M. A. Kaleem[1]*,

S. Jain[2]

Load Balancing Approach in Cloud Computing Using Genetic Algorithm

Abstract: - The era of cloud computing where information are stored over the servers virtually, will flourish how viably the virtual infrastructure are started up and accessible resources are effectively used. In the new normal of pandemic, use of cloud services are increasing at a rapid rate day after day and for the outstanding performance it should be balanced in such a way so that increase demand can be handled & processed in minimal time to achieve business continuity. In cloud environments, heterogeneous virtual machine (VM) configurations are dynamically provisioned to process concurrent user requests. As network uncertainties and user traffic spikes intensify, resource contention becomes inevitable—leading to VM overloads and performance bottlenecks. This scenario triggers a critical need for real-time load balancing, a challenge often categorized as NP-Complete. Efficient load distribution is therefore pivotal in ensuring service-level objectives are met without overburdening specific nodes or leaving others underutilized. Load balancing in cloud computing involves the intelligent allocation of incoming workloads across virtualized resources to optimize throughput and system responsiveness. Numerous algorithms have been proposed in this domain; however, this paper introduces a novel Genetic Algorithm (GA)-driven load balancing strategy that significantly enhances task distribution efficiency across virtual machines by leveraging evolutionary computation principles.

Keywords: Cloud Computing, Load Balancing, Genetic Algorithm, GA, Improved Genetic Algorithm, Virtual Machine Allocation

1. Introduction

Cloud Computing is planned on the basis of the concept of distributing multiple computing resources in the form of service to clients on the internet. It is not easy to set one standard general definition for cloud computing as it is broad concept in which the entireness of the spectrum is incorporated with much efforts that aims at defining a specific cloud attribute. As per the National Institute of Standards and Technology (NIST), cloud computing is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. This standardized definition underscores the core attributes of cloud platforms: elasticity, resource pooling, self-service provisioning, and rapid scalability, all of which form the foundation for modern distributed computing environments. Currently, a substantial part of the WWW is supported with cloud data centre traffic using cloud computing.

To enhance overall system performance and efficiency, load balancing is employed to intelligently distribute incoming workloads across smaller processing units. In the context of cloud computing, cloud load balancing refers to the systematic allocation of workloads across multiple virtualized environments—such as VMs, storage nodes, or network segments—ensuring optimal resource utilization, improved system responsiveness, and reduced energy consumption.

[1]*Corresponding Author: M. A. Kaleem, J.S. University, Shikohabad, UP. 283135, India. Email: meet2atifkhan@gmail.com

[2]S. Jain, J.S. University, Shikohabad, UP. 283135, India.

Cloud load balancing not only balances computational demands across servers and networks but also enables organizations to dynamically assign resources in alignment with application requirements and service-level objectives. It addresses the volatile and bursty nature of Internet-driven traffic by ensuring equitable distribution of workloads, which directly contributes to enhanced system scalability, fault tolerance, and overall user satisfaction.

Efficient load distribution minimizes energy usage by preventing both resource underutilization and server overloading, thus contributing to energy-aware cloud operations. By leveraging this approach, key non-functional attributes—such as low latency, high availability, and failover support—are inherently supported.

Load balancing in cloud infrastructures operates at two hierarchical levels:

- **Application-to-VM Mapping Level:** At this level, incoming tasks or services are assigned to appropriate virtual machines (VMs), considering factors such as application size, deadline, or computational complexity.
- **VM-to-Host Mapping Level:** Here, the load balancer determines the physical host on which the selected VM will run. This mapping ensures that physical machine resources are efficiently used, balancing workloads across the underlying hardware.

Various algorithms have been extensively adopted to achieve efficient load balancing in cloud computing environments [2]. Prominent among them are Round Robin, Active Monitoring Load Balancer (AMLB), Throttled Load Balancing (TLB), Ant Colony Optimization (ACO)-based Load Balancer [3], as well as the Min-Min and Max-Min Scheduling Algorithms. Each of these techniques is designed with distinct operational principles and optimization goals. For instance, Round Robin emphasizes simplicity and fairness in task distribution, whereas AMLB and TLB incorporate real-time resource awareness to dynamically allocate workloads. ACO-based methods leverage bio-inspired intelligence to find optimal load distribution paths, particularly useful in complex and adaptive systems. The Min-Min and Max-Min strategies prioritize task execution based on expected completion times, aiming to balance execution efficiency across heterogeneous resources. These algorithms offer varied trade-offs in terms of task scheduling efficiency, resource utilization, fault tolerance, and energy-awareness, and are typically selected based on specific workload patterns, application criticality, and service-level objectives within the cloud infrastructure.

