Journal of Harbin Engineering University Vol 44 No. 5
ISSN: 1006-7043 May 2023

Performance Evaluation on Load Balancing Algorithms in Cloud Computing Environment: A Comparative Study

Ms. Shruti Tiwari [1], Dr. Chinmay Bhatt [2]
[1]Research Scholar, Dept. Of Computer Science Engineering,RKDF College, Bhopal (M. P.), India
[1]shruti.tiwari08@gmail.com
[2] Professor, Dept. Of Computer Science Engineering, RKDF College, Bhopal (M. P.), India
[2]chinmay20june@gmail.com

Abstract— In the realm of parallel and distributed computing, cloud computing is quickly becoming a viable alternative to the use of traditional physical hardware. Clouds are groupings of virtual resources that can be generated on demand to fulfill the particular needs of individual users. These requirements may be quite specific. Due to the large number of users, tasks, and data, processing in the cloud is also extremely significant. Load balancing is of tremendous benefit in cloud virtual environments where vital tasks are performed. Load balancing is a crucial topic of study because of the increasing prevalence of cloud computing and the consequent increase in the number of customers expecting more services and better outcomes. Recently, efficient work scheduling has become one of the cloud computing industry's primary obstacles. The primary objective of this study is to compare and contrast the most popular scheduling and load-balancing approaches as they apply to the cloud computing environment. The comparative analysis of HISA, DI-HISA, LBA, LB-HEFT, and LBMPSO techniques is provided in this work to illustrate the performance evaluation over 50 cloudlets and 5 virtual machines. The experiment's findings demonstrate that our second suggested approach, DI-HISA, obtained the shortest makespan of 84.87 milliseconds when compared with the other methods.

Index Terms— Cloud Computing, Load Balancing, Virtual Machine, Task Scheduling Algorithms, Makespan.

I. Introduction

CLOUD Computing is a network technology that is built on the internet. It has helped speed up the development of information technology by using internet-based computing tools to serve customers with different needs. It provides resources like software and hardware, as well as websites for software creation and evaluation tools[1][2]. Services are used to carry out this resource distribution. The last two are examples of cloud computing that are classified as PaaS and SaaS, or platform and software as a service. The former is an example of a cloud service that is referred to as "infrastructure as a service" (IaaS)[3]. The term "cloud computing" refers to an interconnected, computing on-demand service that charges customers on an as-needed basis for using a pool of shared computer resources[4]. Companies in this industry that are well-known and influential include Microsoft, Amazon, SAP, Google, VMware, Oracle, IBM, and SalesForce, amongst many more[1][2]. High-tech IT companies make up the majority of the aforementioned cloud providers. Two categories may be used to categorize the cloud computing paradigm. The first is the service delivery model, which describes the usual services provided by a cloud provider. As an example, consider the three essential service models below, all of which are often utilized due to this factor. SaaS, PaaS, and IaaS[5][6]. Aspects such as number of users, kind of organization affiliated with, size of system, and level of access are considered as part of the larger whole that is the cloud computing paradigm. Officially recognizing four unique cloud deployment methods, the "National Institute of Standards and Technology" (NIST) defines cloud computing as follows: The four main categories of clouds are the internal, external, community, and hybrid varieties[7].

The cloud computing paradigm is ideal if its facilities are used as efficiently as is practical. Adequate cloud resource management may help achieve this efficient utilization. Effective resource management

relies on approaches for allocating, scheduling, and scaling available resources. These resources are made available to customers in the form of Virtual Machines (VM) through the use of a hypervisor, which may be software, hardware, or both. The ability to convert a single-user's physical system into a shared virtual one is a major benefit of cloud computing. Because there aren't many virtual tools available right now, a provider of cloud services (CSP) is a very important part of the service offering to customers. While fulfilling requests from users, some VMs will have high user task traffic, while others may experience low user task traffic. A Cloud Service Provider (CSP) with imbalanced machines is the result of a disparity between user loads and used resources[9].

According to the CSP, The Quality of Service (QoS) that is guaranteed by the client and provider in line with their Service Level Agreement (SLA) criteria might be negatively impacted by load unbalancing. Under these circumstances, load balancing, often known as LB, which is a special research topic that piques the attention of academics, is required to be done. In cloud computing, load balancing may be performed at the virtual machine or host computer level[2].

A task uses up VM resources; whenever many processes make demands at the same time, the VM's resources become depleted and cannot be utilized to satisfy the processing requirements of a new job since they are already in use. When such a circumstance occurs, the VM is considered to have reached an overcrowded condition. Tasks will either starve to death at this stage or come to a standstill with little chance of being completed. As a result, it is necessary to move jobs to an additional resource on a different VM. The three primary components of workload migration are load balancing, which examines the current utilization of a machine's resources, resource discovery, which seeks new useful resources, and workload migration, that transfers more tasks to resources that are already available. There are three separate units responsible for carrying out these functions: the workload broker, the resource discovery unit, and the job transfer unit.

Load balancing is the process of distributing tasks among several computers in a distributed system (like the cloud) such that no one device is overworked, underutilised, or sitting idle [10][11]. Load balancing is an approach that seeks to enhance cloud performance by accelerating a wide range of bounded elements, including speed of response, time for processing, stability of the system, etc [12][13]. It is an optimization method used when arranging jobs is a problem that is NP-hard. Researchers have come up with a number of ways to balance the workload, but most of them focus on work planning, job sharing, resource scheduling, allocation of resources, and resource management. To the best of our understanding, we were unable to locate extensive and detailed literature addressing the causes that lead to load unbalancing situations.

Different timing methods can be employed for various activities. The timing methods can use better execution speed and keep the system load balanced. How well the cloud works rests on the formulas used to schedule work. Scheduling algorithms can be either predictive or not. No force can prevent a job from finishing while using a non-preemptive planning mechanism. On the other hand, the advanced scheduling mechanism allows for a variety of reasons to halt a task's execution. We require a set of metrics that include waiting time, in addition, to return time in order to assess how effectively scheduling algorithms perform[14]. The waiting period for a job is the total length of time that the task was in the ready queue, whereas the turnaround period for a task is the whole amount of time that the work takes to complete. The CPU's scheduling technique has no bearing on how long the task takes to complete or how much it takes for results to be returned. It only affects how long a task goes in the ready queue. Often, a job can give some results really early on, and it can keep coming up with new results while the client is getting the results from the past[15].

The remaining parts of this investigation are structured in the following manner. In the second part, the concepts of load balancing and task ordering, in addition to other linked works, are broken down and discussed in more detail. In Section 3, we talked about the task scheduling factors, and in Section 4, we talked about the different methods for arranging tasks. In Section 5, we talked about some common ways to schedule tasks and divide up the work. In the next part, section 6, some data from comparing abilities were shown. Section 7 gives a final review of the result and a list of things that need to be done in the future.

