International Journal of Cloud Applications and Computing
Volume 13 • Issue 1

A Hybrid Binary Bird Swarm Optimization (BSO) and Dragonfly Algorithm (DA) for VM Allocation and Load Balancing in Cloud

Thanwamas Kassanuk, Pibulsongkram Rajabhat University, Thailand*

Khongdet Phasinam, Pibulsongkram Rajabhat University, Thailand

ABSTRACT

The cloud platform is becoming one of the fastest-rising environments in human activities, connecting the whole world in the upcoming decades. The three crucial aspects of cloud computing that enhance the quality of service are load balancing, task scheduling, and resource allocation. To address these issues, the research proposed dynamic degree balance with CPU_based VM allocation policy integrated with hybrid bird swarm optimization (BSO) and dragonfly algorithm (DA). The proposed algorithm focuses on improving the overall performance of the system by limiting DoI, execution time, and response time, while also maintaining system balance. In the CloudSim tool, D2B_CPU based BSO-DA is implemented and evaluated. The simulation results, on the other hand, show that the proposed BSO and DA-based load balancing scheme is significantly more effective in balancing load optimally among virtual machines more quickly than existing algorithms. The proposed method's efficiency is evaluated by comparing it to other existing techniques.

KEYWORDS

Bird Swarm Optimization, cloud computing, Dragon Fly Algorithm, load balancing, optimization, scheduling, virtual machine

INTRODUCTION

Due to advancements in communication technology and internet usage, as well as their ability to solve complex problems, cloud computing is emerging as a networking technology. Using the internet, cloud users have access to hardware and software resources. An Internet-based computing model called cloud computing allows resources such as software, information, services, storage, and servers to be shared with multiple users (Xu et al., 2018; Kumar et al., 2020). Because of its services to customers, Cloud Computing (CC) is an established business model for distributed computing. The CC model provides for the sharing, allocation, and access of IT resources based on individual needs. CC also offers many services, including Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS) and Software-as-a-Service (SaaS) (Xue et al., 2018.). They are useful in different domains, including industrial, business, scientific, etc. Amazon, Oracle, HP, IBM, and Apple use cloud computing techniques (Rawat et al., 2020).

DOI: 10.4018/IJCAC.318698

*Corresponding Author

A CC platform, in general, suffers from three major problems: Load Balancing, Distributed Framework and Virtualization. In Meeting the demands of cloud users and providers, balancing of load and scheduling of tasks are two major problems in the management of cloud resources (Kumar et al., 2018; Ruan et al., 2019). A load balancer allocates tasks to various machines in such a way that job execution times are reduced and virtual machine performance is monitored (Daming et al., 2020; Polepally et al.,2019). Load balancing's ultimate objective is to lower the highest execution time (maximum Makespan time) whereas raising cloud resource utilization (Kumar & Sharma, 2018). Scheduled tasks can be assigned virtualized resources for a specific period. It canbe done with the help of a cloud service broker and a task scheduling algorithm (Xingjun et al., 2020). Scheduling designates that what tasks will take the lowest time to accomplish. The workload is rising due to the growing set of users in the cloud, but the set of virtual machines (VMs) is remaining the same. Due to the consumption of energy constraints, the set of Virtual machines is reduced by the capacity of PMs. Load balancing and Scheduling tasks ensure that no node is over-or under loaded by the workload (Pradhan et al., 2020; Ragmani et al., 2020).

Static and dynamic load balancing approaches are the two major categories of load balancing approaches (Kruekaew et al., 2020). genetic algorithm (GA), artificial bee colony (ABC), ant colony optimization (ACO) and other optimization algorithms are often used in an existing Complex Load balancing research project. Researchers develop many heuristic and meta-heuristic methodologies such as the jaya algorithm, dragonfly optimization algorithm, and bee colony optimization algorithm and to achieve greater load balancing and task scheduling performance (Priya et al., 2019; Gupta et al., 2021; Milan et al., 2019).

Since metaheuristics seek a wider search area than heuristics, they have a greater computational cost, and they are using a directed general check to solve the scheduling issue. To achieve the optimal solution in the shortest period, heuristics reduce the search space for metaheuristics (Raja et al., 2020; Janakiraman et al., 2021). However, balancing of the load is a multi-objective problem: the main goal is to schedule tasks or jobs evenly within available resources in order to minimize the relative imbalance, and a secondary goal is to make the best use of available resources as well as reduce makespan and response time (Fatima et al., 2019; Jena et al., 2020).

Considering these facts, proposed an algorithm that balances the load using dynamic degree balanced with CPU-based VM allocation technique. In addition, a new fast meta-heuristic algorithm is proposed to address the aforementioned problems; bird swarm optimization (BSO) and Dragonfly (DA) algorithm is proposed for allocation of tasks and balancing the load in a cloud environment. A recent algorithm is compared with the proposed algorithm in order to determine how efficient it is. As per the results, the proposed algorithm effectively manages load imbalance. The key contribution of this research is as follows:

- To implement the dynamic degree balanced with CPU-based (D2B_CPU based) VM allocation.
- To implement the hybrid binary bird swarm optimization (BSO) and dragonfly (DA) algorithm.
- Virtual machine tasks can be allocated by identifying the overloads and under loads of VMs by the binary bird swarm optimization (BSO) and its response time can be accelerated by adopting the Dragonfly (DA) algorithm.
- Using CloudSim to build a cloud platform.
- Algorithm reduces the load balancing aspects are as follows: Makespan, Execution time, degree of imbalance, response time and load balancing with an increase in throughput in the cloud applications.

The upcoming sections are planned as follows: Section 2 discusses about the Literature review of numerous LB techniques. The problem statement in allocating tasks and balancing the load are discussed in section 3. Section 4 describes the proposed D2B CPU based VM allocation algorithm with hybrid BSO-DA. The results of this algorithm are presented in sections 5 and 6. Section 7 concludes with the conclusion.

LITERATURE REVIEW

Numerous assignment models, task scheduling, load balancing, and resource allocation methods have been proposed in recent years to minimize the execution and makespan time, optimize the utilization of resources, as well as distribute the load across machines. These works are discussed in this section.

(Ullah et al., 2020) focused on enhancing the job distribution system in VM for cloud computing employing load balancing. As a result, the Bat algorithm fitness function value utilized in the load balancer section was modified. When algorithm iterations are completed, it is time to allocate the task among various VM; consequently, the algorithm was adjusted. The second alteration occurred during Bat's dimension section search procedure. The suggested approach is referred to as the modified Bat algorithm.

(Negi et al., 2021) introduce a hybrid of supervised clustering ML, ANN and IT2FIS based LB algorithm, dubbed CMODLB, for balancing the load. Initial implementation of ANN-LB technique clusters VMs into underloaded and overloaded VMs using BOEK algorithm. Next, user tasks are scheduled for underloaded VMs to optimize resource utilization and load balancing. Using several cloud criteria, TOPSIS-PSO algorithm facilitates job scheduling. The VM manager migrates VMs to balance PM load. If a PM is overcrowded and another PM is underloaded, VM migration decisions are made based on the appropriate criteria. Multiple significant characteristics inform the judgments of IT2FS that facilitates virtual machine (VM) migration.

