VOL. 19, NO. 14, JULY 2024 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

ENHANCING CLOUD PERFORMANCE THROUGH GREY WOLF OPTIMIZATION: A ROBUST APPROACH TO LOAD BALANCING

Kethineni Vinod Kumar[1], A. Rajesh[1] and R. Balakrishna[2]
[1]Department of Computer Science and Engineering Vels Institute of Science Technology and Advanced Studies (VISTAS), PV Vaithiyalingam Rd, Velan Nagar, India
[2]Department of Artificial Intelligence and Data Science Vel Tech High Tech Dr. Rangarajan Dr. Sakunthala Engineering College, Chennai, India
E-Mail: kethineni.vinod@gmail.com

ABSTRACT

Much progress in computing has resulted from the advent of cloud computing. End users may reap the benefits of a plethora of cloud technologies. Services are accessible through online login only. Load balancing is the cornerstone problem in cloud computing that has stumped researchers. Users are happier and systems are more productive when load balancing is used to distribute tasks evenly across all available CPU cores. Moreover, it would be difficult to maintain a load balance across resources since resources are often spread in a dispersed fashion. By using a met heuristics approach, several load-balancing techniques have sought to optimize system performance. In this research, we apply the Optimization of gray wolves (OGW) technique to balance loads reliably among all available resources. In the first step, the OGW algorithm looks for idle or busy nodes, and then it attempts to determine the threshold and fitness function for each of these nodes. Simulation findings in CloudSim confirmed that the suggested approach yields superior outcomes in terms of cost and reaction time.

Keywords: optimization of gray wolves, mouse customized golden eagle optimization, load-balancing system.

Manuscript Received 13 April 2024; Revised 3 August 2024; Published 12 October 2024

1. INTRODUCTION

One definition of cloud computing is "a paradigm for presenting, consuming, and delivering IT services and other shared network resources in a way that takes advantage of the scalability, flexibility, and economics of the cloud" [1]. Users may utilize programmers from anywhere in the globe thanks to cloud computing and services [2]. With this approach, customers may make use of the service when and where it is most convenient for them, independent of their geographical proximity to the cloud. Commercially successful firms like Google, Amazon, and Microsoft all back the cloud computing paradigm. This help is provided via the use of networked computer hardware and software resources [3]. Modern cloud computing infrastructures provide users with access to a plethora of on-demand, virtual services [4]. As defined by the Oxford English Dictionary, "cloud computing" refers to the delivery of hardware and software through the Internet in exchange for payment from end users [5]. Consumers may get their apps via the internet in the form of applications and have them installed on a cluster of network servers rather than on their local PCs. In other words, people can instantly connect and have access to cloud services only if they have access to a mobile device, computer, etc. [6].

Load balancing is the technique of dividing up tasks across available computer systems in an equitable manner to minimize downtime and maximize production [7]. There is a decrease in system efficiency [8] when certain virtual machines (VMs) have a greater load volume than others. It would limit the cloud's capacity and have an impact on how quickly certain functions might be completed. In addition, the cloud system has to allocate demands among its resources when it is hit with a much larger number of requests. The efficiency and usability of the cloud system's resources are crucial to improving its overall performance [9]. The availability of many computing resources in a cloud system allows it to quickly adapt to the needs of its customers. Hence, a method is needed to pick suitable resources in responding to user requests [10], and doing so correctly requires taking into account the features of tasks.

To overcome connection restrictions and establish QoS standards, this work proposes a strategy for choosing the best services to get optimum QoS [11]. Quality of service (QoS) is a major issue in cloud computing. Optimization of gray wolves (OGW) and quality of service (QoS) have been combined to create the suggested solution. The OGW method, which is used for load balancing, was inspired by the natural behaviour of grey wolves [12], which involves the wolves beginning to look for food in their area. First, the nodes in the present investigation need to choose the cluster leader (CH). Typically, the node with the most neighbors is selected as the CH node. To determine whether or not the virtual machine is overloaded, the threshold of nodes is computed; if the load is less than the threshold, the virtual machine is under loaded. After the alpha and beta wolves have located and evaluated the most promising node, they will launch an assault on the prey and ultimately choose that node as their target. The following is a summary of the findings, which show that this strategy is preferable to others in terms of effectiveness.

Quality of Service (QoS) hybridization that better meets user requirements.

- Improving load balancing by decreasing reaction time and the cost of picking VMs.
- Using several different strategies for preserving load balance, conduct a series of experiments to assess the method's efficacy.

The remaining sections of this article are structured as follows: Part 2 discusses the existing literature, whereas Section 3 details the suggested method. Section 4 displayed the collection of experimental data, and Section 5 presented the outcomes of the analysis. To wrap things up, we have a section dedicated to closing remarks and suggestions for further research.

2. RELATED WORK

The load-balancing method of the non-concentrated artificial bee colony (ABC) algorithm was proposed by Kruekaew and Kimpan [13]. Load balancing in the cloud, it was noted, is either growing or shrinking in response to seasonal changes in demand, which served as inspiration from nature. Users' needs are dynamically balanced by the allotted servers. Each of these machines is a "virtual server," and it has its own "virtual service queue."

Each server, much like a honeybee, dances as it processes a request and demand from the queue to determine the necessary benefits. The time it takes the processor to complete a request is one indicator of such benefits. In this case, the honeybee dance represents the billboard. The server might choose benefits for the possible billboards once a request has been processed. It can also do services (thus the scout behaviour) and look at the commercials (like watching a dance). A server may continue in the same virtual server and become credible advertising if the benefits are assessed and large relative to the total benefits of a colony. The servers act as explorers and scouts if nothing else is specified. The evaluation results showed that the aforementioned strategy shortens both makespan and reaction time. Even yet, the suggested method suffers from high prices and limited dependability. The enhanced particle swarm optimization approach was proposed by Devaraj *et al*. [14]. (PSO). This algorithm has the potential to efficiently provide users with resources that are well-suited to their current endeavors. A simulation approach was included to aid the PSO algorithm. The latter approach was used, and it not only helped prevent PSO from becoming stuck in local optimums but also made the process more efficient overall. The PSO method chooses the particle with the smallest point-to-line distance as the global best (gbest). The best options for particles were chosen by using the smallest distance between two points or two lines. In terms of throughput, dependability, and execution time, this approach performs well. Nevertheless, it isn't very useful for tracking things like energy use and service status.

To achieve load balancing, Lilhore *et al*. [15] suggested an exploratory scheduling approach using the hybrid particle swarm optimization (HPSO) algorithm. Our approach aimed to minimise the longest job completion time across all cloud processors while still maintaining load balancing.

It was expected then that the cloud environment and the many processes running inside it would make use of a variety of resources, each with its unique processing capabilities. The time it takes to complete a job may thus vary depending on how it is distributed among available resources. The results demonstrated that the approach might speed up resource exploitation while decreasing the duration of operations. The capabilities in terms of dependability and service monitoring, however, are somewhat restricted.

The load-balancing method of the non-concentrated artificial bee colony (ABC) algorithm was proposed by Kruekaew and Kimpan [13]. Load balancing in the cloud, it was noted, is either growing or shrinking in response to seasonal changes in demand, which served as inspiration from nature. Users' needs are dynamically balanced by the allotted servers. Each of these machines is a "virtual server," and it has its own "virtual service queue."