One of the simplest algorithms in which the quantum of time or interval term is used is Round Robin. The partition of time is performed in it. The specific time quantum is assigned to each node. All the operations of node are performed in this quantum. The scheduling of time quantum is very significant in RR as the similarity may be occurred between the RR scheduling algorithm and the FCFS planning if the time quantum is very large. An additional load is produced on the scheduler while determining the size of quantum that is the limitation of this technique even if this algorithm is the easiest. Furthermore, the higher context switches are included in this algorithm due to which the turnaround time, and low throughput is maximized [4]. The Active Monitoring Load Balancer (AMLB) algorithm is characterized by its dynamic behavior and real-time adaptability. It maintains comprehensive information about all virtual machines (VMs), including their current load status and the number of requests allocated to each VM. Upon receiving a new task, AMLB performs an initial sorting of available VMs and identifies the one with the least load. It then returns the corresponding VM ID to the Data Center Controller, which subsequently dispatches the task to the selected VM. Additionally, AMLB is periodically updated by the controller regarding the current distribution status, ensuring that its allocation decisions reflect near real-time system states. However, a critical limitation of AMLB lies in its lack of historical request tracking—it does not verify whether an incoming request has been previously executed, which may lead to redundant allocation or suboptimal task execution under specific scenarios.

On the other hand, the Throttled Load Balancing (TLB) algorithm employs a more controlled and state-aware mechanism. The TLB load balancer maintains a VM allocation table that includes the index and current status (busy or available) of each virtual machine. This enables the TLB to allocate tasks only to idle and suitably matched VMs, thereby preventing overloads and ensuring that resources are used efficiently. While TLB offers improved precision in allocation, it may result in delayed task execution if no suitable VMs are immediately available [5].

First of all, the client or server creates a request for the data centre so that the appropriate VM is discovered for performing the suggested task. A load balancer is requested by the data centre for the distribution of VM. The load balancer starts the scanning of the index table from starting. This method continues until the VM that is available is first discovered or the index table is thoroughly scanned. In ACO based load-balancing algorithm, the ant colonies are highly influenced by ant colony optimization. The foraging behaviour was used by ants while working together. In this algorithm, the selection of the head node is performed in such [6] a manner that the highest number of neighbour nodes is included in it. There are two directions in which the ants are moved. First is the forward movement, in which the ants move forward in a cloud for the collection of information about the loads of nodes. The second is the backward movement. The ants go backward and redistributing of the load among the cloud nodes is done when any ant discovers the overloaded node in its way. The minimum completion time estimation is initially done using the Min-Min algorithm [7]. The task which takes least completion time is selected and it is allocated to the corresponding node. In contrast to the Min-Min algorithm, which prioritizes tasks with the shortest execution time the Max-Min scheduling algorithm adopts a more latency-aware strategy. Initially, the expected completion times of all tasks are calculated across available computational nodes. Among these, the task with the maximum expected completion time is selected for allocation. By addressing the longest task first, the algorithm mitigates the risk of prolonged system makespan and prevents high-latency jobs from becoming bottlenecks. This allocation cycle is repeated iteratively until all tasks are successfully mapped to their respective nodes, ensuring a balanced trade-off between task fairness and system throughput.

Idea of virtualization is utilised in cloud which prompts to have a load balancing in the cloud. Virtualization implies giving a sensible name to physical assets and at whatever point this name is alluded it will point towards relating physical assets. Different clients will get to cloud at same time and it is extremely important to serve them all with least reaction time and better assistance. As a result, load balancing is generated into outcomes to uniformly change the solicitation of different clients on virtual machines. In view of the fact that as the size of the problem increases, the size of the solution will also grow, it is claimed that load balancing is an NP-Complete problem process. That implies the more the solicitation comes to cloud it will get harder to do balancing among the Different Virtual Machines.