In a cloud environment, (Kumar et al., 2019) presented by hybrid CS-FA for LB. At first, each virtual machine's load and capacity were determined. LB was used to perform task when the virtual machine load exceeded the balanced threshold level. Jobs are alloted to the best Virtual Machines (VMs) by CS-FA and migrates overloaded VM tasks to under loaded VM tasks. In a cloud environment, this algorithm greatly avoids the problem of uneven workload performance. An evaluation of the suggested CS-FA method is compared with existing LB methods such as Honey Bee Behavior LB (HBB-LB), Hybrid Dynamic LB (HDLB) and Dynamic LB (DLB) for the purposes of evaluating their load and capacity.

(Ouhame et al., 2018) suggested a hybrid algorithm to improve VM allocation using GWO and ABC. In VM for cloud computing, execution time, average network, throughput network stability and energy consumption were improved using the presented technique. According to those results, the proposed algorithm improves 1.25% accuracy compared to RAA algorithm, GWO algorithm and ABC algorithm and it is more efficient in VM resource allocation for cloud computing.

(Kaur et al., 2020) have proposed and implemented a method for executing workflows in a cloud infrastructure, for instance, DLD-PLB. Taking the Genome workflow tasks into account, a suggested framework has been developed. In terms of makespan and time, our proposed load balancing optimisation model was compared to an earlier suggested scheme. Earlier load balancing approaches used ACO to optimize underutilized VMs, and a hybrid PEFT-BAT approach to accelerate overflow Virtual machines.

Task scheduling was optimised using a metaheuristic algorithm, and load balancing was implemented by (Ziyath et al., 2018)). Two algorithms were used, MHO-D and MHO-S where MHO-S was used to calculate the confirmed and static properties of virtual machines. MHO-D is in charge of scheduling the tasks for which property values cannot be confirmed and discloses a least property they will retain. The client also specifies a maximum acceptable value for the task completion. MHO-D schedules the tasks optimally based on these values. Based on a comparison with a few existing solutions, the proposed MHO algorithm was found to operate with a minimum number of virtual machines at low execution times while utilizing the VM clusters of the infrastructure providers.

Using hybridization of MPSO and an improved Q-learning algorithm called QMPSO, (Phi et al., 2018) presented a new method of dynamic balancing of workload among virtual machines. Using the gbest and pbest generated by improved Q-learning, the MPSO's velocity was adjusted through the hybridization process. With hybridization, workloads are balanced among the VMs, throughput is maximized, and priorities are maintained by minimizing waiting times. A comparison of the QMPSO simulation results to the existing scheduling algorithm and balancing of load has demonstrated the robustness of the technique.

The FIMPSO algorithm was presented by (Devarajet et al., 2018) for an energy-efficient LB method in cloud. In this method, the benefits of both IMPSO and FF are incorporated. By presenting the FIMPSO algorithm, the average load for making and response time of the tasks was improved. The following metrics were used to evaluate the experimentation results: throughput, makespan, reliability, resource utilization and execution time. Simulation results indicated that the proposed FIMPSO model performed better than the compared methods.

(Arora et al., 2019) presented a hybrid optimization technique, EHGWO, in the CC paradigm for LB by identifying the appropriate VMs for executing the reassigned tasks. The technique reduces workloads from overburdened VMs to maintain performance of the system. Here, the PM, capacity, and load of VM are computed to determine whether or not the LB must be performed. Additionally, TPF and VPF, are considered when reallocating jobs from an overloaded VM to an underutilized VM. The suggested method assigns VM tasks based on newly developed fitness functions.

(Sahana et al., 2029) have described a technique for load balancing using the weighted Round-Robin algorithm that can execute client requests on several servers with little response time. Due to these factors, a cloud-based dynamic load balancer is utilized to handle the problem of load balancing in cloud architecture.

In their study, (Kaur et al., 2020) suggested a framework for resource scheduling and load balancing that maximises VM usage while guaranteeing uniform load distribution. To achieve its optimal performance regarding makespan and cost, the suggested framework combined heuristic techniques with metaheuristic algorithms. PEFT-ACO and HEFT-ACO are two hybrid methods used to implement the suggested framework. For both approaches, the simulated results of cost metrics and makespan has been analysed and compared.

PROBLEM STATEMENT

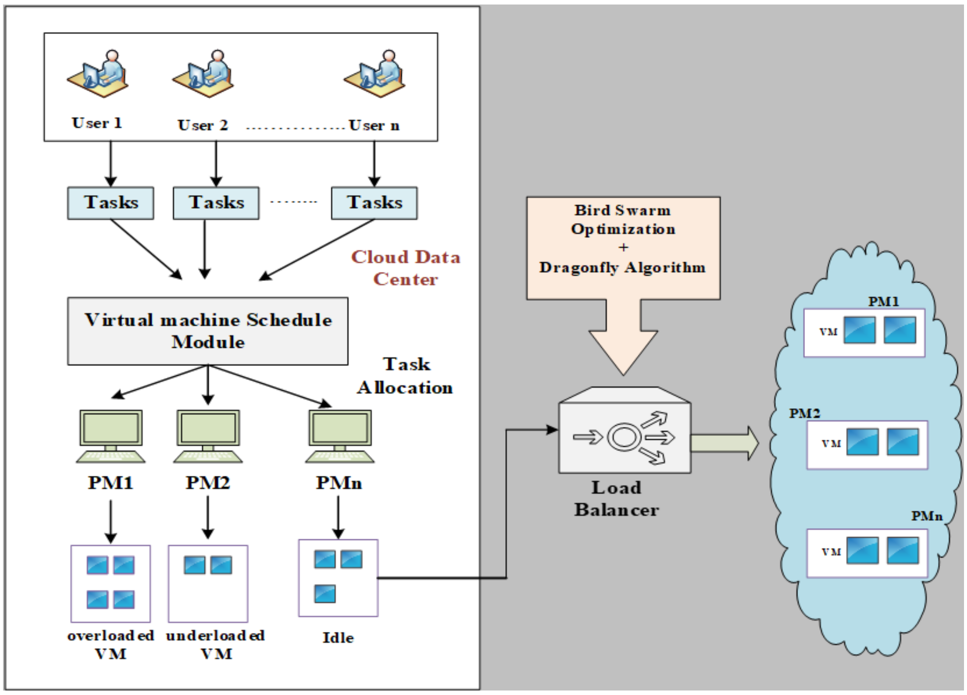
A key component of cloud technology is load-balancing. The load balancing mechanism makes the most of available resources while also managing complex load imbalances. A proper load balance also reduces resource consumption, thereby lowering energy consumption. Numerous approaches for balancing the load in cloud systems have been developed by various scholars. Much of this research, however, suffered from a number of issues. Therefore, the present study focused on virtualization.

The overuse of virtual machines in the early stages and the underuse of them in the late stages has already been established. Resources are sometimes heavily loaded while other resources are idle due to the CPU's random utilization. Load imbalance makes cloud systems ineffective, reducing scalability, reliability, availability and throughput while also increasing migration and response times. An effective load balancing mechanism facilitates balanced resource utilization, improving maximizes the scalability, reliability, availability and throughput and reducing the migration and response time. It is assumed that the proposed algorithm can effectively minimize the degree of imbalance, makespan, response time, load balancing and execution time and maximize throughput.

METHODOLOGY

To maintain a trade-off, load balancing (LB) distributes less power across servers with an equal load. In this research, proposed dynamic degree balanced with CPU (D2B_CPU based) VM allocation. In each data center, users can access some computing resources. Multicloudusers have a variety of tasks, each of which is allotted to the various Virtual machine. The load of the VM can be determined by measuring the processing time of every task. As a result, every task is analyzed at varying rates, resulting in VM load fluctuations. Suppose the overloading VM is shared with the under loading VM in order to maximize resource utilization. Hence, proposed is the hybrid binary BSO and DA load balancing algorithm that improves the metrics such as duration of execution and response time spontaneously. The allocation of tasks by finding the overloaded with the under loaded condition of VMs can be boosted by BSO and its response time can be improved by DA-based strategies.