II. Related Work

Due to cloud computing's recent emergence, research in this subject is in its early stages. For load balancing in cloud computing, various authors provide a variety of techniques. Numerous techniques have been devised for the administration and scheduling of cloud-based workloads. However, combining the two approaches has received significantly fewer resources in previous research. A combination of methods can enhance the system's performance by selecting the most effective resources from those that are available. Lata and Singh [16] recommended the Hybrid Method for Load Balancing in the Cloud (HA-CLB). The objective of workflow scheduling is to optimize makespan, costs, and resource utilization using a combination of deterministic and heuristic techniques.

Mohanty et al. [17] proposed a cloud-based paradigm that makes use of fog to help manage SGs' resources is offered. The suggested fog-aided cloud system makes use of a number of load-balancing strategies to improve its performance. Fair distribution of work between SG users' activities and service providers is achieved via the use of four distinct meta-heuristic algorithms: ant colony optimisation (ACO), particle swarm optimisation (PSO), and gradient-based optimizer (GBO) artificial bee colony (ABC),. Among the several recently developed meta-heuristic optimization algorithms, simulation results show that GBO performs the best.

In this work [18], The implementation of the hybrid heuristic algorithm HHMA on an IaaS network of computers could prove to be beneficial. The cluster was broken up into small pieces using a better K-means clustering method (heuristic) to make the design go faster and shorten the time it takes to download the file. Then, the cockroach swarm optimization (MCSO) approach was then recommended for calculating a node's load ratio based on a variety of variables that accurately represent the load in real-time. After the MCSO method is run, the most suitable value for a full review is used to give the master node authority to the storage node. To ensure that the desired information is sent to the most suitable nodes, two different techniques have been used. Leveraging available resources in this manner has been more effective than previous approaches. The findings of the tests show that the suggested approach reduces network overhead, response times, and memory requirements. Therefore, the proposed heuristic approach is an encouraging and solid strategy for cloud computing financiers.

In this paper [19], a generalised cloud service facility model that fits within the public cloud category is taken into consideration. For managing the request-response architecture with the least amount of delay, a nature-inspired technique, namely particle swarm optimisation, is proposed. The suggested method outperforms the benchmark methods that have been previously published. The simulation is run on the CloudSim platform, and the outcomes show great agreement with the predictive model.

Wilsonprakash and Deepalakshmi[20] proposed Software-Defined Networking (SDN), Using a BPANN with a DA-LB, which stands for Dynamic Agent-Based Load Balancing, allows for optimal utilization of the resources at hand. To move virtual machines in the data centre network effectively, this load balancing method makes advantage of SDN's global visibility. The findings demonstrate that our SDN load-balancing method increases network efficiency in general and is effective for data transfer. The suggested approach improves resource utilization, helps to speed up processing, forecasts loaded virtual machines under high-load situations. The findings of the migration process are contrasted with those achieved with Multi-Path TCP (MPTCP) and the Heuristic algorithm (HA), and it is apparent that our suggested technique yields quicker migration times than MPTCP and HA.

In this work, Mrhari and Hadi [21] Introduce SASPSOLB, a cutting-edge evolutionary strategy that is based on the theory of games and modified particle swarm optimization, with the goal of achieving workload equilibrium in cloud computing. The problem of load balancing has been characterized as an optimization assignment with restricted scope. They apply an equilibrium Nash notion to minimize the anticipated reaction time in order to identify the best solution or one that is almost as good as the best solution. In addition, they take into consideration the fact that users are believed to be participating a noncooperative game, which further reduces the anticipated response time. In this study, a method based on genetic optimization is compared to the methodology

that was really used. Their method performs better than others as far as expected response time and Makespan, based on the simulation results.

Sudhakar et al. [22] This research proposes a solution for load balancing that has shorter response times and takes use of honey bee optimization. According to the results of testing conducted on common data sets, the newly developed request harmony honey bee algorithm demonstrated a considerable improvement over its predecessor by outperforming it in terms of response time by a margin of 5.2% to 9.7% and in terms of processing time by a margin of 13.8% to 68.2% respectively.

In this paper [23], The cloud layer, the fog layer, and the user's layer are the three proposed architectural tiers. Cloud technology and fog offer virtual machines (VMs) that make it easy for users' apps to run quickly. Binary Particle Swarm Optimization (BPSO) is implemented as a way to achieve an even distribution of queries among cloud and fog VMs, and the Genetic Algorithm, or GA, is presented as a solution to the problem. The GA (Generative Algorithm) is proposed as a method for fixing the issue. The effectiveness of the proposed and developed algorithm is evaluated relative to the prevalent PSO and BAT algorithms. Response time (RT) and processing time (PT) may be improved by implementing the Closest Data Centre (CDC), Optimize Response Time (ORT), and Reconfigure Dynamically with Load (RDL). In addition, these factors are used to decide which Data Centre (DC) will get a given request. In order to cut down on computational expenses and speed up both the RT and PT of DCs, the recommended GA and BPSO are put to use.

Kanthimathi and Vijayakumar [24] In the suggested approach, more virtual computers were added using a genetic technique to ensure the most effective virtual machines may be assigned to serve the requests. Requests might be handled quickly and effectively with the allocation of the finest virtual machines. The load might be balanced utilizing the ant colony optimization approach if certain virtual machines were experiencing excessive demand during execution. With the aforementioned method, idle or weakly laden virtual machines would share the additional burden. By turning off the virtual computers once they finish their task or while they are idle, on the other hand, the total energy usage is optimized.

A. Existing Research Gaps

The following areas of research need have been identified in regard to load balancing for the usage of cloud computing, as determined by a survey of the existing literature:

Despite the fact that multiple load-balancing methods have been proposed for usage in cloud computing, there is a need for more detailed studies of various load-balancing methods under diverse conditions. The generalizability of the conclusions of many researchers is compromised by their comparison of just a small number of methods.

A static environment with a consistent workload and resource pool has been used for the overwhelming majority of cloud-based load-balancing research. The workload and the resources' availability, however, might really alter over time. If load-balancing solutions function in highly changeable environments, more investigation is required.

Despite hybrid clouds' increasing popularity, there remains a dearth of studies on their load-balancing strategies. Because hybrid clouds include resources from both private and public clouds, further research is needed to create load-balancing solutions that are appropriate for their complicated nature.

The energy used by cloud computing might be drastically decreased with load balancing. However, there have only been a few studies that have looked at the energy effectiveness of load-balancing techniques. To completely comprehend how different load-balancing techniques, impact the general energy efficiency of cloud computing, further study is necessary.

In order to enhance system efficiency, load balance, and optimization are frequently utilized in conjunction. However, there is a dearth of studies on how well these techniques work in conjunction with cloud computing. It is necessary to do further research on load balancing and optimization techniques that could combine to improve cloud performance and efficiency.

III. Task Scheduling Parameters

Various scheduling methods take into account a number of scheduling factors [4]. In this part, we are

going to discuss these factors for comparing Task Scheduling methods.

A. Execution Time

Execution time is the amount of CPU time or burst time that a computer's system needs to finish a job, in addition to the amount of time needed to provide system amenities.

B. Response Time

Time is taken for a service request to be fulfilled by a computer system. Furthermore, the service being used might be anything from retrieving data from memory to doing a sophisticated database query to loading an entire web page. The system should respond as quickly as possible.

