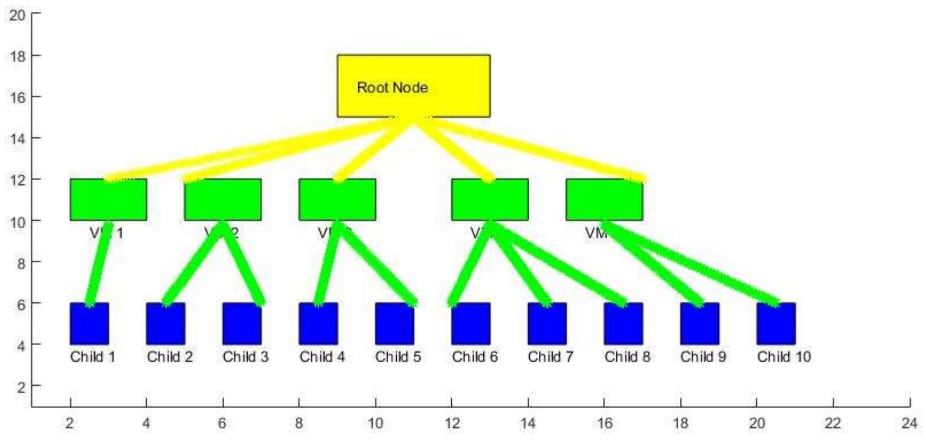

Figure 4 : Deployment of network

Figure 4 The deployment of cloud computing involves the use of a root node, which serves as the primary server for interacting with other nodes. The system also utilizes a restricted number of virtual machines and nodes, where nodes within the virtual machine run cloudlets and respond with their outcomes.

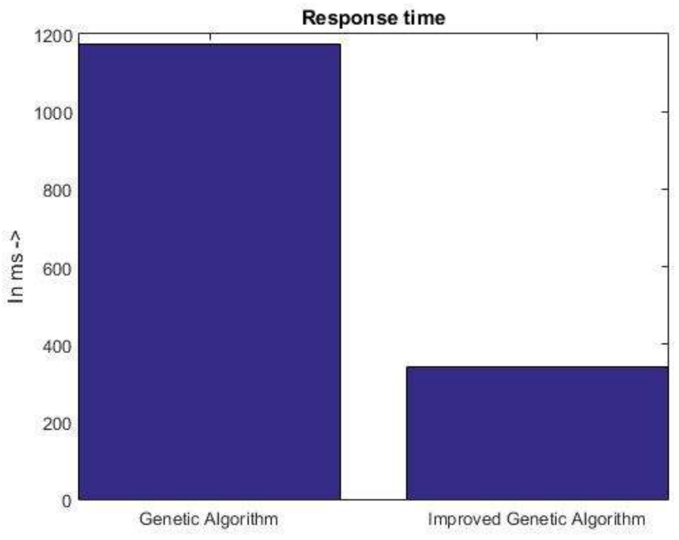

Figure 5: Comparison graph of Response Time

Figure 5 For performance analysis, the improved genetic algorithm was tested against the genetic algorithm to compare their respective reaction times. The results showed that the improved genetic algorithm exhibits faster response times when compared to the genetic algorithm.

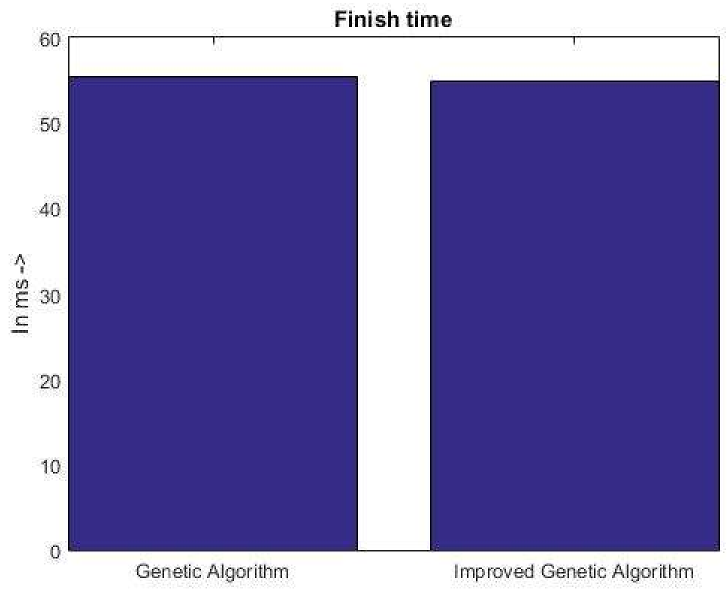

Figure 6: Comparison graph of Finish Time

Figure 6 To analyze performance, the completion times of the genetic algorithm and the suggested improved genetic algorithm were compared. The results indicated that the improved genetic algorithm completes tasks faster than the genetic algorithm.