Figure 1.
Proposed VM allocation and task allocation framework



System Model

The proposed VM allocation and task allocation architecture is shown in Figure 1. In the proposed methodology, tasks are assigned to VMs according to their capacity (load). Load balancing algorithms help to distribute tasks among overloaded VMs and underloadedVMs. As part of the proposed system, there are many data centers, also known as physical machines (PM) that are equipped with virtual machines (VMs) that performs the task of the users. Each cloud user has a different number of tasks to complete on the VM. The proposed work involves two stages. In the first phase, virtual machines are alloted to the host and in phase 2, the load is distributed among the VMs.

In every data center, users can access some computing resources. Users across multiple clouds complete a variety of tasks, each of which is assigned to a different virtual machine (VM). VM load can be calculated by calculating the processing time for every tasks. As a result, each task is processed at a different rate, resulting in VM load variations. If the VM is overloaded, its workload is shared with the under loaded one in order to maximize resource utilization. *C* represents the cloud system, *P* represents the number of PMs, which includes multiple VMs, and it is denoted by *V*. *n* represents the number of PMs in Eq. (1).

$$C = \{P_1, P_2, \ldots\ldots P_k \ldots\ldots \mathrm{P}_n\} \quad 1 < k \leq n \tag{1}$$

P_k is the k^{th} number of PMs, whereas P_n is the n^{th} number of PMs. PMs are made up of several virtual machines, and this is numerically shown in Eq. (2).

$$V = \{V_1, V_2, \ldots\ldots V_i \ldots\ldots \mathrm{V}_m\} \quad 1 < i \leq m \tag{2}$$

*m*represents total number of VMs in the k^{th} PM. Moreover, cloud system includes multiple users so, *l*No. T_s are numerically displayed in the Eq. (3),

$$T = \{T_1, T_2, ...T_j.......\mathrm{T}_l\} \tag{3}$$

T represents a set of tasks, whereas *l* represents the number of tasks T_s in total. A VM is assigned to each task. The cloud system handles tasks without any problems as long as VM workloads are normal. Whenever VM workload status exceeds capacity, an LB strategy must be applied to move the task from overloaded VMs to underloaded VMs. The VMs are based on a few parameters, as shown in Eq. (4),

$$V_i = \{r_i, s_i, b_i, \mathrm{g}_i, m_i\} \tag{4}$$

Memory usage is defined, task migration cost is defined as, the variable is defined as bandwidth, the number of MIPS is defined as and numerous processors are defined as. Tasks have different execution times as well as priority values. Also, tasks are assigned to the VMs according to two main points such as (i) higher priority tasks (ii) task with low execution time are first assigned to the VM. Consequently, communication costs are significantly reduced.

Proposed Approach for VM Allocation and Task Allocation for Load Balancing

Phase 1: Dynamic Degree Balanced with CPU (D2B_CPU based) VM allocation

As an input to the algorithm, a set of VM's is provided. Following that, the algorithm creates data structures for VMTable, UsedPes, FreePes, and Host Allocation table. VMTable is a table that holds data about a virtual machine and its assigned host. UsedPes keeps track of the no. of processors used for each virtual machine. For every host, FreePes keeps track of the no. of free processors and the data about the server is stored in the Host allocation table. The next step is to determine whether or not certain Virtual machines are assigned to the Server. If a host does not have a VM assigned to it, look for hosts that use fewer processors (Pes), then assign a VM to it. Afterward, check if VMs were created successfully on the host. Update the Mabel, Usurpers, FreePes, and Host Allocation tables once all VMs have been allocated. Upon failing to create VM, the algorithm sets a minimum integer value in the Free PesTmp table. Upon completion of the allocation process, the VM is created. After this phase of VM allocation, phase 2 follows, i.e) task allocation.

Phase 2: A Novel Binary BSO and DA for Task Allocation

The proposed work aims to allocate tasks using hybridization of bird swarm optimization and dragonfly algorithm for optimal resource utilization within a short response time of the network. In a cloud environment, a swarm of birds resonates as particles. Distributing tasks among VMs is same as that of birds finding for food. Food sources that are either empty or already explored behave like a VM that is overloaded. Therefore, it is necessary to find a new food source that matches the available resources and to find an under loaded VM to migrate the tasks. The fitness function is used to evaluate particles assigned to a given issue based on the best position. Iterations result in a new best position being determined. Several particles in the cloud have their own fitness values: according to the best fitness value, assigning tasks to VMs.
Step 1: Encoding of solution

Encoding solutions is the key step in scheduling of task. A challenging aspect of task scheduling is allocating a task to a virtual machine. Every solution is composed of a number of tasks and several VMs. Every VM has its own configuration. We can schedule the tasks according to the virtual machine

capacity so that the penalty, resource usage, energy consumption and cost are minimized, while the credit is increased. Here, the initial solution is generated randomly. As an example, consider three tasks *(T_1, T_2, T_3)* containing three subtasks each *(T_1, T_2, T_3)*, totalling nine subtasks. Two physical machines *(PM_1, PM_2)* so that, PM1 is allotted by two virtual machines *(VM_1, VM_2)* and *PM_2* is allotted by three virtual machines *(VM_3, VM_4, VM_5)*. A basic solution format is shown in Table 2. Tasks t_1 and t_5 are allocated to VM_1, VM_2 is allocated by task t_2 and t_9 are, VM_3 is allocated by task t_4 and t_8, task t_3 and t_6 are allocated to VM_4 and VM_5 is allocated by task t_7. This research aims to schedule virtual machines in the most optimal way.
Fitness Function: Fitness values are evaluated by the proposed load balancing algorithm. Each problem is evaluated differently. Our aim is to optimize resource utilization while reducing task makespan and balancing load between VMs. Therefore, consider the aforementioned objective as one objective. A particle with a smaller fitness value will have the best position. Therefore, the fitness function is displayed in equation 5.

$$f_{val} = \frac{1}{Makespan} \times AverageUtilization_{VM} \tag{5}$$

Load balancing involves mapping task set T on the VM set V $(f_{val} : T \rightarrow V)$ in a cloud in which the following goals should be met: (1) the total time span must be as short as possible; (2) resource utilization should be maximized by optimizing resources, and (3) load should be evenly distributed among VMs. These goals are developed using the aforementioned QoS parameters.

Proposed Hybrid Binary Bird Swarm Optimization (BSO) – Dragonfly (DA) Approach

The hybrid LB method is explained briefly in this section. The proposed multi-objective load balancing based scheduling of tasks using hybrid binary Bird Swarm Optimization (BSO) and Dragonfly (DA) that results in low response time, DOI and execution time while balancing the load. Initialize the population along with the set of data centers, while optimizing the set of tasks based on BSO to prevent falling into local optimum. The Bird Swarm Optimization (BSO) and Dragonfly algorithms are employed to optimize the system. The Small Position Value (SPV) rule is used to convert continuous solution to discrete solutions. As the basic BSO is a continuous method for optimization, it will be unable to produce binary solutions to problems such as load scheduling, due to its binary nature. It is important to map the load allocation issue in the cloud infrastructure to a binary version in order to solve it.