C. Makespan

The sum of the time required for finishing a set of activities. The sum of the execution times for all jobs is known as the makespan. For the system to function more effectively, the scheduler's makespan must be as brief as feasible.

D. Throughput

Any system's performance can be measured by how many operations it completes in a given amount of time. In the most general sense, flow is the pace that something can be handled. The main goal of planning algorithms is to make the system work better as a whole.

E. Resource Utilization

In contrast to response time and output, utilization of resources is another way to measure the performance of a system. How many machine resources are being used? is kept track of using resource usage. The scheduling approach should optimise resource utilisation.

F. Load Balancing

In computing, load balancing relates to a method for distributing tasks evenly among a collection of nodes, such as a server farm or a group of servers connected by a network of nodes. A load balancer improves the system's reliability and concurrent capacity for users by distributing the workload over several servers and so lowering the total load encountered on any one server.

G. Fault Tolerance

The ability to continue operating successfully in the event that one or more of a procedure's components fails is known as fault tolerance. It's crucial to use this metric to assess a system's capacity.

H. Energy Consumption

Energy utilization is a measurement of the amount of raw energy utilized to produce the outcome. It is best to use as little energy as possible. Computers, storage units, communication network tools, and client displays are only a few of the information and communication technology tools required in cloud computing environments. In light of this, it is clear that a rapid increase in ICT energy consumption will be a result of widespread adoption of cloud computing services.

I. Scalability

It is an attribute that describes the way a framework, model, or method can organize to take part in activities that do not become bigger or bigger. When used in scenarios with higher practical demands, a scaling approach may be capable to preserve or even improve its degree of efficacy.

J. Performance

The completion of a given job is measured against standards that have already been set for accuracy, cost, accuracy, and speed. In computers, efficiency is measured by how long it takes and how much it costs. A system should be able to finish a user's job in the least amount of time and at the lowest cost. By using a scheduling method, both the customer and the service provider should think about speed.

K. Quality of Service

Best of provider looks at user contact boundaries, such as making dates, efficiency, execution cost, make spread, etc. SLAs, or Service Level Agreements, are agreements between cloud consumers and cloud service suppliers that lay out all of the specifics.

IV. Various Task Scheduling Algorithms

In this part, we address different Task schedule methods and show the pros and cons of the related schedule factors.

A. Multi-Objective Task Scheduling Algorithms

Most schedule systems try to reduce the make time as much as possible without taking into consideration the cost of services. These methods are additionally employed in settings where there is only one cloud. In a multi-cloud setting, the method described in [25] takes into account both the price and how long the service will last. It also looks at how often tools in the cloud are used.

Profit Based Task Scheduling (PBTS) and Cloud Min-Min Scheduling (CMMS) focus only on a single metric. Both of them only look at one number. Experiment findings demonstrate that the suggested approach significantly reduces average resource use while also reducing costs.

B. Multilevel Priority-Based Task Scheduling

Priority-based thinking was implemented earlier Scheduling algorithms assign tasks based on the relative priority and size of the overall programme. Six Sigma control charts are employed in this suggested scheduling method [26], to rank multiple independent process jobs. The scheduler categorizes jobs into several levels and distributes the resources according to the levels' needs for assistance. The outcomes of the experiment demonstrate how effective this strategy is at handling multilayer tasks in workflows. The execution time and the makespan are effectively decreased by this technique. By selecting only one parameter, the proposed technique focuses primarily on independent workflow activities, but it may be extended to include dependent jobs as well.

C. Load Balancing Task Scheduling Algorithm

Load balancing is often not considered in scheduling strategies in order to save difficulty, which might lead to an uneven load. These techniques do not take into consideration the relationship between the amount of jobs issued and the node's load. The performance of the system will suffer greatly if a job is assigned to a node that is overloaded. The suggested algorithm [27] is built on the weighted random and feedback method, which ensures that nodes with extraordinary performance won't be overloaded and that nodes with average performance have the ability to complete jobs. In the setting of cloud computing, the method effectively equalizes the load between under or over-utilized nodes. Scheduling Work using Particle Swarm Optimization.

This algorithm [28] takes into account the Multidimensional Quality of Service (QoS) standards. The Berger model is utilized with an aim to choose resource allocation outcomes that are equal. By dynamically modifying its boundaries, the method modifies the Particle Swarm Optimization (PSO) algorithm. Then, it supports the QoS-DPSO technique, which is dynamic particle swarm optimization based on quality of service. The experiment's findings demonstrate that the aforementioned algorithm is capable of effectively finishing user activities and allocating system resources in a fair manner. The results of the experiment reveal that although this strategy does not take execution cost and reaction time into account, it may effectively address the task scheduling issue in a cloud computing environment.

D. Energy-Efficiency Based Task Scheduling Algorithm

Task scheduling has gained interest as a technique of minimizing the enormous power consumption of data centers in the cloud centers. The ultimate objective of the technique suggested in this work [9], is to get greater power reduction while guaranteeing quality of service (QoS) via the use of dynamic frequency and voltage scaling (DVFS). The technique distributes parallel tasks and processes onto the right CPU and utilizes them at the ideal timing to achieve the required level of performance while also lowering the amount of power that is used. Experiments demonstrate that the strategy is effective in reducing the power consumption of parallel processes by 46.5% while maintaining a constant scheduler efficiency.

E. Cuckoo Optimization based Task Scheduling Algorithm

When a variety of duties are transmitted to the cloud, the programme is designed to respond to all of these individuals, as this yields the fastest response time. In this paper [29], The author executes one of these guidelines using the Cuckoo software. The suggested approach's goal is to arrange the moving parts in such a way that jobs may be accomplished with little exertion. The Cuckoo algorithm employs a wide variety of simulated devices and duties. By looking at many different orders of these machines, the suggested method finds the best way to assign hosts to tasks. The Cuckoo method is becoming more and more popular since it requires less computation and the least amount of reliance on input data. Levy flying makes it possible to do simultaneous broad and local searches. The processing time of the system is reduced by this approach's usage of the Cuckoo algorithm for load-balancing computations in the cloud. In this method, the birds' population is correlated with the order of the computing units, and each levitation flight changes the work from one computer to another. The Cuckoo approach has been shown in simulations to have the potential to increase performance

characteristics by organizing the processing units in the most efficient way.

F. Green Energy-Efficient based Task Scheduling Algorithm

The suggested eco-friendly energy-saving scheduling approach [30] is based on the prioritized Task Scheduling algorithm. Tasks running on virtual machines are prioritized in light of their relevance and the user's desired Service Level Agreement (SLA). This strategy guarantees the bare minimum of resources and restricts a task's resource overuse. The server's energy consumption can be lowered when it is not actively processing requests by using dynamic voltage and frequency scaling (DVFS). As a result, this technique may improve resource utilization while reducing server power usage. According to the simulation, this strategy may reduce power dissipation by 23% when compared to the way mentioned in [31].

