SN Computer Science (2021) 2:395
https://doi.org/10.1007/s42979-021-00771-w

ORIGINAL RESEARCH

A Double Threshold-Based Power-Aware Honey Bee Cloud Load Balancing Algorithm

Anindita Sarkar Mondal[1] · **Somnath Mukhopadhyay**[2] · **Kartick Chandra Mondal**[1] · **Samiran Chattopadhyay**[1]

Received: 28 April 2021 / Accepted: 11 July 2021 / Published online: 30 July 2021

Abstract

Present-day advancement in cloud computing provides ICT infrastructure as a service on a pay per use. Cloud computing provides this infrastructure as a service and as service demand increases, service providers organize large-scale data centers with a lot of resources, and cause of huge greenhouse gases' emission. This data center's huge power demand necessitates the balancing of cloud load. To attain the optimum resource utilization, least processing time of CPU, minimal average response time, and avoiding over-load, cloud load balancing algorithms distributes workload across virtual machines. The key challenge here is to develop such a load balancing algorithm which consumes the least resources to fulfill the service demands. In this paper, a double threshold-based power-aware honey bee load balancing algorithm is proposed for the fair and even distribution of the incoming task requests to all the virtual machines. This paper compares the proposed algorithm with five widely used existing load balancing algorithms. Moreover, we have done the performance analysis using the popular CloudAnalyst simulation toolkit. Results of simulation showed that the proposed algorithm gives a note-worthy outcome for average response time, CPU cost, storage cost, memory cost, and energy consumption in cloud computing to show the resource utilization.

Keywords Cloud computing · Load balancing · CloudAnalyst · Performance comparison

Introduction

Cloud computing provides an information and technological pay-per-use type services [3, 5, 16, 59]. According to the demand of users, software applications, infrastructure, and platforms are served using the Internet. Cloud computing's rudimentary requirement is sharing and providing computational resources such as virtual machines (VMs) based on user demand. Efficient VM allocation for user's request is being carried out using different load balancing mechanisms in cloud computing. It is necessary to attain cloud computations' potentials using effective scheduling mechanisms for minimizing the job execution time. As the number of users increases, the job requests are to be scheduled to increase equally, and the scheduling algorithms cannot achieve their requirements.

Therefore, there is a need for more efficient task scheduling algorithms to minimize the time of computation, energy consumption, and the overall processing cost. A good algorithm of task scheduling can influence the whole cloud system directly. One best example can be a swarm intelligent mechanism [19, 26], i.e., bee colony optimization algorithm [31, 64] that is used in this paper for the development of our new proposed algorithm. Moreover, load balancing and management of cloud resources are crucial research areas in the cloud environment to distribute the workload equally, maximize the rate of resource utilization, and minimize task

"This article is part of the topical collection "Next-Generation Digital Transformation through Intelligent Computing" guest edited by PN Suganthan, Paramartha Dutta, Jyotsna Kumar Mandal, and Somnath Mukhopadhyay".

✉ Anindita Sarkar Mondal
sarkar.anindita5@gmail.com

Somnath Mukhopadhyay
som.cse@live.com

Kartick Chandra Mondal
kartickjgec@gmail.com

Samiran Chattopadhyay
samirancju@gmail.com

[1] Department of Information Technology, Jadavpur University, Kolkata, India

[2] Department of Computer Science and Engineering, Assam University Silchar, Silchar, Assam, India

execution [6, 22, 49, 55]. Therefore, efficient and effective use of optimized load balancing algorithms will provide maximum usage of available resources and thus enhance the system's overall performance and throughput. The load balancing mechanism's primary goal is to distribute the load equally among all nodes for optimizing the resource's service time and the application's response time.

Cloud computing provides computing resources as a utility based on Service Level Agreement (SLA) between users and its cloud service provider. The Amazon EC2 [35], Google App Engine [72], and Microsoft Azure[1] are few major cloud service providers that provide Platform, Infrastructure, and Software-oriented services. Most IT organizations are outsourcing their management to the cloud to avoid the initial capital investment on infrastructure setup and reduce software and hardware maintenance. Thus, the necessity to increase computing services forced service providers to set up highly computational powered large-scale data centers. A huge amount of electrical power is necessary to keep these data centers always in up condition and functioning correctly, resulting in operational cost increment. Additionally, these data centers also produce and release greenhouse gases like carbon dioxide. This appeared as a significant challenge to the cloud service provider. Since idle servers can also consume 50% of the fully utilized power and 5–15% of these idle servers can be needlessly running in data center [9, 11]. Hence, the management of power has become a challenging issue. Virtualization technique is the mainstay of cloud computing that helps for efficient resource utilization by allocating VMs to a single host.

Due to the dynamic workload in a cloud environment, there might exist some unnecessarily running hosts in the cloud system. There can be some hosts that are underutilized and some hosts that are over-utilized in a data center. Therefore, to attain load balancing and to avoid needless power consumption, and get efficient resource utilization, the concept of the live VM migration [40] can be implemented. The technique of migrating VM from one host to another physical host keeps the user still connected, using minimum down-time. Using this live VM migration technique, the VMs from the under-loaded and the over-*loaded host can be allocated to the appropriate server. Therefore, the needlessly running hosts can be switched off. The over VM consolidation process using live VM migration results in degradation of performance. This paper presents the comparative analysis of different provisioning mechanisms to accomplish energy-performance trade-off maintaining the service-level agreement.