Response time is a crucial factor in the effectiveness of Cloud Computing load balancing algorithms, and likely involved in resource allocation and load balancing. The response time is the time taken by the Virtual Manager to react to a client's request based on tasks, according to the LB technique. As response time decreases, system efficiency increases, as the VM balances the load

Table 1.
Format for initial solution

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
VM_1	1	0	0	0	1	0	0	0	0
VM_2	0	1	0	0	0	0	0	0	1
VM_3	0	0	0	1	0	0	0	1	0
VM_4	0	0	1	0	0	1	0	0	0
VM_5	0	0	0	0	0	0	1	0	0

between the VMs efficiently. Load balancing is accomplished using BSO. The Dragonfly Algorithm is used in this research to reduce the time it takes for the system to respond to a user request or to emulate the present operation.

After a single iteration, the VM with the lowest load and maximum resources is used to optimize the allocation of resources to balance the load. Following are the steps to take to implement this proposed hybrid approach.

```
Algorithm for Phase 2:
Input: T = {$T_1$, $T_2$, $T_3$. . .$T_n$}
            VM = {$VM_1$, $VM_2$, $VM_3$, . . ., $VM_m$}
Output: The best potential task-to-VM mapping that is balanced
Step 1: Initialization and Particle definition
        for every particle,
Create an N-dimensional vector
$VM = \{VM_1, VM_2, ...., VM_m\}, where VM_j (j \in \{1,2,...m\})$
     represents the number of VM on which task $T_i (i \in \{1,2,...,n\})$        end
     is going to be processed.
for;
Step 2: Estimate the particle's load
$Load\ on VM(LVM_{i,t}) = \frac{N(T,t)}{S(VM_{i,t})}$
Where, $LVM_{i,t} = Load\ of\ VM_i\ at\ time\ t$
        $N(T,t) = set\ of\ task\ at\ time\ t\ on\ service\ queue$
        $S(VM_{i,t}) = Rate\ of\ service\ rate\ of\ VM_i\ at\ time\ t$
Step 3:To keep track of the presence of low-load VMs, run the
birds swarm, such as to test the        load on the VM, followed
by DA to optimize the response times, i.e. depending on
the         response time for the assigning the task. As the
method becomes more proficient,         response times decrease.
The task's execution time is determined by DA.
Step 4: Determine which VMs to be alloted based on the load.
Step 5: If nodes are not overloaded, look for underloaded nodes.
Upon finding an
        Underloaded user, a virtual machine is allocated and the
process continues until no
        Underloaded VMs are found.
Step 6: Find VM to transfer task
Step 7: Task Migration based on VM group
Step 8: Continue until the whole task queue is completed.
```

As a result of its load scheduling capabilities, the D2B_CPU based BSO-DA reduces response times for cloud requests. By utilizing a wider search area, users get full satisfaction. The results are presented in the upcoming sections. The VMs benefit from this technique since they can be optimized and reduced in time.

SIMULATION RESULTS

The Results of the experiment and implementation specifics about the proposed LBA are evaluated in this section.

Experimental Setup

Experiments were conducted in a simulated environment using Java (jdk 1.8) with the Cloudsim tool. CloudSim is a simulation tool used by developers and researchers to solve cloud-related problems. The research solution can be modeled and evaluated without the need and expense of computing facilities. The simulation tool can be imported into software i.e., Maven, Eclipse etc. Eclipse is used to simulate the Cloud infrastructure, and Windows 10 is the operating system.

To assess the performance of this algorithm, we modeled entities and computing resources in a cloud infrastructure to simulate load balancing and scheduling. Table2: Hardware requirements. Table 2 illustrates the hardware requirement.

Parameter Settings

To determine system load and manage task migration, load balancing necessitates the use of several metrics and techniques. Metrics such as throughput, workflow time, response time and processor performance are affected by balancing the load. This research is focused on execution time, running time DoI, throughput, makespan, speedup and Responding time of hybrid HHHOPIO, HHO, SMA and the D2D_CPU based BSO-DA algorithm. In table 3, listed out the types of entities, parameters, and their corresponding values.

Performance Metrics

The proposed algorithm and other algorithms are evaluated using the following metrics.

1. Response time: Average response time:

$$T_{resp} = \frac{\sum_{j=1}^{n} r_j}{n} \tag{6}$$

Table 2.
Hardware requirements

Component	Specification
Operating System	*Windows(X64 based Processor) 64-bit OS*
Processor	*Intel(R) Core(TM) CPU @ 2.60GHz with 8 GB RAM*
RAM	*8 GB*

Table 3.
Parameter setting for cloud simulator

Entities	Parameters	Values
Cloudlets/tasks	Task Length	5000-1000000
	Total set of tasks	100-500
Virtual machine (VM)	Set of VM	30
	Bandwidth	200,000
	Storage	300 GB
	MIPS/PE	400
	Policy type	*Cloudlet scheduler dynamic workload*

2. Makespan: The time it takes for a cloudlet to be scheduled

$$MT = Max(CT)$$

$$MT_{avg} = \left(\frac{\sum Max(CT)}{n} \right) \tag{7}$$

3. Throughput: Throughput time is calculated using formula (8)

$$Throughput\ \ time = \frac{m}{\max_{1 \le i < m} \{FT_i\}} \tag{8}$$

4. Degree of Imbalance: This indicator measures how evenly tasks are distributed among VMs.

$$DOI = \frac{T_{\max} - T_{\min}}{T_{avg}} \tag{9}$$

$$T_i = \frac{L}{PE_{num_i} \times PE_{MIPS_i}} \tag{10}$$

5. $Execution\ \ time = \dfrac{Task}{\Pr oces \sin g\ \ speed\ \ of\ \ VM}$ (11)
6. Load balance: LB is a critical factor that has a long-term impact on a decentralised system's performance and efficiency.
7. Speedup: This is the ratio of the sequential schedule length calculated by assigning all tasks to the speediest processor to the task schedule's execution time (makespan).

$$Speedup = \frac{\min_{p \in H} \left(\sum_{vi \in V} W_{i,j} \right)}{makespan} \tag{12}$$

DISCUSSIONS

Simulation Analysis with Various Algorithm

According to the proposed methodology, each task is alloted to a Virtual Machine depending on its load (capacity). On the basis of load and execution time, tasks are assigned to VMs. To evaluate our proposed work, it is compared with another existing algorithm. We analyze two different configurations, are as follows (i) VM = 15, PM = 5 and 500 tasks and (ii) VM=30, PM = 10 and 500 tasks.

VM = 15, PM = 5 and 500 tasks

In order to analyze experiments, VM = 15 and PM = 5 and 500 cloudlets were used. The main goal is to distribute the five hundred tasks to the appropriate VMs. This configuration is shown in figures 2, 3, 4, 5, 6 and 7. In Table 4, response time for cloudlets utilizing D2B_CPU based with hybrid binary BSO-DA and existing Load Balancing Algorithm are computed.

Table 4.
Response time analysis of existing and proposed approaches

No. of tasks	Response time (milliseconds)			
	SMA	HHO	HHHOPIO	Proposed
100	7364	6171	4293	3124
200	8528	7211	5525	4462
300	9131	8168	6144	6108
400	10,225	9244	7123	7118
500	11,247	10,556	9366	9245

Response times for various tasks are listed here. According to the table above, it can be seen that with D2B_CPU based on the hybrid BSO-DA algorithm, the number of requests queued has decreased and device response time has increased on comparing with other existing algorithms. Therefore, the proposed algorithm is better suited for balancing the load than existing algorithms. Figure 2 illustrates the response times for proposed and existing approaches.