Genetic Algorithm is well known for taking care of NP-Complete issues. GA is one of method which has a place with the class of transformative algorithms which produces results motivated by innate growth. The Simple GA Concepts are as per the following:

- **Population** - Set of viable solutions for proposed issue.
- **Chromosome -** Sole in populace.
- **Gene**- Variable present in a chromosome.

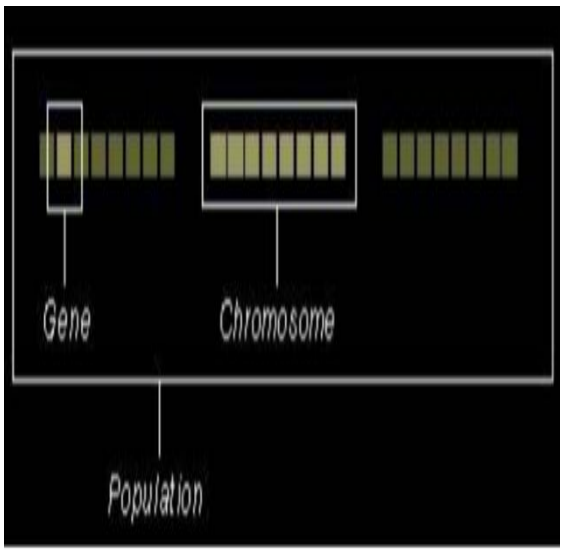

Figure 1: Genetic Algorithm Population

❖ **Fitness Function** - a sort of a target function used to make sense of how close the explanation is accomplishing the set point.

❖ The GA emerge by three operators: -

- **Selection**- Best fittest solutions are selected.
- **Crossover**- More than one parent is selected, to generate a child.
- **Mutation**- amending gene value in chromosome.

2. Literature Review

Karan D. Patel et al. [8] proposed a hybrid load balancing algorithm designed to optimize workload distribution across cloud systems by integrating two distinct scheduling strategies. The first component of the approach is inspired by a modified honey bee foraging behavior, where tasks are scheduled based on dynamic priority levels, simulating the adaptive nature of bee colonies. For non-priority tasks, an enhanced Weighted Round Robin (WRR) algorithm was utilized to ensure fair and systematic distribution. The combined mechanism demonstrated improved system performance by minimizing task completion time and enhancing resource utilization efficiency. The authors also noted that future work would involve incorporating additional Quality of Service (QoS) parameters, such as latency, availability, and reliability metrics, to further refine the algorithm's effectiveness under varying workload conditions.

Guilin Shao et al. [9] proposed a data correlation-aware load balancing strategy aimed at optimizing virtual machine (VM) placement and migration in cloud environments. Their approach specifically addressed the challenges arising from data correlation among co-located VMs, which can lead to uneven load distribution and resource contention. To mitigate unnecessary resource usage, the strategy employed selective VM migration, wherein migration candidates were identified based on similarity metrics between VMs handling correlated datasets. By aggregating load-intensive and interrelated data into cohesive migration units, the approach ensured that migration decisions were both data-aware and load-sensitive. Experimental results demonstrated that this technique effectively reduced coordination overhead while enhancing resource utilization efficiency, offering a scalable solution for dynamic cloud infrastructures.

Pradeep Kumar Tiwari et al. [10] proposed a novel load balancing strategy based on the Dynamic Weighted Least Connection Method (DWLM) to address the persistent issue of load imbalance in cloud computing environments. The performance of the proposed technique was evaluated against established algorithms such as ESCEL and the Push-Pull algorithm, using multiple performance metrics. The study not only highlighted the comparative advantages of the DWLM-based approach but also discussed alternative load balancing techniques and outlined potential directions for future research in cloud load management. To validate the strategy, simulations were initially conducted using the CloudAnalyst simulator, which provided a controlled experimental environment. Furthermore, the authors implemented the algorithm in a real-world cloud environment, where the results demonstrated noticeable variations, emphasizing the gap between simulated and real-time performance outcomes.