Each server, much like a honeybee, dances as it processes a request and demand from the queue to determine the necessary benefits. The time it takes the processor to complete a request is one indicator of such benefits. In this case, the honeybee dance represents the billboard. The server might choose benefits for the possible billboards once a request has been processed. It can also do services (thus the scout behaviour) and look at the commercials (like watching a dance). A server may continue in the same virtual server and become credible advertising if the benefits are assessed and large relative to the total benefits of a colony. The servers act as explorers and scouts if nothing else is specified. The evaluation results showed that the aforementioned strategy shortens both makespan and reaction time. Even yet, the suggested method suffers from high prices and limited dependability. The enhanced particle swarm optimization approach was proposed by Devaraj *et al*. [14]. (PSO). This algorithm has the potential to efficiently provide users with resources that are well-suited to their current endeavors. A simulation approach was included to aid the PSO algorithm. The latter approach was used, and it not only helped prevent PSO from becoming stuck in local optimums but also made the process more efficient overall. The PSO method chooses the particle with the smallest point-to-line distance as the global best (gbest). The best options for particles were chosen by using the smallest distance between two points or two lines. In terms of throughput, dependability, and execution time, this approach performs well. Nevertheless, it isn't very useful for tracking things like energy use and service status.

To achieve load balancing, Lilhore *et al*. [15] suggested an exploratory scheduling approach using the hybrid particle swarm optimization (HPSO) algorithm. Our approach aimed to minimise the longest job completion time across all cloud processors while still maintaining load balancing.

It was expected then that the cloud environment and the many processes running inside it would make use of a variety of resources, each with its unique processing capabilities. The time it takes to complete a job may thus vary depending on how it is distributed among available resources. The results demonstrated that the approach might speed up resource exploitation while decreasing the duration of operations. The capabilities in terms of dependability and service monitoring, however, are somewhat restricted.

Comparison of Previous Approaches

S. No	Mouse Customized Golden Eagle Optimization (MCGEO)	Optimization of gray wolves(OGW)
t	Mouse Customized Golden Eagle Optimization (MCGEO) model is developed for optimal load balancing, which is a conceptual combination of traditional Golden Eagle Optimizer (GEO) and Cat and Mouse-Based Optimizer (CMBO).	OGW is a nature-inspired optimization algorithm based on the social hierarchy and hunting behavior of grey wolves.
2	Improves convergence and addresses the optimization problems in load balancing.	The algorithm simulates the leadership hierarchy of grey wolf packs, including alpha, beta, delta, and omega wolves.
3	The MCGEO achieved a throughput value of ~281.6255, at 150 tasks for the cloud environment-2 this proves the superiority of the proposed approach.	The optimization process involves updating the positions of a population of candidate solutions based on the hunting and collaboration behaviors of wolves.

The cloud server, when receiving lots of service requests then the server needs to assign load equally and this process is called load balancing. Load balances are of two types, hardware and software load balancer. The hardware load balancer implementation cost is so high therefore these types of load balancers are not popular.

Network Load Balancing: This technique is used to balance the network traffic across multiple servers or instances. It is implemented at the network layer and ensures that the incoming traffic is distributed evenly across the available servers.

3. THE PROPOSED MODEL

A vast number of data centers and users are located in different parts of the globe to create the cloud computing ecosystem [19]. The cloud must efficiently arrange a huge number of user requests and offer them the appropriate services [20]. It is essential that cloud computing equitably distributes workloads among available resources. If the cloud wants to keep its users happy, the workload must be balanced among all of its nodes. In the context of distributed, grid, and cloud computing architectures, several load-balancing methods have been developed. The focus of this research is on preserving QoS-driven load balancing. To reduce reaction time and cost associated with locating suitable resources, a novel QoS-based approach to load balancing has been presented. The suggested technique uses the OGW algorithm to keep QoS-based load balancing in place in a cloud computing setting.

3.1 Statement of the Problem

Let's say you have Cloud c, which consists of n actual computers or any physical machine made up of M VMs:

$$C = \{PM_1, PM_2, PM_n\} \quad (1)$$

Each given host computer hosts several virtual machines in this way:

$$PM_n = \{VM_1, VM_2, \ldots, VM_m\} \quad (2)$$

To begin, VM1 represents the first virtual machine, and VMm is the final one (VM).

Similarly, if users make up the cloud the user's function may be denoted as follows:

$$U_i = \{T_1, T_2 \ldots T_j\} \quad (3)$$

The procedure's primary objective is to keep the load evenly distributed among all VMs in a cloud environment, while also minimizing expenses and energy consumption and optimising resource utilisation and quality of service. Loads need to be balanced for an efficient planning approach to be attained. Without it, the system will use an excessive amount of resources trying to do its job. This research proposes a multi-purpose approach based on the OGW algorithm that is both efficient and fair, allowing it to solve this issue. The suggested load-balancing mechanism is shown in Figure-1.

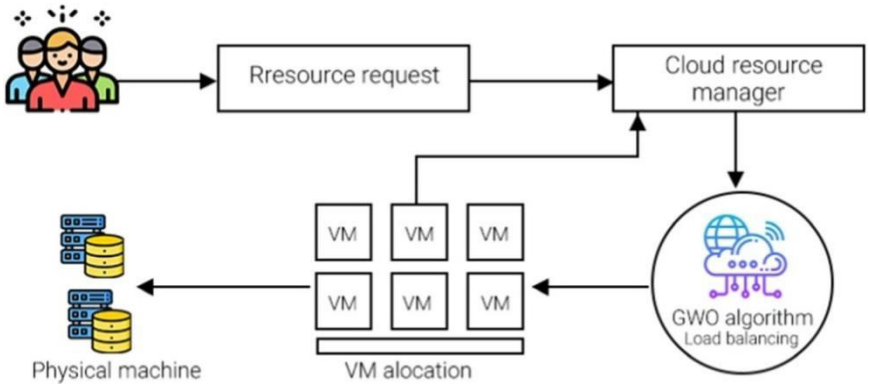

Figure-1. The proposed load-balancing system.

3.2 Virtualization in the Cloud

Using virtualization technologies to merge service providers is the safest option. By the use of virtualization, many virtual machines (VMs) may be run on a single piece of hardware. These virtual machines (VMs) may run independent versions of software [22]. The use of virtualization in this way enables the deployment of many incompatible OSes to run a single programme in parallel. The term "virtualization" refers to the process of making a digital replica of a physical object [23]. This may be an operating system, a computing server, storage, or a network. There is a common misconception that virtualization is the same as cloud computing. Cloud computing relies on several different technologies, one of the most fundamental being virtualization. As compared to virtualization, the cloud's primary goal is resource management [24]. What kind of infrastructure, what kind of environment, and what kind of software as a service are all factors here? Although its primary goal is more efficient use of resources, the term "virtualization" may have a wide range of interpretations depending on the context.

The following are examples of possible applications:

The four main categories of virtualization are server, storage, network, and service.

Figure-2 demonstrates how, with the help of virtualization management, a data centre may host all four types of virtualization-server, network, storage, and service. While other kinds of virtualization are theoretically feasible, their implementation in data centres is still in its infancy.