Based on comparisons between the proposed algorithm and HHHOPIO, HHO and SMA, it is clear that D2B_CPU based on hybrid BSO-DA results in a significant improvement over existing algorithms, as shown in Figure 3. Rather than focusing solely on fast machines, which could overburden one machine over another and slow down overall performance, the suggested approach always selects the most appropriate virtual machines to accomplish the tasks (i.e., increase execution time).

Figure 4 shows the DoI among the VMs. Every algorithm exhibits a different DOI with the set of tasks ranging between 100 and 500. The makespan of the proposed load balancing approach is of about 2.8, 3, 2.7, 2.6, 2.5 for 100, 200, 300, 400 and 500 tasks respectively. Similarly for Spider Monkey Algorithm takes about 3.4, 3.6, 3.7, 3.9, 3.8 milliseconds, HHO takes about 3.2, 3.4, 3.5, 3.1, 3.3 milliseconds, and HHHOPIO takes 3, 3.1, 3.2, 2.8 and 3 milliseconds for 100, 200, 300, 400 and 500 tasks respectively, indicating that our proposed approach is less imbalanced than the existing solutions.

Figure 2.
Comparative analysis of response time

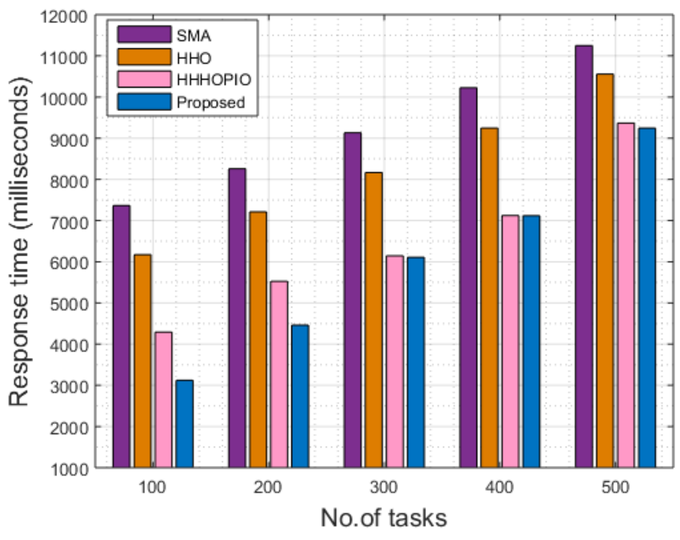


Figure 3.
Comparative analysis of makespan

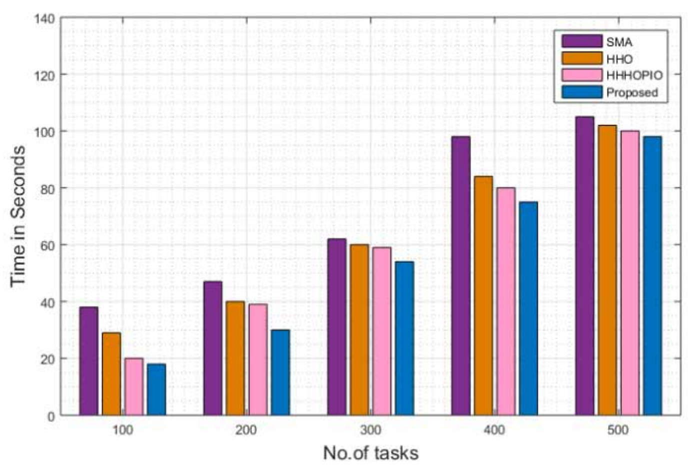
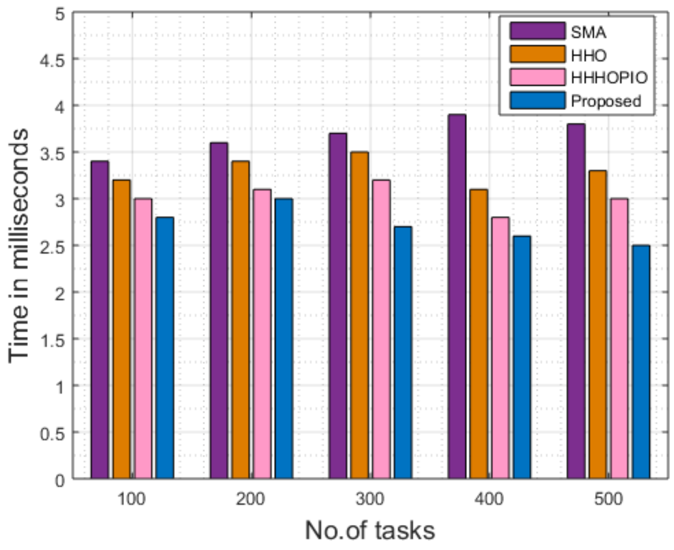


Figure 4.
Comparative analysis of degree of imbalance



As depicted in Figure 5, a detailed average throughput analysis of various approaches was conducted between D2B CPU with hybrid BSO-DA and existing methods. Both the SMA and HHO models performed poorly for 100 tasks, achieving a minimal throughput of 63% and 75%, respectively. Simultaneously, the HHHOPIO approach attempted to perform well by achieving a slight improvement in throughput of 80%. In contrast, the proposed model demonstrated greater performance by obtaining a maximum throughput of 96.5% for 100 tasks. Under 500 tasks, however, both SMA and HHO models had poor average throughput, with a minimal average throughput of just 32% and 38%, respectively. Simultaneously, both HHHOPIO methods attempted to perform better by achieving a modest increase in throughputs of 48%. Under 500 tasks, however, the proposed model had the highest average throughput of 73% by obtaining higher performance.

Figure 6 demonstrates that the suggested optimal load balancing algorithm based on D2B CPU and hybrid BSO-DA takes 31.25 percent less time to execute than existing approaches such as HHHOPIO, HHO and SMA algorithms.

As a result of the findings, it is obvious that the novel technique outperforms other existing methods.

VM = 30, PM = 10 and 500 tasks

Figure 5.
Comparative analysis of throughput

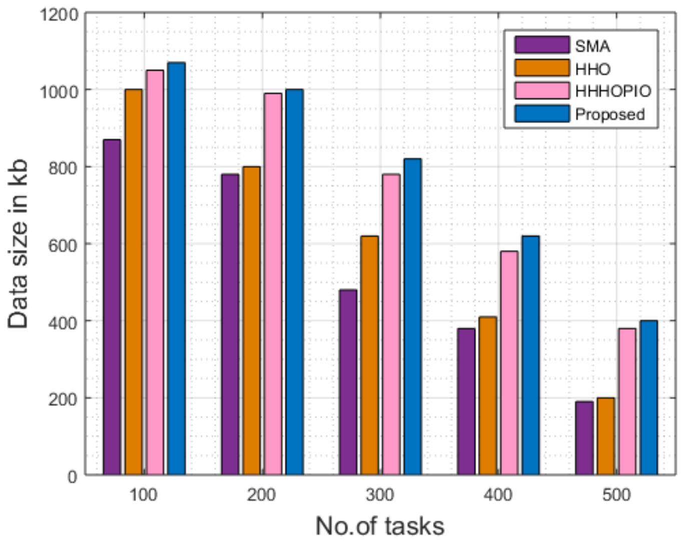
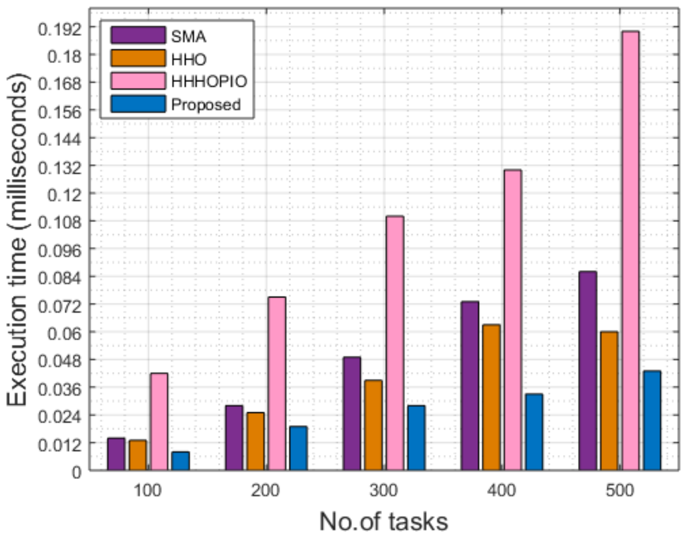