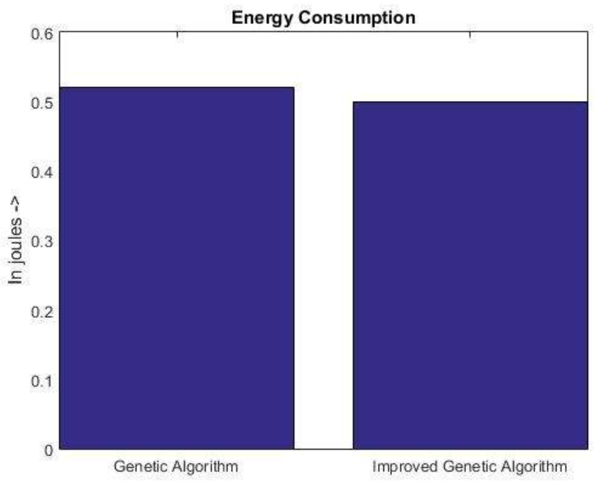

Figure 7: Comparison graph of Energy Consumption

Figure 7 To evaluate the performance, the energy consumption of the genetic algorithm was compared with that of the proposed version. The improved algorithm was found to consume less energy than the genetic algorithm.

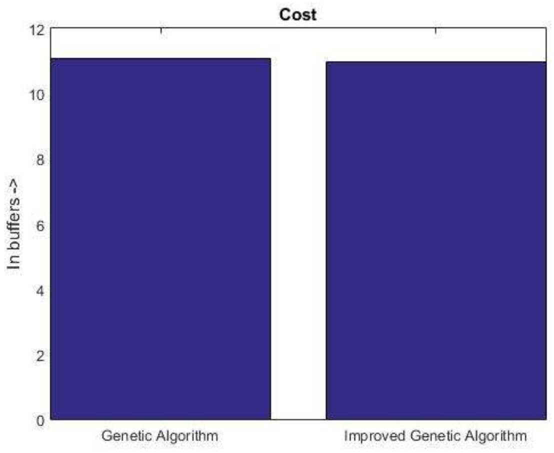

Figure 8: Comparison graph of Cost

Figure 8 For performance analysis, the cost of the genetic algorithm was compared with that of the suggested enhanced version. It was found that the improved genetic algorithm is cheaper than the genetic algorithm.

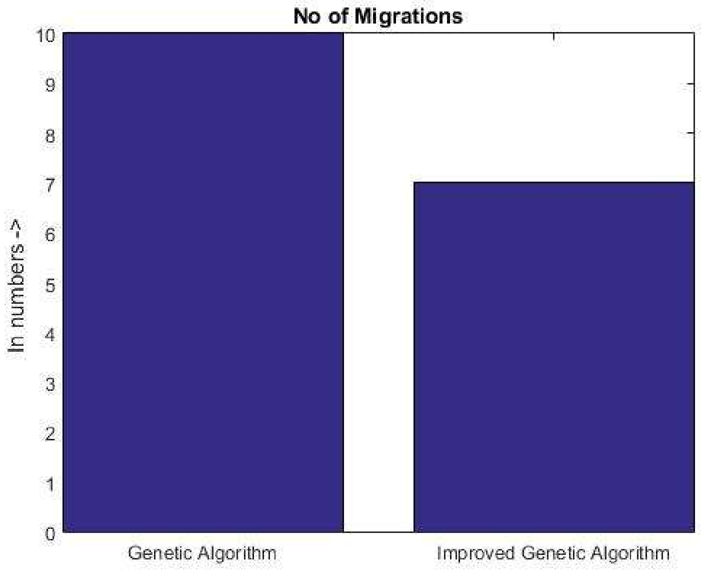

Figure 9: Comparison graph of No of Migrations

Figure 9 The performance study involved a comparison of the number of migrations between the suggested improved genetic algorithm and the genetic algorithm. It was observed that the improved genetic algorithm has a lower frequency of migrations than the genetic algorithm.

V Conclusion

Because of the dynamic nature of cloud computing, the cloud network faces a number of problems, including security, service quality, and the incidence of faults. The main problem with the cloud network that lowers its efficiency is load balancing. The augmented genetic algorithm is the method that is used to already completed work when errors are found in order to execute virtual machine migration. The enhanced evolutionary algorithm that conducts

virtual machine migration is quite sophisticated and takes a long time to run. In this paper, a modification was proposed for the enhanced genetic algorithm to reduce execution time while maintaining reliability and speed to minimize the occurrence of problems. During the period of implementation and comparison, it was discovered that the recommended modified genetic algorithm produced superior results in terms of reaction time, completion time, energy consumption, cost, and the number of migrations.
Further work on the suggested method might include merging it with existing virtual machine migration techniques, comparing it to others, and doing optimization using a hybrid meta-heuristic algorithm approach.