G. Fault-Tolerant Workflow Scheduling Algorithm

In order to reduce service costs while also meeting workload deadlines, the suggested scheduling approach [32], which chooses activities for performance requirements based on two different estimating models (on-demand and spot instances), has been developed. Even in the face of shifting resource utilisation and the untimely extinction of spot instances, the strategy remains fault-tolerant and uncompromising. This technique employs a hybrid of just-in-time (JIT) scheduling and adaptive scheduling algorithms. Leisure time for each job is calculated by the planner. It is the period between the critical route and the bounding box. This solution adaptively moves user tasks dynamically to on-demand instances when work slack time decreases as a result of system failure or performance deviation. The testing results show that using these heuristics reduces the time required for execution by 70% as contrasted with using only one instance.

H. Adaptive Energy-Efficient Task Scheduling Algorithm

Traditional approaches for requesting tasks in a way that saves energy put a lot of weight on a baseline for steady performance and power use. On the other hand, the randomized cutoff cannot be altered to suit the requirements of the system as well as apps. This renders it hard to predict the manner in which the plan will turn out [33], authors provide a two-phase Dynamic Volt Scaling (DVS) and adaptive task duplication system called Adaptive Energy-efficient Scheduling (AES). It begins by suggesting an adaptable threshold-enabled job replication approach, which in turn determines the optimal thresholds, and then it identifies a variety of tasks on DVS-assisted machines that may reduce voltage, hence reducing system power consumption, anytime any work is idle. The findings of the experiments indicate that this technique has the potential to reduce energy consumption and enhance system performance.

Online Optimization for Preemptable Task Scheduling

In this algorithm [34], The author has made a resource-optimized, flexible schedule method for tasks that can be stopped at any time across many clouds Dynamic Cloud List Scheduling (DCLS) and Dynamic Cloud Min-Min Scheduling (DCMMS)are two different dynamic scheduling methods that they suggested for the resource distribution system. The first algorithms stated in [35] do not take task dependence into account[35]. These dynamic algorithms modify the assigned task set at each scheduling step, ensuring that task relationships are preserved. Avoiding resource contention is much improved by this dynamic approach with up-to-date information of workloads. Through careful local modelling of resources and an understanding of their energy use, cloud systems may significantly cut their power consumption.

V. Research Methods

In this work, we studied two new methods that are used to deal with load balancing and job ordering. The presented methods are Harmony Inspired Search Algorithm (HISA), and Dynamic Improved Harmony Inspired Search Algorithm (DI-HISA) with simulated annealing. Both methods are used with simulated annealing. In this study, we have used a pair of optimization methods to isolate a collection of optimization challenges. Additionally, we explored some other load-balancing task scheduling methods, such as LBA (Load Balancing Algorithm), LB-HEFT (Load Balance-Heterogeneous Earliest Finish Time), and LBMPSO (Load Balancing Technique by Using Modified PSO).

A. Harmony Inspired Search Algorithm (HISA)

Harmony Search (HS) is an approach to searching that is based on how jazz players make up music on the spot[36]. In jazz music, the different players try to change

their pitches so that the general harmony sounds best. Using creativity, they start with some harmony and try to get better harmony. This comparison can be used to find search algorithms, that may be utilized to optimize a function with an objective instead of melodies. In the present scenario, the players represent the choice factors, and the rhythms represent the answers. The HS algorithm produces new solutions by iterating over old solutions and making random changes to them. This is similar to how jazz artists come up with new harmony by improvising. Even though this structure leaves a lot of room for opinion, the basic HS method is always shown in the literature as a diagram, as shown in Fig. 1.

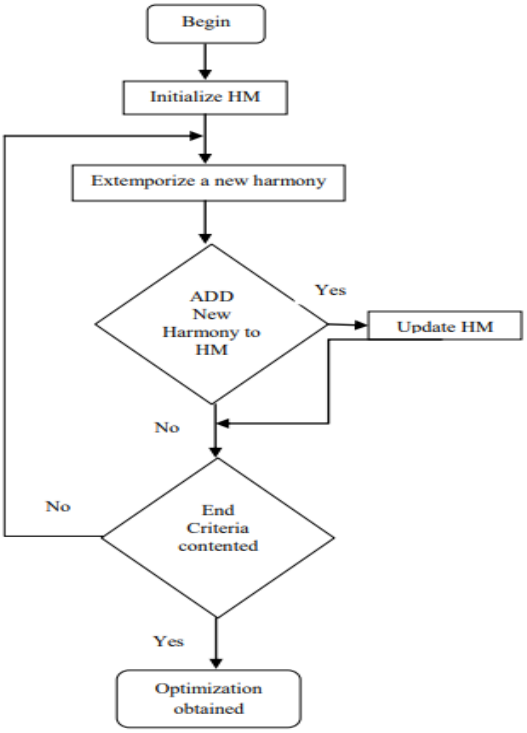

Fig. 1. Flowchart of Harmony Search Algorithm

The HS approach fills the Harmony Memory (HM) with answers that were made at random. The number of answers saved in the HM is determined by the Harmony Memory Size (HMS). Then, a new answer is made step by step, as follows. Each choice variable is made either based on memory and a possible extra change or by picking a number at random. The new solution is calculated using two variables: Consideration Rate for Harmony Memory (HMCR) and Pitch Response (PAR). Each judgement measure is assigned the value of the associated variable in one of the answers in the HM with a likelihood of HMCR, and then this value is changed again with a likelihood of PAR. If not, (with a chance of 1 HMCR), the choice indicator is set to a random number. Following an improvement has been made, it is looked at and matched to the most severe answer in the HM. If it has a higher objective value than the worst solution, it replaces the worst solution in the HM. The process continues until a certain goal is reached[37].

Algorithm 1: HISA-LB Algorithm Pseudocode

Procedure:

1. Start
2. Generate the harmony memory
3. Initialization of all parameters
4. Initialization of Temp
5. Num = 0, ð= 500
6. Place BestX (the best solution in Harmony Memory (HM)) into ƨ
7. BstSol = BstSASol = ƨ
8. For p = (1 to Num) do
9. If (random (0,1) ≤ HM_CR) Then
10. Select 2 vectors solution symbolized as υ1 & υ2 at random from Harmony memory
11. If (random (0,1) ≤ P_A_R) Then
12. υ = put on a PMX crossover to υ1 and υ2
13. newX = the best neighbour amongst some of the neighbours created by υ
14. If (SumCost(newX) < SumCost(worstX)) Then
15. Swap worstX by newX **//Modifying the HM**
16. End of If
17. End of If
18. End of If
19. Else
20. ƨ' = the best neighbor among the generated neighbors of ƨ
21. ΔCost = SumCost(ƨ') – SumCost(ƨ)
22. probability = Random (0,1)
23. If ((ΔCost ≤ 0) or (probability < $e^{-\Delta Cost/Temp}$)) Then
24. ƨ = ƨ', newX = ƨ'
25. If (SumCost(newX) < SumCost(worstX)) Then
26. Swap the worstX by newX **// Modifying the HM**
27. End of If
28. If (SumCost(ƨ) < SumCost(BstSASol)) Then
29. BstSASol = ƨ
30. End of If
31. End of If
32. Temp = Modify (Temp)

33. End of Else
34. If (SumCost(BestX) < SumCost(BstSol))
35. BstSol = BestX, Num = 0
36. End of If
37. If (SumCost(BstSASol) - SumCost(BstSol) ≥ ð)
38. ƨ = BstSol
39. End of If
40. End of For
41. Get result BstSol together with its fitness function value
42. Stop.