Violetta N. Volkova et al. [11] emphasized that cloud computing, as an emerging Internet-based paradigm, primarily caters to commercial computing needs by offering scalable and centralized resource management. The study highlighted the significance of load balancing in enhancing the performance and responsiveness of centralized cloud servers. To evaluate the effectiveness of various load balancing strategies, the authors employed CloudAnalyst, a widely used simulation tool [12], to model and analyze algorithmic performance under controlled scenarios. The comparative analysis revealed that the Throttled Load Balancing algorithm outperformed other techniques in terms of average response time, while also registering higher data center processing times. These results underscore the trade-offs involved between faster request handling and increased resource engagement at the infrastructure level.

Chunyan Xue et al. [13] proposed a lightweight and autonomous load balancing strategy tailored for cloud environments, in which information processing requests are assigned dynamically based on the real-time resource availability of individual servers. Notably, the approach operates without introducing any external load balancing

devices, thereby simplifying deployment and reducing infrastructure dependencies. The task distribution mechanism leverages a pull-based model, wherein servers actively retrieve tasks according to their current load conditions. Resource allocation within the cluster is performed dynamically, and registration nodes can join or leave the system seamlessly—without requiring inter-node communication—which enhances system flexibility and scalability. The cluster size can be adjusted on demand, allowing the environment to adapt to fluctuating workloads efficiently. By mitigating issues such as bottlenecks in key nodes, unbalanced task assignments, and centralized controller failures, the proposed method significantly reduces resource wastage and lowers the operational cost of the cluster system. Additionally, it effectively addresses the single-point bottleneck problem commonly associated with traditional load balancing devices, thereby improving system resilience and performance.

J. Mercy Faustina et al. [14] introduced a Self-Governing Agent-based Load Balancing (SGA-LB) strategy, wherein an autonomous migration agent was employed to facilitate dynamic and adaptive load balancing within cloud environments. The proposed mechanism evaluated the real-time state of the system to distribute workloads efficiently across computing nodes, particularly in highly dynamic and heterogeneous infrastructures. The load balancing decision was made based on multiple node-level parameters, including CPU utilization, memory consumption, and network bandwidth usage, thereby ensuring a more granular and responsive allocation process. The effectiveness of the SGA-LB approach was assessed using multiple performance metrics, demonstrating its ability to enhance system responsiveness and achieve balanced resource utilization. The results confirmed that this method significantly improves overall system performance by proactively adapting to fluctuating workloads and maintaining equilibrium across distributed resources.

Manoj Kumar Patra et al. [15] proposed a randomized load balancing strategy for containerized cloud environments, utilizing a Local Search-based “Balls into Bins” model. In this approach, computational tasks are modeled as balls and servers (or containers) as bins, aiming to distribute workloads uniformly across all available resources. Initially, a fully connected undirected graph representing the interface topology was constructed, which was then transformed into a minimum edge-weight graph to reduce network overhead and communication cost. The experimental evaluation demonstrated that the proposed method achieved a uniform load distribution across all containers, effectively minimizing the disparity between the maximum and minimum server workloads within the cloud network. The technique also highlighted the potential of probabilistic methods for achieving low-latency and cost-efficient load alignment in dynamic containerized infrastructures.

Table 1. Comparative Table

Author & Year	Technique Used	Targeted Problem	Key Features	Pros	Cons	Result Summary
Karan D. Patel et al. [8]	Hybrid (Modified Honey Bee + Weighted Round Robin)	Priority-based & fair task distribution	Dynamic prioritization; WRR for non-priority tasks; enhanced resource utilization	Improved performance, reduced task completion time	Lacks integrated QoS consideration (noted as future work)	Improved efficiency with potential for further QoS integration
Guilin Shao et al. [9]	Data-Correlation-Aware VM Migration	VM-level resource contention due to data correlation	Similarity-based migration units; data-aware VM allocation	Reduces overhead; improves utilization	Requires data similarity metrics; overhead of migratio	Effective reduction in coordination overhead