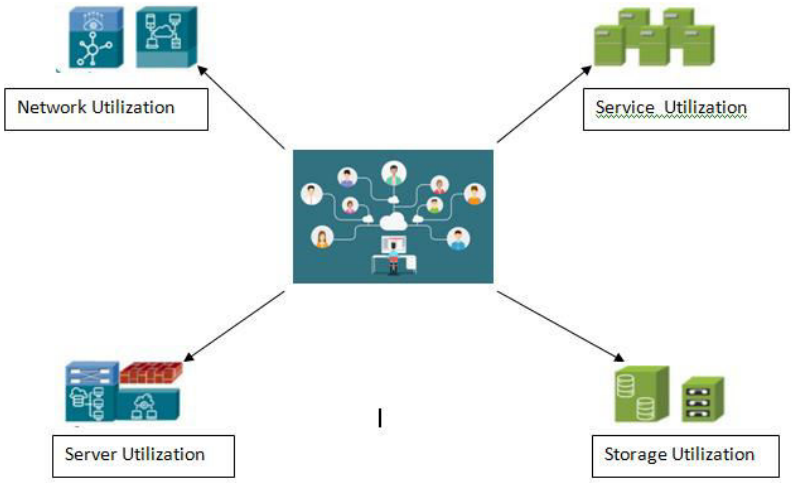

Figure-2. Types of virtualization.

3.3 The Immigration of Virtualization Machines

Virtual machine (VM) immigration is a strategy used to improve cloud data centre resource management [25]. The flexibility and scalability of cloud data centres benefit from this influx of immigrants. In this context, immigration is moving the contents of a computer's hard drive or CPU to another system. Energy management, load balancing, and distribution, reliability in the face of failure, speeding up responses, enhancing service quality, fixing bugs, and maintaining servers are all part of this process. When managing resources, load balancing and server consolidation are two primary motivations for immigration.

Balancing the load, the objective is to avoid excessive resource disparity across data center physical servers by spreading the processing load evenly among them.

Interaction of Servers Optimization techniques are often employed throughout the VM setup process to determine whether a certain physical host is the best choice for a given VM. Hence, reducing the total number of physical hosts helps save money on power bills. In monitoring VM performance and resource use, dynamic resource allocation algorithms make choices about load balancing or combining service providers (Figure-3).

3.4 The Parameters of Service Quality

Timeframe and use of materials it's a look at how long typical tasks take about the total time it takes to finish the processing. Being the polar opposite of utilisation, which prioritizes service quality optimization in the cloud, this metric aims to provide the lowest possible level of service [26]. Utilization of a VM is defined as the proportion of the makespan of jobs when the VM is active (Eq. 4). The following steps were taken to define the issue: members of the collection T1, T2, T3,..., Tn are inconsistent with one another, and the VMs in the form of VM1, VM2,..., VMm, which are all self-sufficient. It is possible to define the overall utilisation rate using the following five equations, where M represents the total number of virtual machines if the time it takes to perform task Ti in VMj is treated as PTij and the finishing time VMj is indicated as CTj. Time spent using resources should be extended as much as possible.

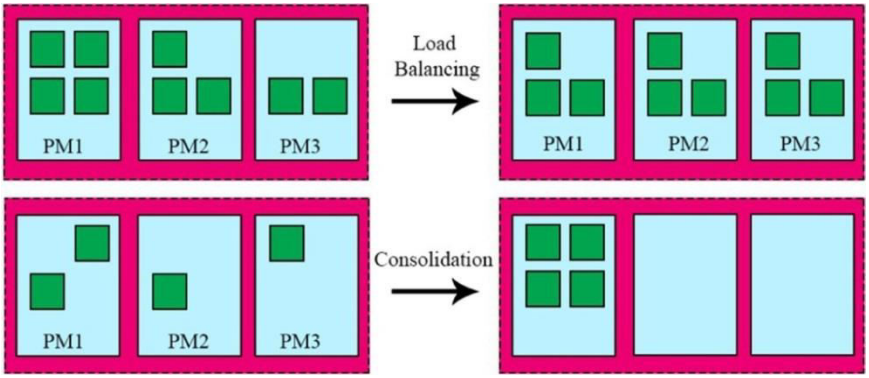

Figure-3. Migration with the aim of load balancing and combining servers.

$$\text{Makespan} = \max\{CT_j | J = 1, 2, \ldots, M\} \quad (4)$$

$$\text{Utilization VM}_i = \frac{\sum_{(i=1)}^{n} PT_{ij}}{\text{makespan}} \quad (5)$$

$$\text{Resourceutilization} = \frac{\sum_{i=1}^{n} \text{Utilization}_{VM_j}}{m} \quad (6)$$

Costs A user's cost per request to a virtual machine (VM) is determined by how many resources (including RAM, CPU, virtual machine, and data transfer) are being utilised. Using Eq. 7 [27], we can calculate this interest rate:

$$\text{Cost} = \sum_{i=1}^{K} C_i * T_i \quad (7)$$

Where K is the number of virtual machines (VMs) allocated to user requests, Ci is the cost per VM, and Ti is the amount of time a user may make use of a VM.

The lag time is measured by the following equation [10] and indicates a service's ability to provide the desired service under any conditions and within the time frame specified.

$$RA_{RK} = \frac{RES_K}{REC_K} \quad (8)$$

In RESk, the amount of work resent to Rk in a certain period, expressed in milliseconds, is shown. More than that, RECk displays the total number of requests that have been gathered.

3.5 Standardizing the Values of QoS Parameters

Several units are used to measure various aspects of the quality of service for various services. In contrast, the objective function must be assessed consistently across all parameters. Thus, it is necessary to standardise the values of all QoS-related parameters on the same scale. In reality, it is feasible to acquire consistent measurements of the QoS metrics by normalising them. For this reason, it is common practice to normalise parameters to a number between 0 and 1. In general, you may divide the stated parameters into two categories: maximisation and minimization. [10] Equations 9 and 10 show the normalisation relationships for the maximum and minimum parameters, respectively.

$$N_{CS.Q}i = \begin{cases} \frac{Q^i_{\max} - CS.Q^i}{Q^i_{\max} - Q^i_{\min}} & Q^i_{\max} \neq Q^i_{\min} \\ 1 \end{cases} \quad (9)$$

$$N_{CS.Q}i = \begin{cases} \frac{CS.Q^i - Q^i_{\min}}{Q^i_{\max} - Q^i_{\min}} & Q^i_{\max} \neq Q^i_{\min} \\ 1 \end{cases} \quad (10)$$

The ith parameter's value in the candidate service CS is denoted by CS. Qi in the aforementioned equations, whereas its normalised value is denoted by Ncs. Qi. Moreover, the maximum and lowest values of the ith parameter across all services are denoted by Qi max and Qi min, respectively.

3.6 Fitness Function

Dealing with user-imposed constraints, locating available nodes, and allocating work to them are all crucial parts of the QoS-based load balancing challenge, as is optimising a fitness function. The quality of service (QoS) characteristics for the new merged services must be optimised using the fitness function. The tendencies of the effects of the positive and negative parameters on the evaluation function are opposed to one another. To solve this issue, we must normalise all QoS factors (using the method described in the previous section) such that their positive and negative effects converge. Using Eq. 11, we can define the solution's fitness function, where W is a positive value that shows the significance of each QoS-related parameter as chosen by the users.

$$\text{Fitness} = \sum_{i=1}^{4} W_i * Q_i \quad (11)$$

3.7 The Optimization of Gray Wolves Algorithm

To improve the effectiveness of wolf hunts, this paper presents an updated version of the Grey wolf meta-heuristic algorithm [28]. Grey wolves, who belong to the family of coyotes, have inspired this algorithm [29]. The alpha () of a pack of grey wolves may be either a man or a female. The alpha wolf, or pack leader, makes all the important choices, such as when to sleep and when to get up, and gives commands to the other wolves in the pack. Wolf alphas have innate abilities that allow them to lead and keep watch over their packs (Table-1).