B. Dynamic Improved Harmony Inspired Search Algorithm

In Harmony Search (HS) approach, a solution that is likely to work is called harmony, and each of the solution's choice variables is a note. HS has a harmony memory (HM) that has a set amount of harmonies (N) stored in it. Say that the objective is to make the fitness function (f) as small as possible or as big as possible, given d choice factors. Here's how to describe this optimization problem:

$$Min\ (or\ Max) f = (x_1, x_2, .. x_d) \quad (1)$$

Where f is the fitness function, *xi* is the choice variable I (i = 1, 2..., d), and d is the problem size.

For this research, the fitness function is calculated by eq. (2)

$$Fit\ =\ T_{wait} + Makspan + RU \quad (2)$$

Where Fit = fitness function of Dynamic improved HISA-LB, T_{wait} = waiting time. Now is the time to wait for the work to be assigned to a separate virtual machine, RU = Resource utilization.

Algorithm 2: DI-HISA Load Balancing Approach

Pseudocode:

1. Begin
2. Generate the harmony memory
3. Initialization of all parameters, tmp (temperature), Num (t) = 0, T_{max}= 5000
4. Calculate the al parameters values
5. Evaluate the fitness function by formula

$$Fit\ =\ T_{wait} + Makspan + RU$$

6. Place BestX (the best solution in Harmony Memory (HM)) into δ
7. BstSol = BstSASol = δ
8. For p = (1 to t) do
9. Update the HMCR by given formula

$$HM\_CR\ (t)\ =\ HM\_CR_{max} - \frac{(HM\_CR_{max} - HM\_CR_{min}) * t}{T_{max}}$$

10. If (random (0,1) ≤ HM_CR) Then
11. Select 2 vectors solution symbolized as υ1 & υ2 at random from Harmony memory
12. Update PAR by dynamic change strategy by given formula

$$P\_A\_R(t) = \frac{(P\_A\_R_{max} - P\_A\_R_{min})}{\frac{\pi}{2}} * \arctan \mathrm{t} + P\_A\_R_{min}$$

13. F_W changes dynamically with number of iterations by given formula

$$F\_W(t) = F\_W_{max} - \frac{(F\_W_{max} - F\_W_{min}) * t}{T_{max}}$$

14. If (random (0,1) ≤ P_A_R) Then
15. υ = put on a PMX crossover to υ1 and υ2
16. newX = the best neighbour amongst some of the neighbours created by υ
17. If (SumFit(newX) < SumFit(worstX)) Then
18. Swap worstX by newX **//Modifying the HM**
19. End of If
20. End of If
21. End of If
22. Else
23. δ' = the best neighbor among the generated neighbors of δ
24. ΔFit = SumFit(δ') – SumFit(δ)
25. probability = Random (0,1)
26. If ((ΔFit ≤ 0) or (probability < $e^{-\Delta \mathrm{Fit}/tmp}$)) Then
27. δ = δ', newX = δ'
28. If (SumFit(newX) < SumFit(worstX)) Then
29. Swap the worstX by newX **// Modifying the HM**
30. End of If
31. If (SumFit(δ) < SumFit(BstSASol)) Then
32. BstSASol = δ
33. End of If
34. End of If
35. tmp = Modify (tmp)
36. End of Else
37. If (SumFit(BestX) < SumFit(BstSol))
38. BstSol = BestX, t = 0
39. End of If
40. If (SumFit(BstSASol) - SumFit(BstSol) ≥ T_{max})
41. δ = BstSol
42. End of If
43. End of For
44. Get result BstSol together with its fitness function value
45. Stop.

Output: Dynamic task allocation

C. Load Balancing Algorithm (LBA)

This section addressed an approach to dynamic load balancing was discussed with the end objective of reducing makespan and increasing overall resource utilization in cloud settings. Following the creation of n jobs with randomly generated task lengths ranging from 20000 MI to 400000 MI, we established m heterogeneous virtual machines. The processing power of a virtual machine varies from one instance to the next based on factors such as its processor speed (in MIPS), memory, and so on. When the task queue is empty, the procedure for bubble sorting is used to assemble the tasks and virtual machines in descending order according to task length as well as processing speed, and each job is then allocated to a virtual machine using the FCFS principle. The assigned task IDs have been generated in an array that was the identical size as the machine being used for each virtual machine. After giving the task to a virtual machine, start the load-balancing process, check the current state for every VM, and start keeping track of the virtual machine's status. Using the equation, you can figure out the load on every single virtual machine and the overall load placed on the data Centre at the time in question. The last step is to figure out what every virtual machine and data in the centre can do. Next, begin to determine if the anticipated demand on each virtual machine is less than its capacity, as well as whether the anticipated strain on the data center is lower than its capacity. In the event that the supposition is accurate, load balancing in the current cloud architecture is feasible. Find out how many virtual machines are loaded properly, not at all, or both. We established the virtual machine's threshold value as follows: the virtual machine is considered underloaded if its capacity is used at a rate below 25% of its maximum level, and it is considered overloaded if it is utilized at over 80 percent of its maximum level. We created two variables, UM and OM, which stand for 25% and 80% of a virtual machine's capacity, respectively. UM_VM and OM_VM stand for the total amount of underloaded and overloaded virtual machines, respectively. Find the UM_VM and OM_VM files, sort them in either descending or ascending order and then look at the results to get started moving work from a virtual machine that is overloaded to a simulated computer that has less work. The amount of period required for a task to be transferred from a particular virtual machine to a different one is added to the amount of time it takes to do the work in its entirety as well as the makespan of the task. To calculate task transmission time, divide the duration of the transmitted job by the bandwidth between virtual machines. See what happens when you use the load-balancing algorithm. On an already-existing virtual machine, load balancing cannot be performed if this condition is false. Horizontal scaling is the idea that the cloud resource provider has to start up a new virtual machine. In step 13 of the suggested method, we have provided the following circumstances when the predicted demand at the data center exceeds the data center's capacity: If 20 to 50% of virtual machines are overcrowded and no virtual machines are underloaded, It is the responsibility of the cloud resources broker to deploy 20% more virtual machines. If more than half of the virtual machines are overcrowded, the cloud resource brokers should deploy an additional 30% of virtual machines. The state of some virtual machines may change from "balanced" to "underloaded" if the cloud resource broker notices that the volume of upcoming work has decreased. At that point, the cloud resource broker should scale back the number of virtual machines to correspond with the algorithm's conditions[38].