					n analysis	
Pradeep Kumar Tiwari et al. [10]	Dynamic Weighted Least Connection (DWLM)	Load imbalance across VMs	Weighted connection tracking; compared with ESCEL & Push-Pull	Balanced load; tested in simulation and real environments	Disparity in simulated vs real-world results	Validated with real-world testing; reliable but context-sensitive
Violetta N. Volkova et al. [11]	Throttled Load Balancing (Comparative Study)	Centralized server response optimization	Simulated multiple algorithms; performance comparison	Throttled LB showed best response time	Higher data center processing time; infrastructure trade-offs	Throttled LB effective in response, but costlier on backend
Chunyan Xue et al. [13]	Autonomous Pull-Based Dynamic Balancing	Decentralized load distribution without external devices	Pull mechanism; self-adjusting cluster; no node-to-node communication	Low overhead; high scalability; no single point of failure	Lacks central coordination; may underperform in some high-traffic cases	Scalable and cost-efficient with fault-tolerant properties
J. Mercy Faustina et al. [14]	Self-Governing Agent-Based Load Balancer (SGA-LB)	Real-time dynamic load balancing in heterogeneous systems	Autonomous agents; considers CPU, memory, bandwidth	Fine-grained control; high responsiveness	Agent management overhead; complex decision logic	Effective in dynamic environments; high resource balance
Manoj Kumar Patra et al. [15]	Randomized 'Balls into Bins' (Graph-Based)	Uniform workload alignment in containerized clouds	Graph topology optimization; local search heuristics	Minimized load variance; reduced network cost	Graph construction overhead; limited to homogeneous task types	Balanced workload distribution with efficient graph model

3. Research Methodology

The core focus of this research is load balancing in cloud computing, specifically through the use of dynamic task scheduling techniques. Unlike static schedulers, dynamic scheduling makes real-time decisions during task execution, considering factors such as resource availability, inter-node communication overhead, energy efficiency, and application workload variability. This approach also supports task migration, enabling reallocation based on the current status of cluster resources to prevent bottlenecks and optimize performance. Notable applications of dynamic scheduling include resource-aware planning and energy-efficient task scheduling, both aimed at maximizing system throughput while adhering to job deadlines.

In this study, a Genetic Algorithm (GA)-based strategy is employed for cloud task scheduling. GAs are adaptive, population-based metaheuristic techniques inspired by the principles of natural selection and biological evolution. As a global stochastic search method, GA iteratively evolves a population of candidate solutions to approach optimal task distributions. Each individual in the population, referred to as a genotype, is typically represented by a bit-string chromosome encoding potential scheduling solutions. The evolution proceeds through four key phases: selection, crossover, mutation, and fitness evaluation, each contributing to refining the task scheduling toward improved resource utilization and reduced energy consumption.

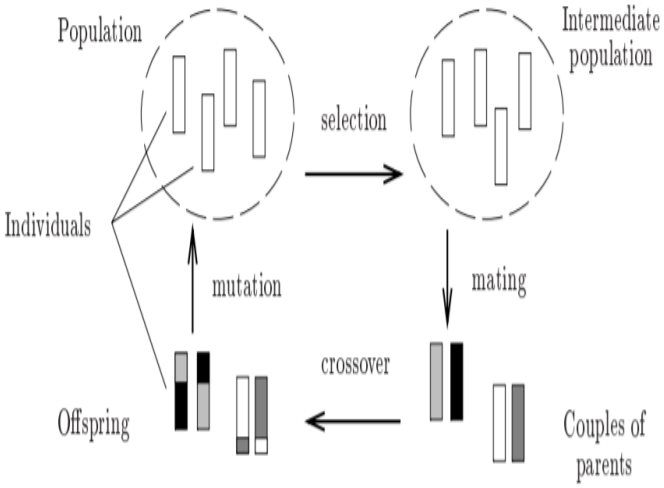

Figure 2: Four Phases of Genetic Algorithm

The execution of a Genetic Algorithm (GA) begins with the generation of an initial population of individuals, typically initialized at random. Each individual represents a candidate solution encoded as a bit-string chromosome. The next phase involves the creation of an intermediate population through a fitness-based selection process, wherein individuals are chosen probabilistically based on their relative fitness values. This mechanism mimics natural selection, favoring the propagation of individuals with higher fitness by allocating them a greater number of reproductive opportunities.