Figure 6.
Comparative analysis of execution time



For this experiment, 500 tasks, 10 PMs and 30 VMs were used. These machines were used to schedule tasks. Results are shown in Figures 8-13. Table 5 shows the response time of cloudlets using D2B CPU, which is based on a hybrid BSO-DA and an existing Load Balancing Algorithm.

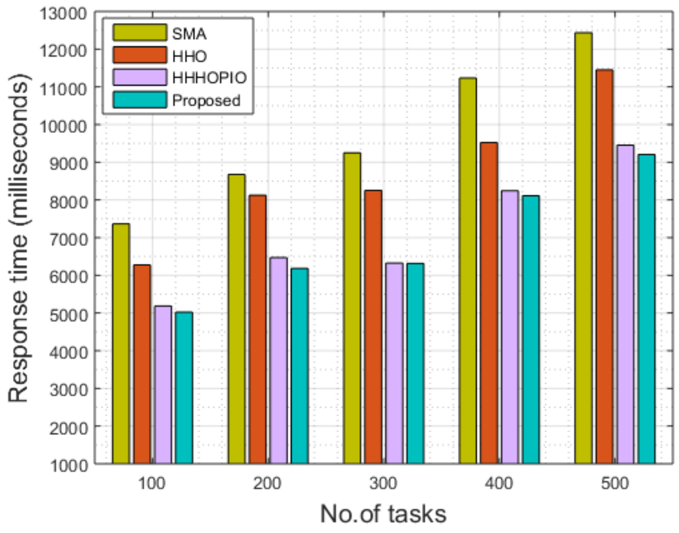
The varying response times for various jobs are tabulated above. Figure 7 shows the response time values acquired for a set of cloudlets utilizing various techniques, including the proposed algorithm. The experiment reveals that the novel algorithm outperforms other algorithms in terms of response time. The simulation results proved that when the amount of cloudlets grows, the performance of the other comparison algorithms improves in terms of response time. However, the proposed strategy outperforms the challenge.

Figure 8 provides an in-depth comparison of various scheduling approaches using D2B CPU utilizing the hybrid BSO-DA method based on make span. The graph demonstrates that the proposed approach produced significantly better outcomes than the competing scheduling techniques. On comparing the results for 500 tasks, the SMA and HHO approaches required a maximum make span of 98 and 94, respectively. Next, HHHOPIO techniques with a make span of 86 required a considerably shorter make span. Nevertheless, the presented algorithm proved effective with a minimum make span of 81. It should be noticed that the makespan duration increases gradually as the number of tasks increases.

Table 5.
Response time analysis of existing and proposed approaches

No. of tasks	Response time (milliseconds)			
	SMA	HHO	HHHOPIO	Proposed
100	7368	6278	5185	5024
200	8678	8125	6471	6182
300	9251	8254	6325	6318
400	11,235	9524	8243	8112
500	12,436	11,452	9452	9206

Figure 7.
Comparative analysis of response time



In figure 9 and 10, throughput and execution times of all algorithms, viz. D2B_CPU based with hybrid BSO-DA and HHHOPIO, HHA, SMA algorithms are compared and presented respectively. Cloudlets varied in count from 100 to 500. The proposed methods are found to gain effective result for both performance parameters i.e. execution time and throughput time.

Figure 10 shows a thorough comparison of the execution times of various scheduling methods and the proposed algorithm. The graph demonstrates that the proposed method significantly outperformed existing algorithms. The maximum execution time required by SMA and HHO techniques was 0.015ms and 0.014ms, respectively, when the results were evaluated. The HHOPIO algorithm showed slightly a higher execution time of 0.04 milliseconds. However, the proposed method produced a great outcome in 0.013ms. In contrast, when measuring the execution times for 500 tasks, SMA and HHO techniques required 0.078ms and 0.080ms, respectively. The HHHOPIO method required a

Figure 8. Comparative analysis of makespan

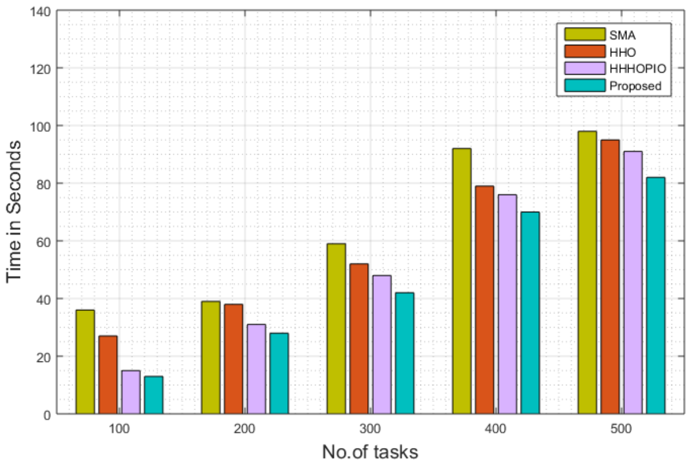
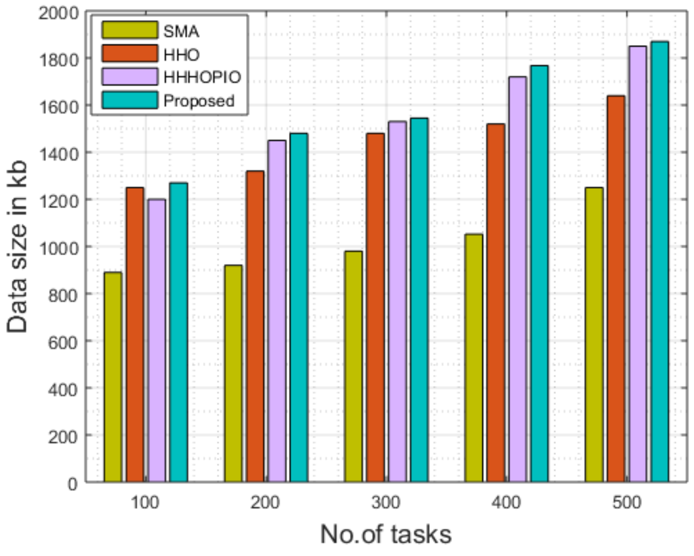


Figure 9. Variation in throughput vs no. of tasks



higher execution time of 0.21ms. However, the proposed algorithm obtained a better outcome with an execution time of 0.045 milliseconds.

Figure 11 depicts the relationship between changes in the DoI and variations in the set of cloudlets. Figure 11 shows that the suggested algorithm has the least DoI for all sets of tasks ranging from 100 to 500. Furthermore, present algorithms are found to have the highest DoI for all tasks in the range of 100 to 500. As a result, the proposed algorithm-based approach reduces imbalance.