Reference

[1] **Poonam and Suman Sangwan (2023)** "Fuzzy Firefly Based Intelligent Algorithm for Load Balancing in Mobile Cloud Computing" Computers,Materials& Continua Tech Science Press DOI: 10.32604/cmc.2023.031729

[2] **Kanellopoulos, D.; Sharma, V.K. (2022)** "Dynamic Load Balancing Techniques in the IoT: A Review" Symmetry 2022, 14, 2554. https:// doi.org/10.3390/sym14122554

[3] **Li, Y.; Li, J.; Sun, Y.; Li, H. (2022)** "Load Balancing Based on Firefly and Ant Colony Optimization Algorithms for Parallel Computing" Biomimetics 2022, 7, 168. https://doi.org/ 10.3390/biomimetics7040168

[4] **Xiang Wang and Ying Zhang (2022)**" Enterprise Human Resource Optimization Algorithm Using PSO Model in Big Data and Complex Environment" Hindawi Journal of Environmental and Public Health Volume 2022, Article ID 1244660, 10 pages https://doi.org/10.1155/2022/1244660

[5] **Santhosh Kumar Gorva and Latha ChannagiriAnandachar(2021)** "Effective Load Balancing and Security in Cloud using Modified Particle SwarmOptimization Technique and Enhanced Elliptic Curve Cryptography Algorithm" International Journal of Intelligent Engineering and Systems, Vol.15, No.2, 2022 DOI: 10.22266/ijies2022.0430.18

[6] **Monesh Kumar and Prof. Devendra Singh Rathore (2020)** "An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing" International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org © 2020 IJCRT | Volume 8, Issue 8 August 2020 | ISSN: 2320-2882

[7] **Ouyang, M., Xi, J., Bai, W., & Li, K. (2022).** Band-Area Resource Management Platform and Accelerated Particle Swarm Optimization Algorithm for Container Deployment in Internet-of-Things Cloud. IEEE Access, 10, 86844–86863. https://doi.org/10.1109/access.2022.3198971

[8] **Alshathri, S., M. Talaat, F., & A. Nasr, A. (2022).** A New Reliable System For Managing Virtual Cloud Network. Computers, Materials & Continua, 73(3), 5863–5885. https://doi.org/10.32604/cmc.2022.026547

[9] **Rabiu, S., Huah Yong, C., &Mashita Syed Mohamad, S. (2022, September 7)**. A Cloud-Based Container Microservices: A Review on Load-Balancing and Auto-Scaling Issues. International Journal of Data Science, 3(2), 80–92. https://doi.org/10.18517/ijods.3.2.80-92.2022

[10] **Kanellopoulos, D., & Sharma, V. K. (2022, December 2)**. Dynamic Load Balancing Techniques in the IoT: A Review. Symmetry, 14(12), 2554. https://doi.org/10.3390/sym14122554

[11] **Garima Verma (2022)**"Efficient Load Balancing in Cloud EnvironmentUsing Improved Spider Monkey Optimization" DOI: https://doi.org/10.21203/rs.3.rs-1345351/v1

[12] **Li, J., Tian, X., & Liu, J. (2022, August 3)**. Dynamic Data Scheduling of a Flexible Industrial Job Shop Based on Digital Twin Technology. Discrete Dynamics in Nature and Society, 2022, 1–10. https://doi.org/10.1155/2022/1009507

[13] **Hieu N. Le and Hung C. Tran (2022)**"Ita: The Improved Throttled Algorithm of Load Balancing On Cloud Computing" International Journal of Computer Networks & Communications (IJCNC) Vol.14, No.1, January 2022DOI: 10.5121/ijcnc.2022.14102

[14] **Vidya S. Handur and Santosh L. Deshpande et.al (2021)**" Particle Swarm Optimization for Load Balancing in Distributed Computing Systems– A Survey" Turkish Journal of Computer and Mathematics Education Vol.12 No.1S (2021), 257-265

[15] **Senthil Prabakaran and RamalakshmiRamar (2021)** "Software Defined Network: Load Balancing Algorithm Design and Analysis" The International Arab Journal of Information Technology, Vol. 18, No. 3, May 2021

[16] **Nabila Djennane, RachidaAoudjit, Samia Bouzefrane.(2018)**"Energy-Efficient Algorithm for Load Balancing and VMs Reassignment in Data Centers" FiCloud Workshops, IEEE, Aug 2018, Barcelone, Spain. pp.225-230, 10.1109/W-FiCloud.2018.00043. hal-03181645