Once the intermediate population is populated, individuals are grouped into pairs, and a crossover operator is applied to each pair with a probability P_c. The crossover operation facilitates genetic recombination by exchanging segments of genetic material between the two selected parent individuals, thereby generating new offspring. A common example is the one-point crossover, where a random crossover point is chosen along the bit string, and the tails of the two parent chromosomes are swapped beyond that point. This process yields two new offspring, which are then inserted into a new intermediate population.

If the crossover is not applied (i.e., with probability $1\text{-}P_c$), the original parent pair is carried over unchanged into the next population. Together, the selection and crossover phases constitute the cooperation step of the GA, where solutions interact to explore new regions of the search space and evolve toward optimality.

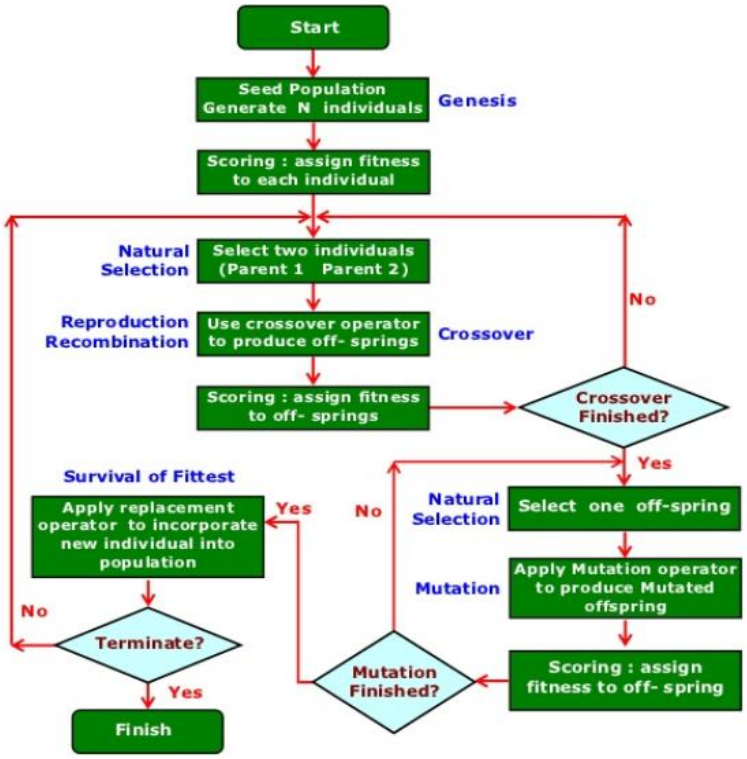

Figure 3: Flowchart of Genetic Algorithm

The final phase in the execution of a Genetic Algorithm (GA) involves the application of a mutation operator, which introduces genetic diversity into the population by randomly altering individual chromosomes. This phase, often referred to as the self-adaptation step, is critical in preventing premature convergence and ensuring broader exploration of the solution space. A commonly used mutation technique is the bit-flip operator, which inverts the value (from 0 to 1 or vice versa) of a randomly selected bit in the bit string representation of an individual. This mutation is applied to each individual with a predefined probability P_m.

In Following mutation, the newly generated intermediate population is used to update the current population. In the Generational Replacement GA, the intermediate population is of equal size to the original, and it entirely replaces the previous generation thereby renewing the population in each iteration. In contrast, the Steady-State GA maintains a smaller intermediate population, where new offspring do not necessarily replace their parents but can overwrite any individual in the population based on selection criteria. The execution of the algorithm continues until a termination condition is met, typically defined by a fixed number of generations or convergence of fitness values.

4. Algorithm of Proposed Method

Step 1: Start

Step 2: Incoming request priority is computed on its timestamp.

Step 3: Population initialised on above jobs priority.

Step 4: Best chromosome should be selected on the basis of fitness.

Step 5: Perform Crossover Mapping partially.

Step 6: Use Swapping Technique for Mutation.

Step 7: Chromosome to be added in population.

Step 8: Termination condition should be checked.