The grey wolf is the pack's second-highest-ranking member after the alpha wolf. Beta wolves, whether male or female, hold the position of second in command in Grey wolf packs. The pack's alphas make the calls, while the pack's betas help get the word down to the pack's lower ranks. Furthermore, the alpha wolves rely on the beta pack for information and insight. One of the beta wolves will take over as alpha if the leader dies. The delta () wolves, who are third in rank, are divided into four groups: spies, guards, hunters, and supporters. The delta wolves assist keep the rest of the pack safe by patrolling the perimeter and alertly responding to threats. Delta guards care after for the pack's elderly, frail, and ailing members, while Delta hunters ensure everyone has plenty to eat. When a beta wolf dies, the most senior delta wolf is elevated to beta rank. In OGW, alphas are seen as the optimal strategy. The omega comes in last, followed by the delta, and then the beta.

Table-1. Stages from discovering until exploiting the game.

First stage	Detecting, following, and approaching the game
Second stage	Proceeding, surrounding, and harassing the game
Third stage	Attacking the game

Grey wolves hunt by surrounding and encircling their prey in a circling pattern.

In the following, we offer the mathematical model of this behaviour.

$$D = |C.X_p(t) - X(t)| \quad (12)$$

$$X(t+1) = X_p(t) - A.D \quad (13)$$

Where T represents the current iteration, A and D are efficient vectors, Xp represents the hunting area, and X represents the grey wolf's movement.

$$A = 2a.r_1 - a \quad (14)$$

$$C = 2.r_2 \quad (15)$$

A pack's alpha is the dominant male member. Betas and deltas will sometimes go on such a search. Updated locations are calculated using the number of surviving omega wolves and the top three solutions are stored. The following formulas describe how such activities are carried out:

$$D_a = |C_1 * X_a - X| D_\omega = |C_2 * X_\omega - X| \quad (16)$$

$$D_\delta = |C_3 * X_\delta - X| \quad (17)$$

$$X_1 = X_a - A1 * (D_a) * X_2 = X_\omega - A2 * (D_\omega) \quad (18)$$

$$X_3 = X_\delta - A3 * (D_\delta) \quad (19)$$

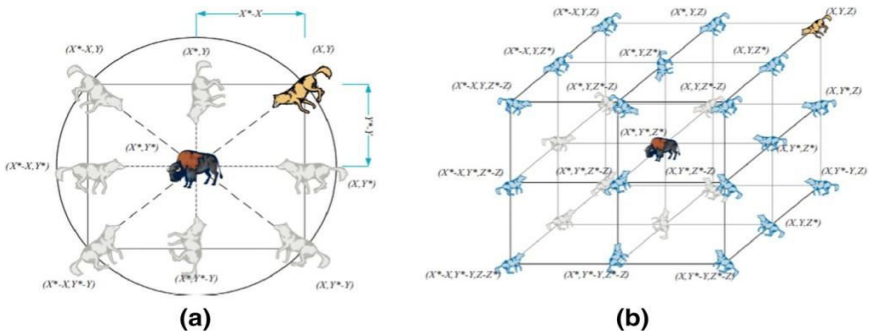

Figure-4. The location and placement vector of D3 and D4.

$$X(t+1) = \frac{X_{1+X_{2+X_3}}}{3} \quad (20)$$

To find the optimal solution in an N-dimensional search space, we may use a strategy like to that seen in Figure-4, in which grey wolves roam in cubes. The OGW algorithm is shown in pseudo-code 1 below.

The pseudo-code of the OGW algorithm 1

1 Set the initial values of the population size n, parameter a, coefficient vectors A and C, and the maximum number of iterations maxiter
2 Set t=0
3 for (i=1: n) do
4 Generate an initial population Xi (t) randomly
5 Evaluate the fitness function of each search agent (solution) f (Xi)
6 End for
7 Assign the values of the first, second, and third best solutions Xα, Xβ, and Xδ, respectively
8 Repeat
9 for (i=1: n) do
10 Update each search agent in the population as shown in Eq. (20)
11 Decrease the parameter a from 2 to 0
12 Update the coefficients A, and C as shown in Eq. (14) and (15), respectively
13 Evaluate the fitness function of each search agent (vector) f(Xi)
14 End for
15 Update the vector Xa, Xb, and Xδ
16 Set t=t+1
17 Until (t≥Maxiter). {Termination criteria are satisfed}
18 Reduce the best solution Xα

3.8 The Proposed Method

Getting Your Hands on the Application The objective of the suggested approach is to achieve a real-time implementation. Time to completion is of the utmost importance in such applications.

The user requests that the action be done within a certain amount of time and specifies that time restriction. The user provides their input for the time restriction.

The CH is Chosen To begin, a central head (CH) node must be chosen. The nodes with the most neighbours are selected as CH for this purpose.

It is this node machine's job to spawn wolves, which are then dispersed at random to the VMs in the vicinity.

The OGW OGW process begins with the CH node's alpha wolf. As the alpha wolf, the pack leader, and the beta wolf, who helps make choices, begin to wander about other nodes, the pack is well on its way to the hunt. Locating a node the server is then given a list of potential nodes, each of which has been deemed suitable based on the predicted loads.

Establishing a cutoff point Each VM's load has been analysed based on a defined threshold. Overload occurs if the VM's load is greater than the threshold; underload occurs if the load is less than the threshold.

If $(L_{vmj} \leq \Psi)$

(VM) is under loaded

Else if $(Lvmj \geq \Psi)$

(VM) is over loaded

End

The load threshold of each VM can be calculated as follows:

$$\psi = AL + \sigma \quad (21)$$

Hence, if the load of each machine is more than the threshold, that VM is among the nodes with overload machines, and no work is allocated to it; AL represents the mean of VMs in a data centre, and represents the standard deviation of the mean load of the VM in a data centre. While not idle, the machine is working hard. Now, a fitness factor of load balancing is computed for each of the under-load VMs, as the least-loaded machine is chosen from among them.

$$LB = 1 - L_{vmj} \quad (22)$$

That's why it's important to maximise a VM's LB as much as possible so that it can run your desired software.

To calculate how long it will take to do the chores, after the method and the user's time constraint have been applied, the algorithm determines whether or not the time it will take the programme to complete the job falls within the user-specified time. Estimating how long it

will take to execute each programme is essential, and this can be done by dividing the app's weight (w) by the speed of the VM's processor (j). An application's execution time on a VM may be approximated using the following formula: where Wi is the application's weight and SPj is the VM's processor speed.

$$T(I,j) = \frac{w_i}{SP_j} \quad (23)$$

The Minimalist MinrrT Minimalist MinTT After an estimate of how long it will take for each virtual machine to execute the programme has been calculated, it must be verified that the total time is within the user's tolerance. The application's Quality of Service (QoS) is determined for these resources if and only if the expected time falls within the specified range. This virtual machine (VM) is now idle since it has no tasks associated with it.

If (Mintt≤T (I,j)≤Maxtt)
Reliability assessment
Else
Reliability=0
End

Methods for Assessing Quality of Service Quality of service (QoS) is the probability that a system will function as expected and without interruption for a certain period under specified circumstances. Quality of service (QoS) in software may alternatively be defined as the likelihood that all allocated resources will stay effective until all tasks are finished. Using Eq. (24), we can determine a VM's quality of service.

$$R = e^{-\lambda\left[\frac{w_i}{w_{vm}}\right]} \quad (24)$$

Where is the number of failures in a certain amount of time, Wi is the weight of the ith job, and Wvm is the weight of the tasks currently running in a virtual machine.
Identifying the weight carried by each node.