Figure 10.
Variation in time for execution vs no. of tasks

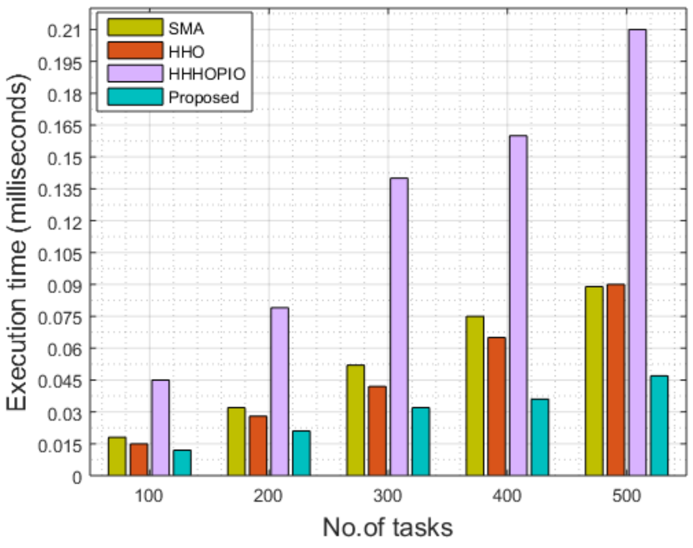


Figure 11.
Variation in DoI vs no. of tasks

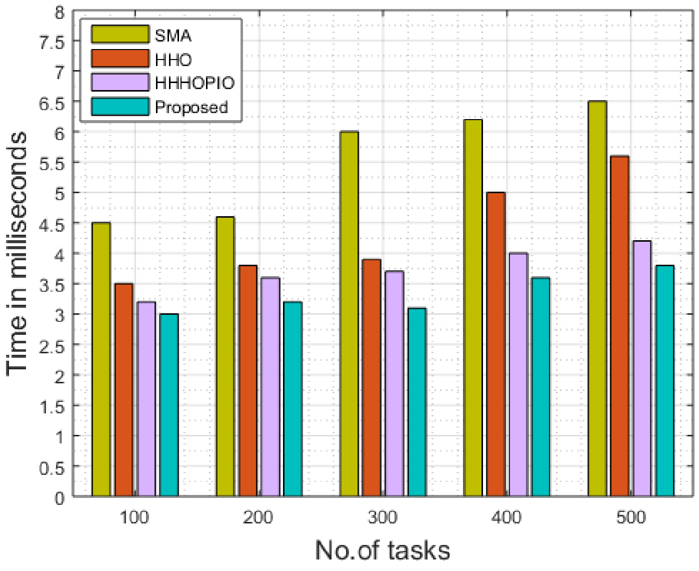


Figure 12 displays the comparative experiment on efficiency of the proposed algorithm on comparing with other existing approaches. It shows that the efficiency of HHO is about 89%, efficiency of PIO is about 92%, ACO is about 94%, and HHHOPIO is about 97% and proposed algorithm outcomes the other algorithm with an efficiency of about 98.2%.

According to Table 6, the proposed algorithm has a significantly least execution time than HLBZID, FSA, and RR. This indicates that the proposed method can execute quickly. Nevertheless, the proposed approach often produces the best results. For 100 tasks, the running time of the proposed method is approximately 270.45 seconds, and for 200 tasks, it is approximately 573.25 seconds. The lesser execution time of the presented method on comparing to the existing HLBZID with 574.5 seconds, FSA with 1595.21 seconds, and RR with 696.64 seconds demonstrates its superiority. Similar analyses have been conducted for 300, 400, and 500 tasks, and there is only a minor change in running time.

Furthermore, as shown in Table 7, the proposed algorithm exhibits a significant speedup improvement. This enhancement demonstrates that the suggested technique is quite effective. For 100 jobs, the proposed method outperforms HLBZID, FSA, and RR by 3.87%, 7.16% and 15.95%, respectively. Similarly, the proposed approach outperforms HLBZID, FSA, and RR for 500 tasks by 2.72%, 9.42%, and 17.82%, respectively.

As a result of the experiment, it is obvious that the novel hybrid effort achieves better outcomes than other works.

Figure 12.
Efficiency comparison with various algorithms

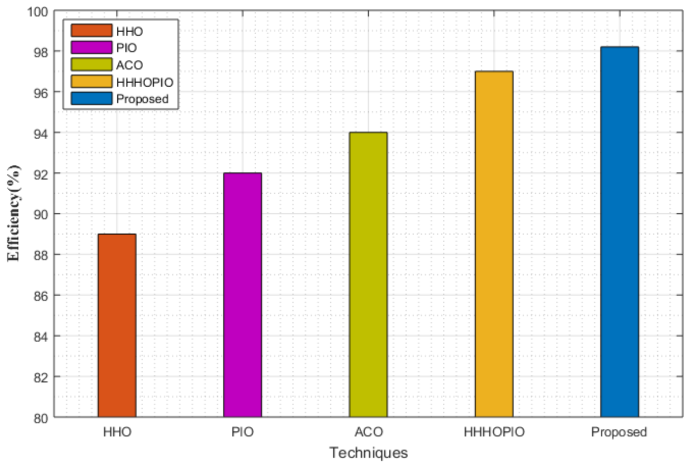


Table 6.
Running time analysis of existing and proposed approaches

No. of tasks	Running time (Sec)			
	HLBZID	FSA	RR	Proposed
100	274.05	836.54	286.73	270.45
200	574.5	1595.21	693.64	573.25
300	788.07	2119.29	915.91	781.52
400	2788.63	6237.17	3049.39	2640.36
500	6288.3	13746.23	7299.9	5980.12

Table 7.
Speedup analysis of existing and proposed approaches

No. of tasks	Speedup			
	HLBZID	FSA	RR	Proposed
100	904	418	198	920
200	935	425	205	947
300	940	430	213	956
400	967	444	223	975
500	982	450	229	994

CONCLUSION AND FUTURE WORKS

Load Balancing and scheduling are the most crucial task in the CC environment. Various research approaches for load balancing and scheduling were presented, however each had limitations, such as failing to account for task scheduling time fluctuations and virtual machine parameters. To address this problem, this study suggested a metaheuristic optimization technique that combined job scheduling with load balancing. Based on D2B_CPU-based BSO-DA, this paper proposes a hybrid method for Load Balancing to reduce the time it takes for every client request to be processed. The incoming requests were balanced optimally utilizing the binary Bird Swarm Optimization and Dragonfly algorithm based on the overloaded and underloaded VM's. This approach is aimed at improving important parameters in CC: throughput, makespan, response time, DOI, execution time, running time, speed, and efficiency. To evaluate the performance of the proposed method, various simulation parameters, such as the number of tasks, virtual machines, and data size, are varied throughout the experimentation phase. Two cloud configurations are evaluated using CloudSim for the purpose of experimenting with VM allocation and task scheduling methods. For makespan, response time, load balancing, DOI, throughput, and execution time, the researchers compared the proposed technique to known algorithms (hybrid Harries Hawks Optimization and Pigeon inspired Optimization, Spider Monkey Algorithm and Harries Hawks Optimization). Although the explained proposal has demonstrated significant improvements in throughput, makespan, response time, degree of imbalance, speedup, efficiency, execution time, and running time, the datacenter's power consumption remains one of the key factors that has a significant impact on load balancing's effectiveness. Therefore, future work will comprise of extending our proposed method by including datacenter power usage and VM live migration.