Step 9: End

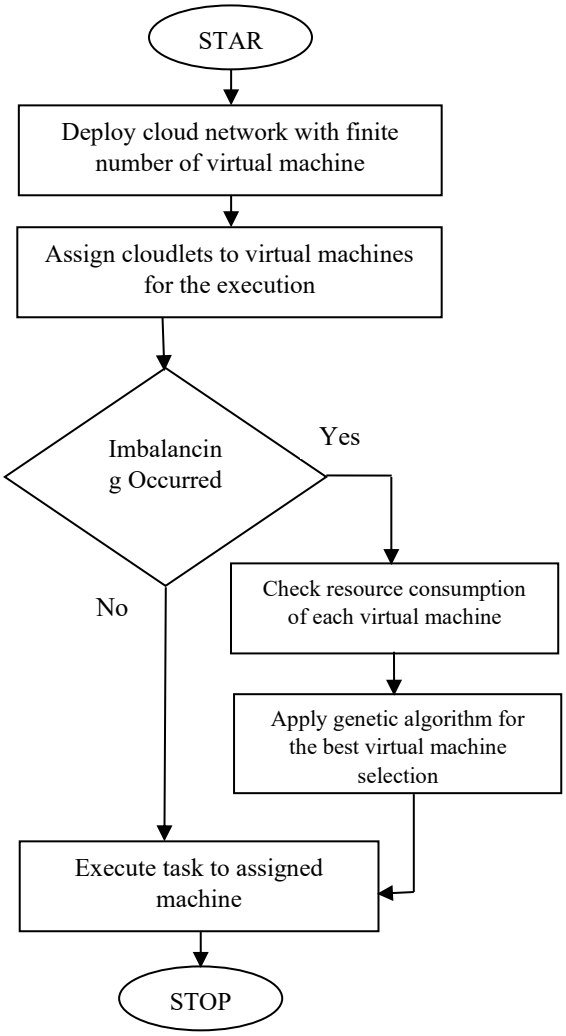

Figure 4: Proposed Flowchart

5. Result and Discussion

In this study, the genetic algorithm for cloud computing load balancing is applied. The weight parameter is applied to the load balancing genetic algorithm. In terms of some parameters shown in this section, the results are analysed.

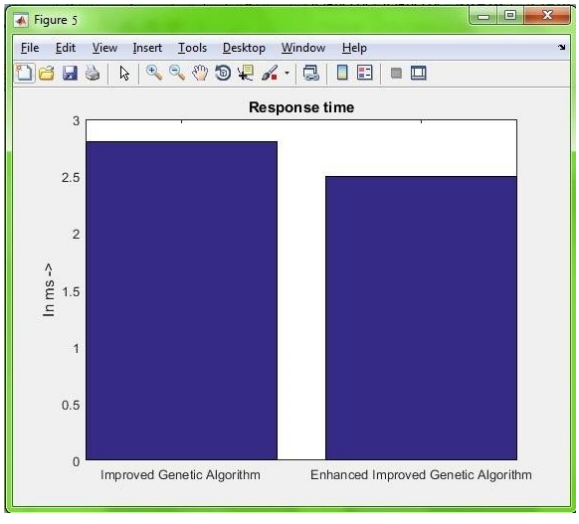

Figure 5: Response Time

As shown in Figure 5, The genetic algorithm's response time is compared to the enhanced genetic algorithm. It is analyzed that when genetic algorithm weights are applied, response time is reduced.

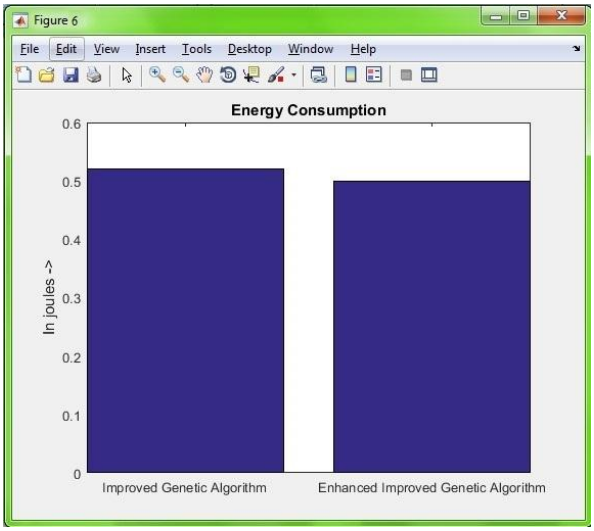

Figure 6: Energy Consumption

The energy consumption of the genetic algorithm is compared, as shown in Figure 6, to the improved genetic algorithm. It is analyzed that energy consumption is reduced when the weights are implemented with the genetic algorithm.