Each virtual machine's workload, or "Load," is the sum of all of the tasks that have been allocated to it.

$$L_{vmj} = \frac{\text{num}_{\text{task}}}{\text{rate} - s_{vnj}} \quad (25)$$

Where Lvmj is the VM load (j), Numtask is the number of tasks, and Rate Svmj is the service rate (in j) for the VM (VM). It must now be decided if this virtual machine is one of the overworked or underutilised ones. If the latter is true, the computer is not given any work to do. Moreover, the following formula may be used to determine the typical VM load in a data centre:

$$\text{AL} = \frac{1}{m}\sum_{i=1}^{m} L_{vmj} \quad (26)$$

There are M VMs in total.

Hence, the following formula may be used to determine the data center's VM load standard deviation: Where L is the average load of the VMs in a data centre, Lj is the load of the jth VM, and M is the total number of VMs and is the standard deviation of the load in the VMs.

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(L_j - AL)^2} \quad (27)$$

When surrounded by wolves and contemplating which node to pick: If XP (t) represents the values of the location vector for the jth virtual at time T, RJ is the QoS for the jth VM, and LBJ is the fitness factor of the jth VM's load, the suggested technique has k Wolves choose an under loaded VM with suitable QoS to give tasks to it.

$$p_i^k(t) = \begin{cases} 0 \\ \dfrac{[X_p(t)]^\alpha [R_j]^\beta [LB_j]^\gamma}{\sum [X_p(t)]^\alpha [R_k]^\beta [LB_k]^\gamma} \end{cases} \quad (28)$$

Making a strategic node choice and launching an attack: After the alpha and beta wolves have finished their evaluation and confirmation of a node, they will launch an assault on the game and choose that node as the optimal one. Figure-5 is a flowchart representation of the suggested procedure.

4. THE SIMULATION ENVIRONMENT

This new approach was simulated and tested using CloudSim. It's a free and open-source simulation toolkit for the cloud. It was developed at the CLOUDS lab of the University of Melbourne's Department of Computer and Software Engineering [30]. Data centres, virtual machines, applications, users, computational resources, and rules for managing these components are all defined and managed with the help of the CloudSim tools (e.g., scheduling). Users may assemble the pieces to test out novel cloud-based methods of operation. In addition, CloudSim may be used to assess the efficacy of tactics from a variety of perspectives, including cost-benefit analysis and the reduction of application launch times.

The use of CloudSim is boundless since classes may be added or removed and it is possible to create and implement fresh guidelines. CloudSim functions as a set of modular components from which you may construct your unique simulation environment. As a consequence, Cloudsim is not a turnkey tool where you can just adjust certain settings and then utilise the output in your project. The different CloudSim layers are shown in Figure-6. At the simulation's base are cloud resources like servers and data centres that run throughout the simulation's period. The cloud services, such as the allocation of resources like CPUs, RAM, bandwidth, etc., sit on top of this layer. VM

services and UI frameworks make up the top two levels. All the tests were run on a Lenovo workstation equipped with a Core i7 processor running at 4.2 GHz and 16 GB of main memory.

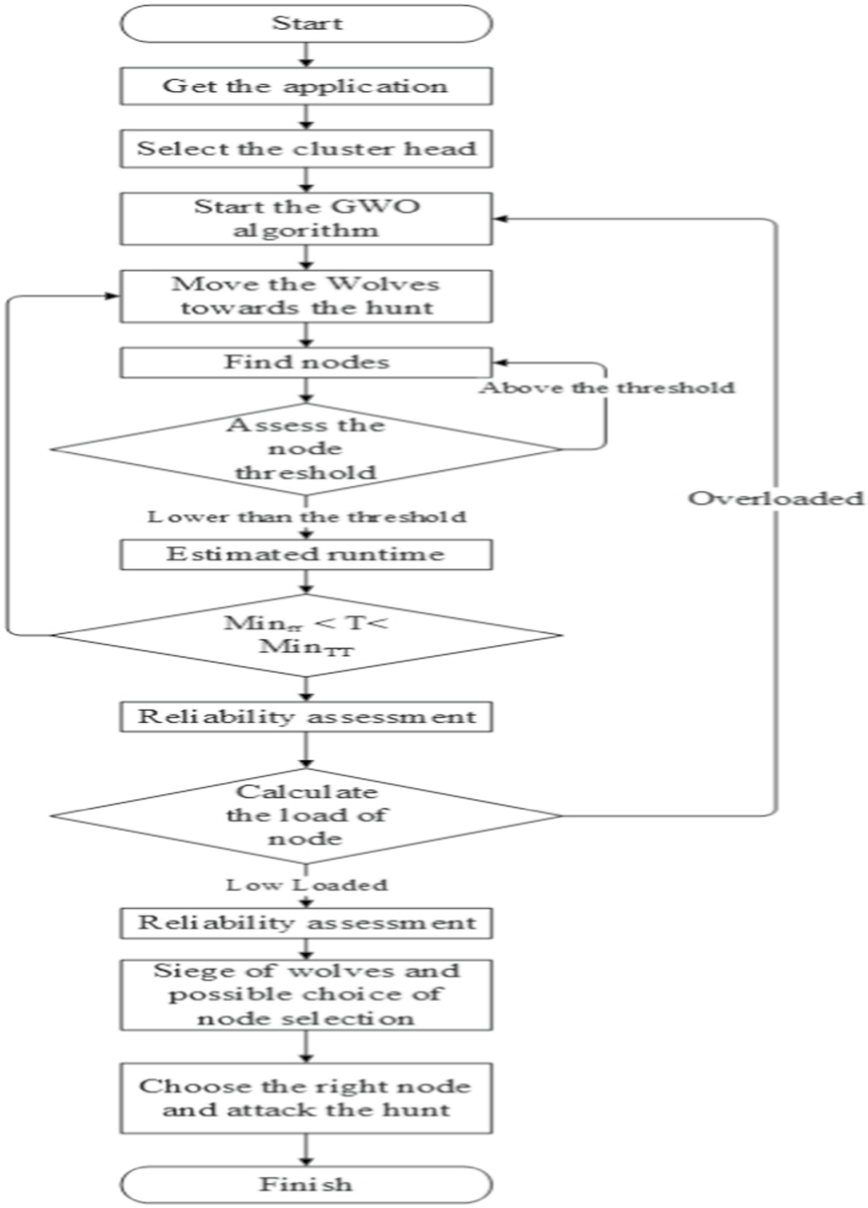

Figure-5. Flowchart of the proposed method.

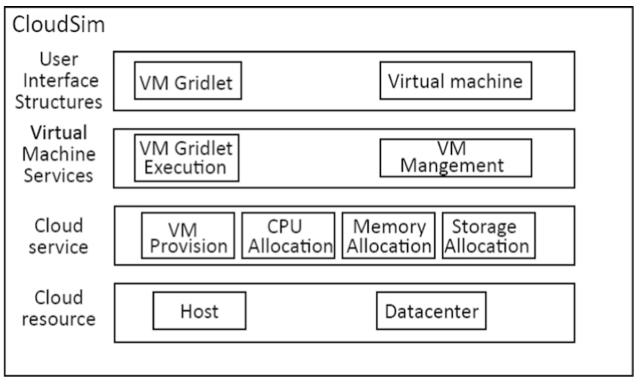

Figure-6. The architecture of CloudSim.