Conflict of interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

ACKNOWLEDGMENTS

We declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere.

AVAILABILITY OF DATA AND MATERIAL

Not applicable

CODE AVAILABILITY

Not applicable

AUTHORS' CONTRIBUTIONS

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

ETHICS APPROVAL

This material is the authors' own original work, which has not been previously published elsewhere. The paper reflects the authors' own research and analysis in a truthful and complete manner.

REFERENCE

Arora, P., & Dixit, A. (2020). An elephant herd grey wolf optimization (EHGWO) algorithm for load balancing in cloud. *International Journal of Pervasive Computing and Communications*.

Daming, L., Su, Q., Deng, L., Cai, K., Cai, Z., & Mohammed, B. O. (2020). Load balancing mechanism in the cloud environment using preference alignments and an optimisation algorithm. *IET Communications*, *14*(3), 489–496. doi:10.1049/iet-com.2019.0800

Devaraj, A., Saviour, F., Elhoseny, M., Dhanasekaran, S., Lydia, L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing*, *142*(1), 36–45.

Fatima, A., Javaid, N., Butt, A. A., Sultana, T., Hussain, W., Bilal, M., Akbar, M., & Ilahi, M. (2019). An enhanced multi-objective gray wolf optimization for virtual machine placement in cloud data centers. *Electronics (Basel)*, *8*(2), 218.

Gupta, A., Bhadauria, H. S., & Singh, A. (2021). Load balancing based hyper heuristic algorithm for cloud task scheduling. *Journal of Ambient Intelligence and Humanized Computing*, *12*(6), 5845–5852. doi:10.1007/s12652-020-02127-3

Janakiraman, S., & Deva Priya, M. (2021). Improved Artificial Bee Colony Using Monarchy Butterfly Optimization Algorithm for Load Balancing (IABC-MBOA-LB) in Cloud Environments. *Journal of Network and Systems Management*, *29*(4), 1–38. doi:10.1007/s10922-021-09602-y

Jena, U. K., P. K. Das, and M. R. Kabat. (2020). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*.

Jena, U. K., P. K. Das, and M. R. Kabat. (2020). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*.

Kaur, A. & Kaur, B. (2019). Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment. *Journal of King Saud University-Computer and Information Sciences*.

Kaur, A., Kaur, B., Singh, P., Devgan, M. S., & Toor, H. K. (2020). Load balancing optimization based on deep learning approach in cloud environment. *International Journal of Information Technology and Computer Science*, *12*(3), 8–18.

Kruekaew, B., & Kimpan, W. (2020). Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. *International Journal of Computational Intelligence Systems*, *13*(1), 496–510. doi:10.2991/ijcis.d.200410.002

Kumar, K. P., Ragunathan, T., Vasumathi, D., & Prasad, P. K. (2020). An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. *Algorithms*, *1*, 423.

Kumar, K. P., Ragunathan, T., Vasumathi, D., & Prasad, P. K. (2020). An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. *Algorithms*, *1*, 423.

Kumar, M., & Sharma, S. C. (2018). Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. *Computers & Electrical Engineering*, *69*, 395–411. doi:10.1016/j.compeleceng.2017.11.018

Milan, S. T., Rajabion, L., Ranjbar, H., & Navimipour, N. J. (2019). Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments. *Computers & Operations Research*, *110*, 159–187.

Negi, S., Rauthan, M. M. S., Vaisla, K. S., & Panwar, N. (2021). CMODLB: An efficient load balancing approach in cloud computing environment. *The Journal of Supercomputing*, *77*(8), 8787–8839.

Ouhame, S., Hadi, Y., & Arifullah, A. (2020). *A hybrid grey wolf optimizer and artificial bee colony algorithm used for improvement in resource allocation system for cloud technology*.

Phi, N. X., Tin, C. T., Luu, N. K. T., & Hung, T. C. (2018). Proposed load balancing algorithm to reduce response time and processing time on cloud computing. *Int. J. Comput. Netw. Commun*, *10*(3), 87–98.

Polepally, V., & Shahu Chatrapati, K. (2019). Dragonfly optimization and constraint measure-based load balancing in cloud computing. *Cluster Computing*, *22*(1), 1099–1111. doi:10.1007/s10586-017-1056-4

Pradhan, A., &Bisoy, S. K. (2020). A novel load balancing technique for cloud computing platform based on PSO. *Journal of King Saud University-Computer and Information Sciences*.

Priya, V., Sathiya Kumar, C., & Kannan, R. (2019). Resource scheduling algorithm with load balancing for cloud service provisioning. *Applied Soft Computing*, *76*, 416–424. doi:10.1016/j.asoc.2018.12.021

Ragmani, A., Elomri, A., Abghour, N., Moussaid, K., & Rida, M. (2020). FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, *11*(10), 3975–3987. doi:10.1007/s12652-019-01631-5

Raja, R., & Karthikeyan, A. (2020). An improved GSO based task scheduling (IGSOTS) algorithm for load balancing in cloud environment. *PAIDEUMA J*, *13*(5), 12–30.

Rawat, P. S., Dimri, P., & Saroha, G. P. (2020). Virtual machine allocation to the task using an optimization method in cloud computing environment. *International Journal of Information Technology*, *12*(2), 485–493. doi:10.1007/s41870-018-0242-9

Ruan, X., Chen, H., Tian, Y., & Yin, S. (2019). Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds. *Future Generation Computer Systems*, *100*, 380–394. doi:10.1016/j.future.2019.05.036

Sahana, S., Mukherjee, T., & Sarddar, D. (2020). A conceptual framework towards implementing a cloud-based dynamic load balancer using a weighted round-robin algorithm. [IJCAC]. *International Journal of Cloud Applications and Computing*, *10*(2), 22–35.

Ullah, A., & Chakir, A. (2022). Improvement for tasks allocation system in VM for cloud datacenter using modified bat algorithm. *Multimedia Tools and Applications*, 1–15.

Xingjun, L., Shao, Z., Cheng, H., & Mohammed, B. O. (2020). A new fuzzy-based method for load balancing in the cloud-based Internet of things using a grey wolf optimization algorithm. *International Journal of Communication Systems*, *33*(8), e4370. doi:10.1002/dac.4370

Xu, P., He, G., Li, Z., & Zhang, Z. (2018). "An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization." *International Journal of Distributed Sensor Networks,* 14(12), 1550147718793799.

Xue, L. S., Majid, N. A. A., & Sundararajan, E. (2018)."Dynamic virtual machine allocation policy for load balancing using principal component analysis and clustering technique in cloud computing. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10*(3-2), pp. 47-52.

Ziyath, S., & Senthilkumar, S. (2021). MHO: Meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services. *Journal of Ambient Intelligence and Humanized Computing*, *12*(6), 6629–6638.

Thanwamas Kassanuk is a Faculty of Food and Agricultural Technology, Pibulsongkram Rajabhat University, Phitsanulok, Thailand

Khongdet Phasinam is currently working as an Assistant Professor in the School of Agricultural and Food Engineering, Faculty of Food and Agricultural Technology, Pibulsongkram Rajabhat University, Thailand. He received his B.Eng. degree in agricultural engineering, the M.Eng. degree in energy management engineering, and the Ph.D. degree in agricultural and food engineering from Suranaree University of Technology (Thailand) in the year 2007, 2010, and 2016, respectively. His current research interests include power and machinery, computer aided design, renewable energy, and smart farm systems.