Algorithm 3: Load Balancing Algorithm

Task scheduling Operation

1. Generate n task $T_1, T_2, T_3 \ldots T_n$
2. Sort the task in decreasing order based upon their length
3. Create m diverse virtual machines and sort them by processing speed in decreasing order.
4. For loop for $\forall\ T_i \in 0$ 0 to n-1 and for all virtual machine $R_j \in 0$ to m-1.
5. If $(T_1 \neq \emptyset)$ then allocate in FCFS (First Come First Serve) order End VM for loop and End task for loop

Load Balancing Operation

6. Start the loop for VM_j and T_i
7. Equations 1 to 4 may be used to determine the workload at a virtual machine and data center.
8. If(TL_{vm} >DC), go to step 13, otherwise
9. Determine the total number of UM (UM_{VM}) and OM virtual machine (OM_{VM})

UM_{VM}=.25* CVM_j, OM_{VM}=.8* CVM_j,

10. Sort the underloaded (UM) and overloaded (OM) virtual machines in ascending and descending order,

respectively.

11. While $\mathrm{OM!} = \emptyset \;\&\&\; \mathrm{UM!} = \emptyset$
    For loop for OM
    If OM & UM exist then transfer the task from OM to UM $T_i \rightarrow VM_j$ | Until load at $VM_j \leq OM \;||\; VM_j \geq UM$
    Likewise, determine the task transfer time. (TL_i/bandwidth).
12. After the tasks have been moved about, check to see whether any of the virtual machines are still in an Overwhelmed condition.
13. Calculate total number of OM (OM_{VM}), if (n/10< $OM_{VM} < n/3$) then boot 20% new virtual machines otherwise boot 30% new virtual machines and distribute the workload to all virtual machines fairly.
14. After increasing the VM continuously check the condition ($TL_{vm} > DC$) if yes, repeat the 13 steps. *a)*
15. Calculate the total number of UM (UM_{vm}) if ($\mathrm{n}/5 < UM_{vm} < n/2$) then reduce 20% VM, otherwise reduce 30% VM and repeat the step until load is balanced.

D. Load Balance-Heterogeneous Earliest Finish Time (LB-HEFT)

A. The HEFT algorithm [36, 37] is considered the most
B. popular scheduling algorithm for a heterogeneous envi-
C. ronment due to its good performance with respect to
D. makespan and simplicity. According to the HEFT algo-
E. rithm, the tasks presented in a DAG are scheduled onto a
F. range of heterogeneous machines. This algorithm consists
G. of two phases; Rank and Processor Selection phases. The
H. Rank phase aims to provide a priority for each task. The
I. Processor Selection phase is concerned with allocating a
J. suitable processor for each task.

The HEFT algorithm [39][40] is an extremely often used scheduling method in a heterogeneous setting because to its excellent makespan performance and minimal complexity. The DAG's jobs are assigned to a variety of heterogeneous computers using the HEFT method. The Rank and Processor Selection stages of this technique are its two distinct parts. To assign a priority to each work, the Rank phase is used. To choose the best processor for each job, go to the Processor Selection step.

In the context of cloud computing, the objective of the Suggested Load Balance-HEFT (LB-HEFT) approach is to circumvent the restrictions imposed by the already implemented HEFT algorithms (i.e., load balance). These limits have to do with the amount of time and resources that can be used. The task Ranking stage and the Task-VM Matching track are the two parts of this process. In the section that is entitled "Tasks Ranking," each individual piece of work is given a place in the ranking. During the phase that is referred to as "Task VM Matching," the duties are apportioned among the suitable virtual machines (VMs), with the distribution of load and the earliest feasible execution time being taken into account. This stage also helps ensure the most efficient use of the resources that are available.

Tasks Ranking Phase

Each step in the given process DAG will be ranked on the heterogeneous cloud platform according to the results of the Tasks Ranking phase. The given assignments are of the form $T = [T_1, T_2, \ldots., T_n]$. If $T_1 < T_n$, then T_i is parent of T_n. Adding the task length (TL), which here stands for the amount of cloudlet instructions to be run by the virtual machine, has resulted in a little adjustment to the original Rank equation. A higher rank value for the job indicates that it has a greater difficulty level. The job with the higher-ranking value (ti) will be executed more quickly to decrease the makespan, as shown in equation (3).

$$Rank(t_i) = \overline{W_i} + max_{tj \epsilon succ(t_i)} \left(\bar{C}_{i,j} + Rank(t_j)\right) + T_L \quad (3)$$

The jobs should then be put into the Rank [T] list following their Rank values are put in decreasing order. The suggested LB-HEFT algorithm's job ranking pseudo-code is shown in Algorithm 3.

Algorithm 3: Tasks Ranking of LB-HEFT Algorithm

Input: Automation process and VM collection
For every task t_i in DAG of the workflow **Do**
Identify the average running time of t_i for each VM.
If task t_i is the last task in DAG **Then**
Rank (t_i) = average execution time on all VMs
Else
$Rank(t_i) = \overline{W_i} + max_{tj \epsilon succ(t_i)} \left(\bar{C}_{i,j} + Rank(t_j)\right) + T_L$
End If
End For

9. Set up a list of jobs so that their Rank [T] numbers go from best to lowest.

b) Task-VM Matching Phase

To figure out which virtual machine, or VM, is optimal for every task in the Rank [T] list, we calculate the virtual machine's initial time (ST(VM)) using Equation (4). In a ST[VM] list, arrange virtual machines (VMs) in order of decreasing of their ST(VM) values. The Task-VM Matching Phase provides priority to the virtual machine (VM) that has the earliest planned start time (EST(VM)) and the fewest number of tasks. This is done so that the workload may be distributed more evenly among all of the available virtual machines (VMs).

In contrast to the HEFT method, which prioritizes the virtual machine (VM) with the earliest end time, this phase focuses on the VM with the later anticipated beginning date for carrying out the work; this VM should also finish the fewest tasks.

$$ST(\text{VM}) = FT_{vm} + CC_{(i,j)} \quad (4)$$

where FT_{vm} is the total completion time of all tasks in the virtual machine and, $CC_{(i,j)}$ is the cost of communicating between task t_i from the Rank [T] list and the very last job in the virtual machine, t_j. The pseudo-code for Task-VM matching in the suggested LB-HEFT algorithm is shown in Algorithm 4.

Algorithm 4: Pseudo-code for the proposed LB-HEFT algorithm's Task-VM matching

Input: Rank [T] list, and ST [VM] list

1. vm_k = first component in ST[VM] list
2. While (Task t_i) in Rank [T] list does not belong to a VM.)
3. **Do**
4. **If** (vm_k has smallest no. of tasks)
5. Chosen VM = vm_k
6. vm_k. total assigned tasks ++
7. Update (ST [vm_k] list)
8. Arrange the list using an ascending sort (ST[VM]).
9. **Else**
10. Determine the earliest available starting time and use it vm_{k+1} from ST[VM] list
11. **End If**
12. **End While**

E. Load Balancing Technique by Using Modified PSO (LBMPSO)

This section discussed a model that, as shown in Fig. 2, maximizes resource allocation while reducing execution time. In this model, PSO is used for mapping in order to zero in on the allocation of resources approach of never leaving any jobs in buffer and maximizing total execution time. In order to enhance the makespan and dispersion of resources, algorithms must be enhanced. the overall structure is broken up into a variety of buffers that contain a specific sort of task and resource data[41].