4.1 Simulation Data and Parameters

CloudSim was used to simulate cloud computing at the SaaS level for this investigation. This simulator allows for the modelling of a virtual setting and the facilitation of the availability of associated resources. To cut down on makespan and QoS degradation in cloud computing environments, the suggested solution aims to deliver an efficient OGW-based scheduling mechanism. As you'll see below, we put the suggested technique through its paces by conducting several experiments to test it. Experiment one defined a cloud-based data centre with three hosts. Virtualization was possible on each of these hosts, as was resource sharing across many VMs. Table-2 lists the specifics of such hosts' software, whereas Table-3 lists the simulation parameters used by OGW methods. As

the suggested OGW algorithm is a metaheuristic, it was compared to well-known metaheuristics algorithms such as the ABC [13], PSO [14], and GA [18].

Table-2. The technical characteristics of the host.

Host ID	Processing cores	Processing speed	Mips RAM	MB Hard	MB Bandwidth Mbps
1	4	5000	204,800	1,048,576	102,400
2	2	2500	102,400	1,048,576	102,400
3	1	1000	51,200	1,048,576	102,400

Table-3. The parameters of the algorithm.

Parameters	Value
Population size (no. of solutions)	80
Maximum iterations	100
C1, C2	1.49445
R1, R2	Random numbers between 0 and 1

4.2 The Obtained Results

The makespan results showed that the suggested approach is much more efficient than the alternatives. The OGW method is preferable because it completes processing in less time than competing algorithms and finds appropriate nodes for load transfer more quickly. The suggested approach is a QoS/OGW hybrid algorithm. The OGW algorithm for load balancing was developed with inspiration from the foraging strategies of Grey wolves in the wild. In the present investigation, the alpha wolf first searches the loads that are unrelated to any service or job and then commands the other wolves to attack the unrelated loads. The ABC algorithm also outperformed the others since it can locate the most appropriate loads throughout the search process. Initial choices made by the round-robin (RR) scheduling are crucial to the performance of the honeybee-mating method. The figure clearly shows that the ABC balancing approach outperforms the refined GA algorithm. That's because, unlike the GA algorithm, the ABC method's load balancing can identify both overloaded and underutilised virtual machines. Figure-7 provides a clearer example of this.

The time taken to respond by each of the three algorithms studied is shown in the figure below. Twenty, thirty, forty, fifty, sixty, seventy, and eighty jobs have been designated. One of these criteria is reaction time, and to find out how long it takes to solve aggregate functions as a whole, we add up the times it takes for each service. Taking into account the reaction time parameter, the preceding graphic shows that the suggested method performs well. The reaction time diagram in the bee algorithm begins with the worker bee, who initiates the search for nectar and then alerts the other bees to its location through a waggle dance.

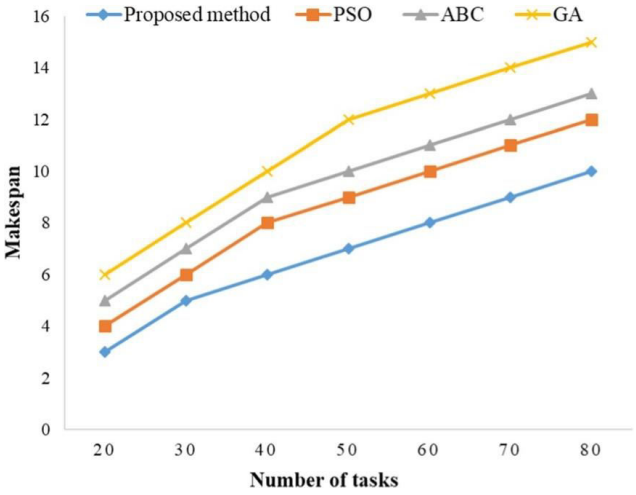

Figure-7. The comparison of makespan.

The suggested approach is quite close to the artificial bee algorithm. Figure-8 provides a clearer example of this.

The suggested approach demonstrates how resource use may improve response time and customer satisfaction. In addition, the strategy may result in the fair allocation of workload across VM, which boosts the use of resources at once and shortens the total processing time. Also, the OGW algorithm is shown to provide superior outcomes when compared to competing algorithms and to have faster access to available resources. Clearly shown in Figure-9 are the outcomes of this strategy.

According to Figure-10, the OGW algorithm has achieved ideal performance in terms of cost reduction, and it has been able to lower costs to a greater degree than the other algorithms while producing simulation results that are almost identical to those of the other three algorithms. The cost evaluation index quantifies the amount of money spent on each virtual machine (VM) by factoring in the amount of memory, number of processes, size of the virtual machine, and the amount of bandwidth consumed. In this scenario, the workload is nearly identical across all algorithms; the algorithms' conditions, number, and hardware features are also comparable; and the algorithms' running times are comparable; therefore, the algorithm that uses VM less is deemed good because it can complete its tasks with the fewest resources.

According to the gathered data, the suggested method is less flexible than GA, PSO, and ABC. 100

iterations of the GA, PSO, and ABC algorithms were used to evaluate the proposed algorithm's convergence. Convergence was tested over 80 tasks using 200 service candidates per job over 100 iterations, as shown in Figure-11.

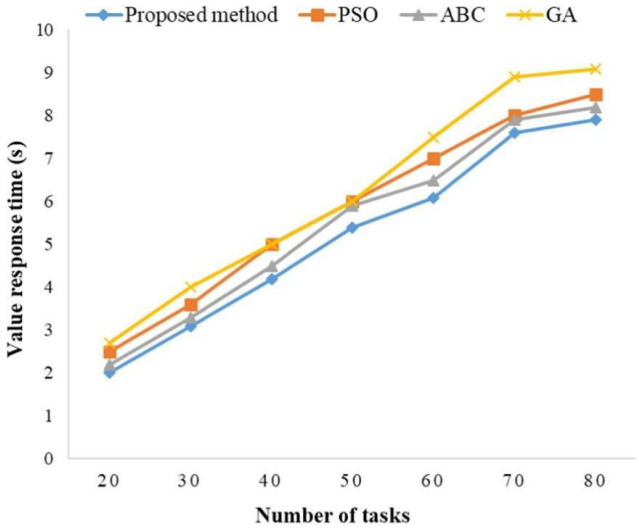

Figure-8. The comparative diagram of response time.

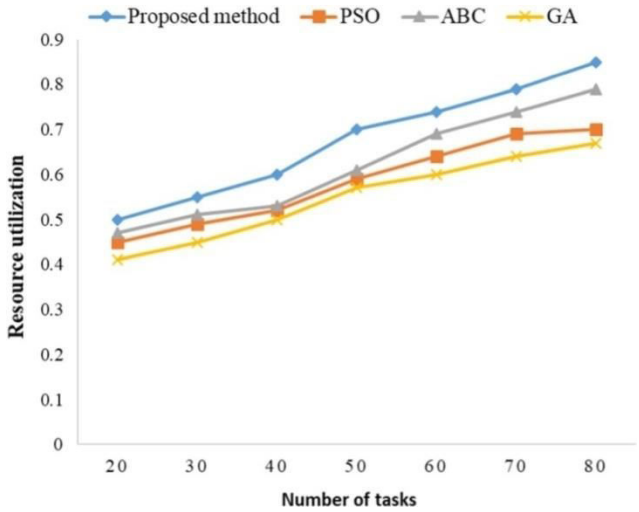

Figure-9. The diagram of resource utilization.