6. Conclusion

The primary issue looked by cloud computing is load balancing which maintains a strategic distance by most algorithms presently. Genetic Algorithm is one of the dynamic and unified load balancing approach which utilises its tasks, for example population, chromosomes, mutation, with some moreover highlighted condition. The difficult proclamation characterized in the paper presents the outcome examination utilizing MATLAB with hereditary calculation in distributed computing. In our proposed strategy, we have considered time & energy yet there can be numerous different boundaries which can be thought of. Thus the analysis executed infers that improved genetic algorithm gives better outcomes in terms of response time, energy and all the virtual machines are appropriately balanced with the designated tasks coming about better execution by guaranteeing that no single Virtual Machine is either over-stacked or under-stacked. Additionally, a basic methodology of GA has been utilized however sort of selection techniques and crossover might be applied as a future work to produce outcomes that are progressively efficacious, increasingly successful and tuned.

7. References

[1] Narayan Joshi, Ketan Kotecha, D B Choksi, Sharnil Pandya, “Implementation of Novel Load Balancing Technique in Cloud Computing Environment”, 2018, International Conference on Computer Communication and Informatics (ICCCI)

[2] Vishalika, Deepti Malhotra, “LD_ASG: Load Balancing Algorithm in Cloud Computing”, 2018, Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)

[3] Mao-Lun Chiang, Hui-Ching Hsieh, Wen-Chung Tsai, Ming-Ching Ke, “An improved task scheduling and load balancing algorithm under the heterogeneous cloud computing network”, 2017, IEEE 8th International Conference on Awareness Science and Technology (iCAST)

[4] Ling Tang, “Load Balancing Optimization in Cloud Computing Based on Task Scheduling”, 2018, International Conference on Virtual Reality and Intelligent Systems (ICVRIS)

[5] Hussain A Makasarwala, PrasunHazari, “Using genetic algorithm for load balancing in cloud computing”, 2016, 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)

[6] Haiying Shen, Liuhua Chen, “A Resource Usage Intensity Aware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters”, 2020, IEEE Transactions on Cloud Computing, Volume: 8, Issue: 1

[7] J. Mercy Faustina, B. Pavithra, S. Suchitra, P. Subbulakshmi, “Load Balancing in Cloud Environment using Self-Governing Agent”, 2019, 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)

[8] Karan D. Patel, Tosal M. Bhalodia, “An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing”, 2019, International Conference on Intelligent Computing and Control Systems (ICCS)

[9] Guilin Shao, Jiming Chen, “A Load Balancing Strategy Based on Data Correlation in Cloud Computing”, 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)

[10] Pradeep Kumar Tiwari, Sandeep Joshi, “Dynamic weighted virtual machine live migration mechanism to manages load balancing in cloud computing”, 2016, IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

[11] Violetta N. Volkova, Liudmila V. Chemenkaya, Elena N. Desyatirikova, Moussa Hajali, AlmothanaKhodar, Alkaadi Osama, “Load balancing in cloud computing”, 2018, IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)

[12] M. A. Kaleem and P. M. Khan, “Commonly used simulation tools for cloud computing research,” in 2015 International Conference on Computing for Sustainable Global Development, INDIACom 2015, 2015, pp. 1104–1111.

[13] ChunyanXue, “Method and Implementation of Server Load Balancing in Cloud Computing”, 2018, 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)

[14] J. Mercy Faustina, B. Pavithra, S. Suchitra, P. Subbulakshmi, “Load Balancing in Cloud Environment using Self-Governing Agent”, 2019, 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)

[15] Manoj Kumar Patra, Dimple Patel, Bibhudatta Sahoo, Ashok Kumar Turuk, “A Randomized Algorithm for Load Balancing in Containerized Cloud”, 2020, 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)