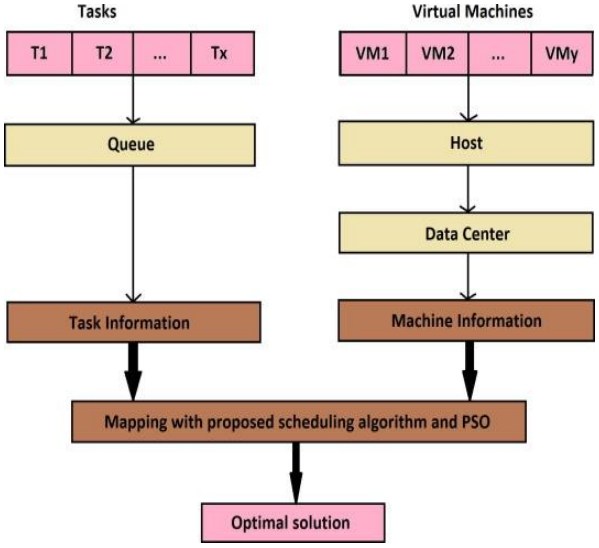

Fig. 2. LBMPSO System

The model schedules the anticipated number of tasks that can be completed using the current cloud infrastructure and its fitness score. The fitness of each particle is then calculated. Alterations are made to the values of Pbest and Gbest while the programme runs. When the present fitness value of a particle is compared to its Pbest value, the optimal placement of the particle is found. Similar to how it analyses the current fitness measurements for the whole population to get the best fitness value, Gbest position, this method also displays this information. The resource manager checks in on the health of the virtual machines and updates the task scheduler on its status. After that, the scheduler of tasks will decide whether the work should be assigned to a VM while still maintaining the VM's load balancing.

VI. Comparative Results

This section describes the comparison results with different novel methods with our proposed methods i.e.,

HISA and DI-HISA load balancing algorithms. The suggested load-balancing methods are compared with other known methods like the LBA, LB-HEFT, and LBMPSO algorithms in terms of makespan with 50 jobs and 5 virtual machines.

TABLE I. PERFORMANCE COMPARISON WITH DIFFERENT LOAD BALANCING APPROACHES

References	Approaches	Makespan (ms)
Kumar and Sharma [38]	LBA	1096
Mahmoud et al. [42]	LB-HEFT	234
Pradhan and Bison [41]	LBMPSO	130.23
Tiwari and Bhatt (Proposed-I) [37]	HISA-LB	99.7
Proposed-II	DI-HISA-LB	84.87

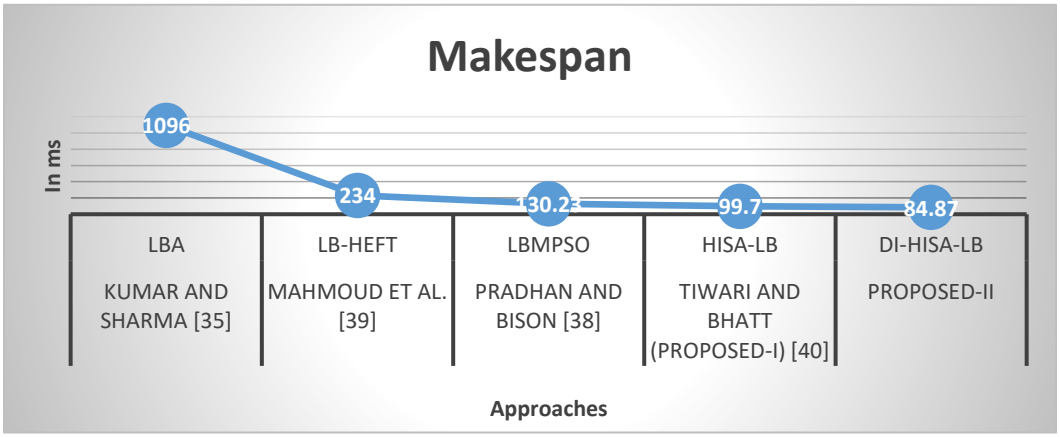

Fig. 3. Comparative Results for 50 Tasks on 5VMs

Table 1 and Figure 3 exhibit the contrasting outcomes of the performance using our suggested techniques for task 50, which compares the various ways based on their makespan. On this graph, the x-axis shows the total number of approaches, and the y-axis shows the number of milliseconds in the makespan. According to these results, LBA method is conducted by Kumar and Sharma in [38]. The makespan achieved by this approach is 1096 milliseconds which is highest. LB-HEFT method is conducted by Mahmoud et al. in [42]. The makespan achieved by this approach is 234 milliseconds. LBMPSO method is conducted by Pradhan and Bison [41]. The makespan achieved by this approach is 130.23 milliseconds. HISA-LB method is conducted by me and this is our first proposed method. The makespan achieved by this HISA-LB approach is 99.7 milliseconds, and the second proposed method is DI-HISA. This proposed method achieved only 84.87 milliseconds makespan, respectively. It is clearly shown that our both proposed methods obtained better outcomes than the other existing methods. But the DI-HISA (proposed-II) method outperforms the others.

VII. Conclusion and Future Work

Load balancing is a critical operation in online computing for optimizing resource utilization. With the introduction of the cloud computing environment, several experiments for determining and presenting the optimal load-balancing algorithm have been presented. We addressed task scheduling strategies and performance parameters within this study. Static load balancing systems, on the one hand, provide for the most straightforward modelling and tracking of the surroundings, but on the other, they have difficulty mimicking the diverse characteristics of cloud computing. Some studies attempted to modify unique and traditional load-balancing algorithms such as HISA, DI-HISA, LBA, LB-HEFT, and LBMPSO. These algorithms are compared with the proposed methods to evaluate their performances i.e., makespan over tasks 50 and 5 virtual machines. The experimental results validate the DI-HISA algorithm's advantage over its competitors.

The primary challenges in task scheduling are load balance, response time, resource utilization, and memory storage. Adding new parameters to existing scheduling algorithms may improve their general efficacy

in a cloud environment, opening the door to the development of more effective algorithms.

References

[1] P. Pradhan, P. K. Behera, and B. N. B. Ray, "Modified Round Robin Algorithm for Resource Allocation in Cloud Computing," 2016, doi: 10.1016/j.procs.2016.05.278.

[2] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *Journal of King Saud University - Computer and Information Sciences*. 2020, doi: 10.1016/j.jksuci.2018.01.003.

[3] K. V. Reddy *et al.*, "Research Issues in Cloud Computing," *Online*, 2011, doi: 10.1109/cise.2010.5677076.

[4] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," 2011, doi: 10.1109/SERVICES.2011.105.

[5] M. U. Bokhari, Q. M. Shallal, and Y. K. Tamandani, "Cloud computing service models: A comparative study," 2016.

[6] Z. Mahmood, "Cloud Computing: Characteristics and deployment approaches," 2011, doi: 10.1109/CIT.2011.75.