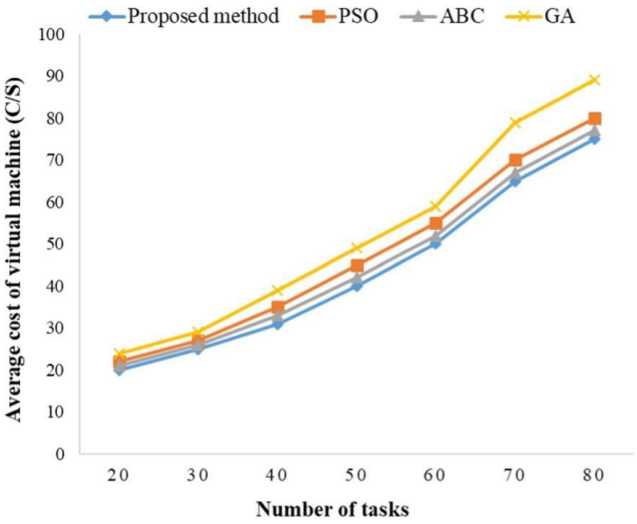

Figure-10. The average costs of running a VM are according to the number of tasks.

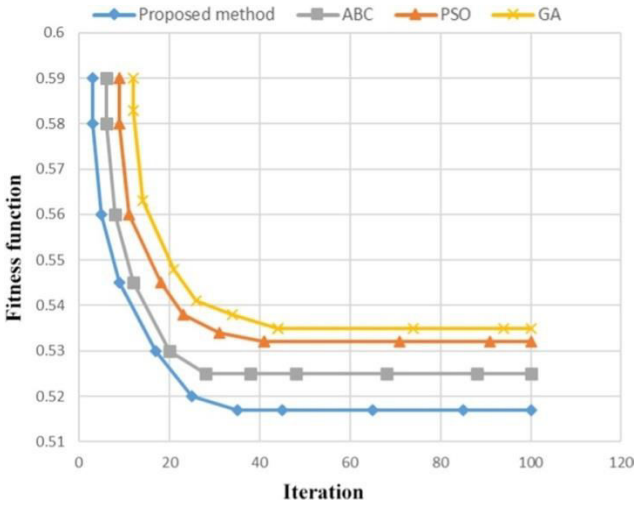

Figure-11. The convergence diagram for 80 tasks in 100 iterations.

5. ANALYSIS OF RESULTS

There are a variety of load-balancing techniques that have been suggested for use in hybrid cloud, grid, and distributed computing settings. Of course, there are benefits and drawbacks to everything. The present research makes use of an OGW algorithm based on QoS to balance loads. The purpose of the research was to find the optimal load distribution among the available resources, ensuring that no resources were overloaded or underloaded. For this reason, the QoS and the load of each resource were calculated, \ and sand load balancing was maintained using the OGW method. The application was transferred to a server with a better quality of service and fewer requests. The efficiency and effectiveness of the cloud system were improved thanks to the novel approach used in the present research to maintain load balancing based on the QoS. The simulation results also showed that the suggested technique is more stable than the competing methods, with less time spent on tasks and less imbalance. The suggested technique was studied and compared to

other approaches; the findings confirmed that it is the optimal algorithm. The OGW method outperformed the ABC, GA, and PSO algorithms across all metrics thanks to its weighted search and careful consideration of each node's contribution. In contrast to previous research, this one prioritises picking a virtual machine (VM) that isn't overworked and one that is most amenable to users' needs. The suggested solution improved the VMs' workload distribution by adjusting the load balancing algorithm. Because of this, more resources were put to use than before. The use of the heuristic algorithms and the suggested strategy resulted in time and cost savings compared to other approaches (see Figure-12).

Makespan was also studied, and it was discovered that the suggested approach is much better than the alternatives.

That is to say, the alpha wolves go for the machines that are less busy first. After deciding where to send the cargo, they give the CH the go-ahead to do so. Figure-13 shows the average outcomes.

Response time was also studied, and it was shown that the algorithm had the potential to provide better outcomes than other approaches. While the ABC algorithm produced a faster and more competitive solution than the one suggested in this research, the OGW algorithm outperformed the others in 80 challenges. The typical outcomes are shown in Figure-14.

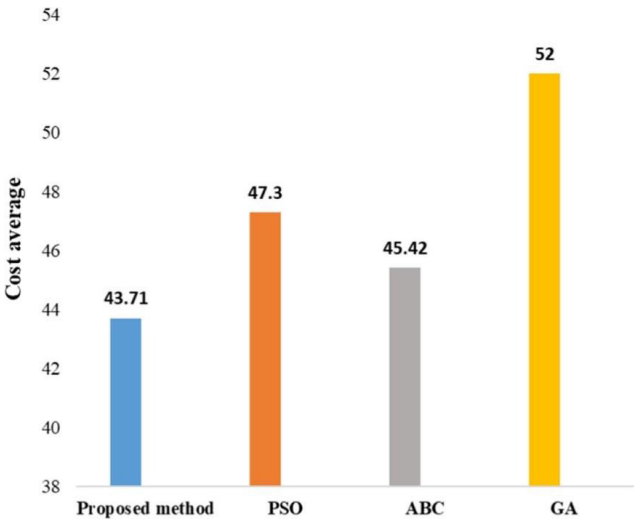

Figure-12. The average costs for different 80 tasks.

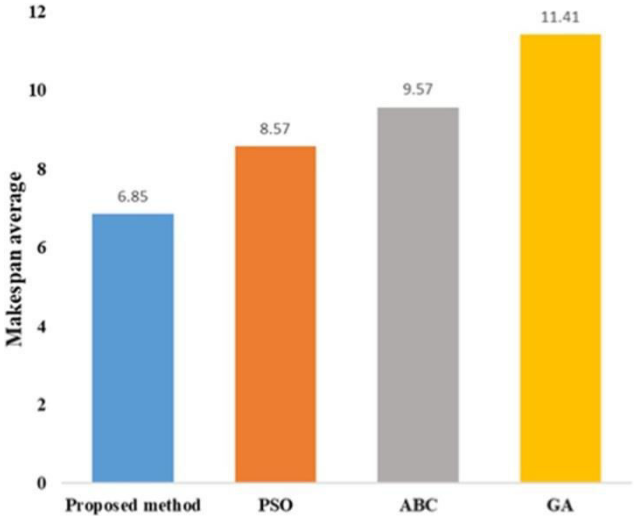

Figure-13. The average results of makespan in 80 tasks.

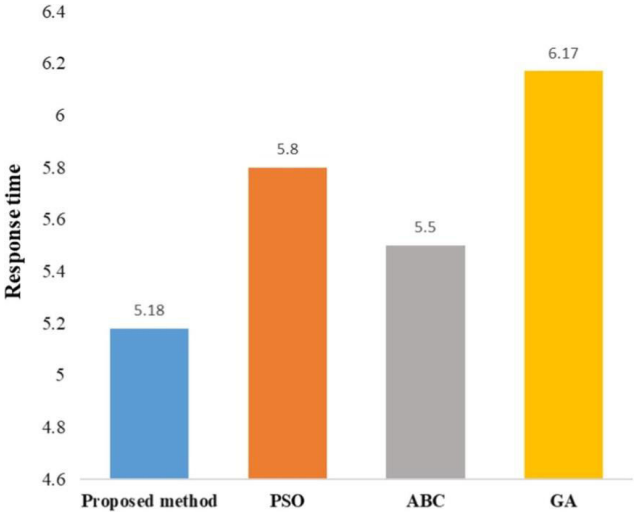

Figure-14. The average results of response time in 80 tasks.

Table-4. Results for power consumption and throughput using no of tasks.

No .of Tasks	80	90	100
No. of Iterations	100	110	120
Power Consumption	0.000527	0.000117	0.000049
Throughput	0.063777	0.046179	0.09515
Memory Utilization	0.000586	0.002352	0.001166
Server load	1.31	1	2.32
Turn Around time	0.422224	0.181626	2.367719

6. CONCLUSIONS

The capacity to come up with a wide variety of novel and effective approaches to resolving problems is what we mean when we talk about creativity and innovation. In this article, we explored the idea of load balancing as a means of discovering nodes whose service values are below the threshold at which they begin searching for prey in their surroundings. Before sending his pack after them, the leader wolf checks out the services that aren't essential to completing any particular mission. As compared to other approaches, the outcomes produced by this technique were superior. The suggested approach effectively decreases task response times while increasing makespan usage during load balancing. Consider the following for future research that aims to build upon and refine the present study's findings. Thus, future research should focus on providing a metaheuristic that makes use of machine learning and the MCDM approach. The suggested solution might also benefit from the deployment of data centres and virtual machines. Additionally, they might encompass more than just dependability, including issues like usability, safety, and scalability. In the future, dependent activities will do load balancing dynamically.

REFERENCES

[1] Haji L. M., Ahmad O. M., Zeebaree S. R., Dino H. I., Zebari R. R., Shukur H. M. 2020. Impact of cloud computing and the internet of things on the future internet. Technol Rep Kansai Univ. 62(5): 2179-2190.

[2] Kumar J., Rani A., Dhurandher S. K. 2020. Convergence of user and service provider perspectives in mobile cloud computing environment: taxonomy and challenges. Int J Commun Syst. 33(18): e4636.

[3] Goldberg D. W., Bowlick F. J., Stine P. E. 2021. Virtualization in Cyber GIS instruction: lessons learned constructing a private cloud to support development and delivery of a WebGIS course. J Geogr High Educ. 45(1): 128-154.

[4] Sefati S., Abdi M., Ghafari A. 2021. Cluster-based data transmission scheme in wireless sensor networks using black hole and ant colony algorithms. Int J Commun Syst. https://doi.org/10.1002/dac. 4768

[5] Eswari S., Manikandan S. Competent data transmission function in cloud computing with high probability aesthetic.

[6] Hayyolalam V., Pourghebleh B., Kazem A. A. P., Ghafari A. 2019. Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques. Int. J Adv Manuf Technol. 105(1): 471-498.

[7] Golchi M. M., Saraeian S., Heydari M. 2019. A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: performance evaluation. Comput Netw. 162: 106860.

[8] Alicherry M., Lakshman T. 2013. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In: 2013 Proceedings IEEE INFOCOM, 2013. IEEE. pp. 647-655.

[9] Nurmi D. *et al*. 2009. The eucalyptus open-source cloud-computing system. In: 2009 9th IEEE/ ACM International Symposium on Cluster Computing and the Grid, 2009. IEEE. pp. 124-131.

[10] Zanbouri K., Jafari Navimipour N. 2020. A cloud service composition method using a trust-based clustering algorithm and honeybee mating optimization algorithm. Int J Commun Syst. 33(5): e4259.

[11] Sefati S., Navimipour N. J. 2021. A QoS-aware service composition mechanism in the Internet of Things using a hidden Markov model-based optimization algorithm. IEEE Internet Things J. https:// doi.org/10.1109/JIOT.2021.3074499

[12] Tikhamarine Y., Souag-Gamane D., Ahmed A. N., Kisi O., El-Shafe A. 2020. Improving artifcial intelligence model accuracy for monthly streamflow forecasting using Optimization of Gray Wolves (OGW) algorithm. J Hydrol. 582: 124435.

[13] Kruekaew B., Kimpan W. 2020. Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. Int. J Comput Intell. Syst. 13(1): 496-510.

[14] Devaraj A. F. S., Elhoseny M., Dhanasekaran S., Lydia E. L., Shankar K. 2020. Hybridization of Firefly and Improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. J Parallel Distrib Comput. 142: 36-45.

[15] Lilhore U. K., Simaiya S., Maheshwari S., Manhar A., Kumar S., Cloud performance evaluation: hybrid load balancing model based on modified particle swarm optimization and improved metaheuristic firefly algorithms.

[16] Kokilavani T., Amalarethinam D. G. 2011. Load balanced min-min algorithm for static meta-task scheduling in grid computing. Int J Comput Appl. 20(2): 43-49.

[17] Goyal S. K., Singh M. 2012. Adaptive and dynamic load balancing in the grid using ant colony optimization. Int. J. Eng. Technol. 4(4): 167-174.

[18] Makasarwala H. A., Hazari P. 2016. Using genetic algorithm for load balancing in cloud computing. In: 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2016. IEEE. pp. 1-6.

[19] Garg S. K., Yeo C. S., Anandasivam A., Buyya R. 2011. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. J Parallel Distrib Comput. 71(6): 732-749.

[20] Buyya R., Yeo C. S., Venugopal S., Broberg J., Brandic I. 2009. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Futur Gener Comput Syst. 25(6): 599-616.

[21] Kashyap D., Viradiya J. 2014. A survey of various load balancing algorithms in cloud computing. Int. J. Sci. Technol. Res. 3(11): 11-119.

[22] Smimite O., Afdel K. 2020. Containers placement and migration on a cloud system. arXiv: 2007. 08695.

[23] Hao F., Lakshman T., Mukherjee S., Song H. 2009. Enhancing dynamic cloud-based services using network virtualization. In: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures. pp. 37-44.

[24] Dillon T., Wu C., Chang E. 2010. Cloud computing: issues and challenges. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, 2010. IEEE. pp. 27-33.

[25] Bari M. F., Zhani M. F., Zhang Q., Ahmed R., Boutaba R. 2014. CQNCR: optimal VM migration planning in cloud data centers. In: 2014 IFIP Networking Conference, 2014. IEEE. pp. 1-9.

[26] Ashouraei M., Khezr S. N., Benlamri R., Navimipour N. J. 2018. A new SLA-aware load balancing method in the cloud using an improved parallel task scheduling algorithm. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), 2018. IEEE. pp. 71-76.

[27] Ghobaei-Arani M., Rahmanian A. A., Souri A., Rahmani A. M. 2018. A moth-fame optimization algorithm for web service composition in cloud computing: simulation and verification. Softw Pract Exp. 48(10): 1865-1892.

[28] Mirjalili S., Mirjalili S. M., Lewis A. 2014. Grey wolf optimizer. Adv Eng. Softw. 69: 46-61.

[29] Betka A., Terki N., Toumi A., Dahmani H. 2020. Grey wolf optimizer-based learning automata for solving block matching problems. SIViP. 14(2): 285-293.

[30] Mishra S. K., Sahoo B., Parida P. P. 2020. Load balancing in cloud computing: a big picture. J King Saud Univ. Comput. Inf. Sci. 32(2): 149-158.

[31] Siddiqi M. H., Alruwaili M., Ali A., Haider S. F., Ali F., Iqbal M. 2020. Dynamic priority-based efficient resource allocation and computing framework for vehicular multimedia cloud computing. IEEE Access. 8: 81080-81089.