[7] R. Buyya, C. Vecchiola, and S. Thamarai Selvi, *Mastering cloud computing: Foundations and applications programming*. 2013.

[8] N. Jain and S. Choudhary, "Overview of virtualization in cloud computing," 2016, doi: 10.1109/CDAN.2016.7570950.

[9] S. Afzal and G. Kavitha, "Optimization of Task Migration Cost in Infrastructure Cloud Computing using IMDLB Algorithm," 2018, doi: 10.1109/ICCSDET.2018.8821193.

[10] R. Achar, P. S. Thilagam, N. Soans, P. V. Vikyath, S. Rao, and A. M. Vijeth, "Load balancing in cloud based on live migration of virtual machines," 2013, doi: 10.1109/INDCON.2013.6726147.

[11] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes, "Workload modeling for resource usage analysis and simulation in cloud computing," *Comput. Electr. Eng.*, vol. 47, pp. 69–81, 2015, doi: https://doi.org/10.1016/j.compeleceng.2015.08.016.

[12] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing," 2015, doi: 10.1109/C3IT.2015.7060176.

[13] A. Dave, B. Patel, and G. Bhatt, "Load balancing in cloud computing using optimization techniques: A study," 2016, doi: 10.1109/CESYS.2016.7889883.

[14] . P. S., "A SURVEY OF VARIOUS SCHEDULING ALGORITHM IN CLOUD COMPUTING ENVIRONMENT," *Int. J. Res. Eng. Technol.*, 2013, doi: 10.15623/ijret.2013.0202008.

[15] N. Goel and R. B. Garg, "A Comparative Study of CPU Scheduling Algorithms," *Int. J. Graph. Image Process. |Vol 2|issue*, 2012.

[16] S. Lata and D. Singh, "A Hybrid Approach for Cloud Load Balancing," 2022, doi: 10.1109/ICACITE53722.2022.9823569.

[17] A. Mohanty, S. Samantaray, S. S. Patra, M. A. I. Ahmad, and R. K. Barik, "An efficient resource management scheme for smart grid using GBO algorithm," 2021, doi: 10.1109/ESCI50559.2021.9396784.

[18] G. Senthilkumar and M. P. Chitra, "A Novel hybrid heuristic-metaheuristic Load balancing algorithm for Resource allocationin IaaS-cloud computing," 2020, doi: 10.1109/ICSSIT48917.2020.9214280.

[19] H. Rai, S. K. Ojha, and A. Nazarov, "A hybrid approach for process scheduling in cloud environment using particle swarm optimization technique," 2020, doi: 10.1109/EnT50437.2020.9431318.

[20] S. Wilsonprakash and P. Deepalakshmi, "Artificial Neural Network Based Load Balancing on Software Defined Networking," 2019, doi: 10.1109/INCOS45849.2019.8951365.

[21] A. Mrhari and Y. Hadi, "A Load Balancing Algorithm in Cloud Computing Based on Modified Particle Swarm Optimization and Game Theory," 2019, doi: 10.1109/ICoCS.2019.8930807.

[22] C. Sudhakar, R. Jain, and T. Ramesh, "Cloud load balancing - Honey bees inspired effective request balancing strategy," 2019, doi: 10.1109/GUCON.2018.8675112.

[23] A. A. Butt, S. Khan, T. Ashfaq, S. Javaid, N. A. Sattar, and N. Javaid, "A cloud and fog based architecture for energy management of smart city by using meta-heuristic techniques," 2019, doi: 10.1109/IWCMC.2019.8766702.

[24] M. Kanthimathi and D. Vijayakumar, "An Enhanced Approach of Genetic and Ant colony based Load Balancing in Cloud Environment," 2018, doi: 10.1109/ICSNS.2018.8573608.

[25] S. K. Panda and P. K. Jana, "A multi-objective task scheduling algorithm for heterogeneous multi-cloud environment," 2015, doi:

10.1109/EDCAV.2015.7060544.

[26] A. Bala and I. Chana, "Multilevel priority-based task scheduling algorithm for workflows in cloud computing environment," 2016, doi: 10.1007/978-981-10-0129-1_71.

[27] Q. Zhang, Y. Ge, H. Liang, and J. Shi, "A load balancing task scheduling algorithm based on feedback mechanism for cloud computing," *Int. J. Grid Distrib. Comput.*, 2016, doi: 10.14257/ijgdc.2016.9.4.04.

[28] A. Xu, Y. Yang, Z. Mi, and Z. Xiong, "Task scheduling algorithm based on PSO in cloud environment," 2016, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.196.

[29] A. Moradbeiky and V. Bardsiri, "A Novel Task Scheduling Method in Cloud Environment using Cuckoo Optimization Algorithm," *Int. J. Cloud-Computing Super-Computing*, 2015, doi: 10.21742/ijcs.2015.2.2.02.

[30] C. M. Wu, R. S. Chang, and H. Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Futur. Gener. Comput. Syst.*, 2014, doi: 10.1016/j.future.2013.06.009.

[31] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," 2009, doi: 10.1109/CCGRID.2009.16.

[32] D. Poola, K. Ramamohanarao, and R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," 2014, doi: 10.1016/j.procs.2014.05.047.

[33] W. Liu, W. Du, J. Chen, W. Wang, and G. Zeng, "Adaptive energy-efficient scheduling algorithm for parallel tasks on homogeneous clusters," *J. Netw. Comput. Appl.*, 2014, doi: 10.1016/j.jnca.2013.10.009.

[34] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on IaaS cloud systems," *J. Parallel Distrib. Comput.*, 2012, doi: 10.1016/j.jpdc.2012.02.002.

[35] O. H. Ibarra and C. E. Kim, "Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors," *J. ACM*, 1977, doi: 10.1145/322003.322011.

[36] Z. W. Geem and J. C. Williams, "Harmony search and ecological optimization," *Int. J. Energy Environ.*, vol. 1, pp. 150–154, 2007.

[37] C. B. Shruti Tiwari, "HISA-LB: A Novel Load Balancing Approach for Improvising Harmony Memory in Cloud Environment," *Math. Stat. Eng. Appl.*, vol. 71, no. 4, pp. 9029–9055, 2022.

[38] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm to minimize the makespan time and utilize the resources effectively in cloud environment," *Int. J. Comput. Appl.*, 2020, doi: 10.1080/1206212X.2017.1404823.

[39] A. K. Maurya and A. K. Tripathi, "On benchmarking task scheduling algorithms for heterogeneous computing systems," *J. Supercomput.*, 2018, doi: 10.1007/s11227-018-2355-0.

[40] R. Sahal, M. Nihad, M. H. Khafagy, and F. A. Omara, "iHOME: Index-Based JOIN Query Optimization for Limited Big Data Storage," *J. Grid Comput.*, 2018, doi: 10.1007/s10723-018-9431-9.

[41] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *J. King Saud Univ. - Comput. Inf. Sci.*, 2020, doi: 10.1016/j.jksuci.2020.10.016.

[42] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "An efficient load balancing technique for task scheduling in heterogeneous cloud environment," *Cluster Comput.*, 2021, doi: 10.1007/s10586-021-03334-z.