Testing and analyzing various scheduling and allocation algorithms for developing applications in a real cloud environment are a truly challenging issue. The main part of the cloud environment is the cloud storage system, such as Object-Based Schema-Oriented Data Storage System (RSoS System), [47, 48] and Openstack Swift [4]. Since most cloud manifest applications, changing a number of requests incoming and testing algorithms in a real cloud environment results in a lot of costs. The effectiveness of testing an algorithm for implementation in the cloud needs a simulation environment that can provide an environment that is close to the real cloud and can produce results that help in the analysis of the algorithms, so that they can be deployed on real clouds. The CloudSim is a toolkit that supports the modeling of systems and characteristics of cloud systems like virtual machines (VMs), data centers, and resource allocating algorithms. It implements generic application provisioning mechanisms that can be easily extended with minimum effort. It provides both modelings and simulation in cloud environments and also it reveals custom interfaces to implement the algorithms and provisioning mechanisms for allocating VMs under the cloud computing environment. Several researchers are using CloudSim in their experiment. In absence of this type of simulation platforms, both cloud customers and cloud providers have to rely either on theory and evaluations lacking exactness and accuracy, or on try-error techniques which results to ineffective performance and creation of revenue.

There is an absence of tools for evaluating the needs of large-scale applications in the cloud with respect to the geographical distribution of workloads for users and servers. For filling this disadvantage in tools to evaluate and model applications and cloud environments, CloudAnalyst [70] had been suggested. It was developed to simulate large-scale cloud applications and study the functioning of these applications with respect to several deployment configurations. Developers can get help from CloudAnalyst about the insights of distribution of applications across the infrastructures of cloud and performance optimization of applications and providers using Service Brokers. CloudAnalyst was extended from CloudSim, and some of its abilities and characteristics are extended. CloudAnalyst isolates the programming process from the simulation process. It allows a modeler to repeatedly perform simulations and perform a sequence of simulations, taking little variations of parameters in a fast and easy way. It is applied for inspecting characteristics of large-scale applications of the cloud.

In this paper, the performance of cloud load balancing is calculated for the distributed data centers using the CloudSim toolkit program. The performance analysis and simulation will be performed using the CloudAnalyst tool. It provides an easy way to evaluate new algorithms in the utilization of clouds, taking into consideration load balancing and scheduling algorithms. This can also be used to evaluate algorithms' competence from various aspects

[1] https://docs.microsoft.com/en-us/azure/storage/.

such as cost, application response time, execution time, etc. This also provides support for the evaluation of the Green IT strategies [15, 67]. The user uses it as the blocks for building a simulated environment and can add new algorithms for scheduling and load balancing. Thus, it is reasonably flexible for being used as a library by allowing writing the desired scenario using the Java program.

This work's specific contributions include a systematic study of the proposed double threshold-based power-aware approach of load balancing mechanism in a cloud environment. The flowcharts portraying its functional control structures show how power-aware strategies and bees foraging behavior inspire load balancing for a distributed cloud system; evaluation and comparative analysis of the performance of the proposed load balancing technique with respect to other load balancing mechanisms using CloudAnalyst simulator toolkit based on various simulation parameters. The key contributions of this article are presented below.

- To enhance the cloud computing performance in data transfer cost, memory cost, storage cost, CPU cost, response time, and processing time uniform distribution of tasks within the available virtual machines of a data center is made through the task scheduling approach.
- Also, shows how the reduction in power consumption of the data center is handled through the task scheduling approach.

The rest of this paper is organized as follows. Section "Issues and Problems Related to Load Balancing in Cloud" discusses some issues that are related to the load balancing in the cloud environment. This section also discusses various parameters considered in the load balancing and provides the problem formulation of this article. Next, in Section "Related Work", a brief study on the researched topic has been presented here. After that, section "Existing Load Balancing Algorithms" gives a comparative study of existing and proposed algorithms concerning their pros, cons, and various load balancing metrics. Then, in section "Materials and Methods", we have presented the proposed load balancing algorithm's flowcharts and modified the proposed load balancing algorithm. The simulation configuration and result analysis are presented in section "Simulation and Results Analysis" which deals with the discussion about performance analysis of the proposed algorithm along with other considered load balancing algorithms. In the end, Section "Conclusion" concluded with some future scope of the work.

Issues and Problems Related to Load Balancing in Cloud

Necessity of Load Balancing

Load balancing in a cloud environment is a technique of distributing the surplus local dynamic workload equally among all the nodes and used to accomplish a high resource utilization and satisfaction of user ratio. It also ensures that no single node gets over-loaded and enhances the system's overall performance. Proper balancing of load helps to optimally utilize the resources available, thus minimizing the consumption of resources. Load balancing maximizes throughput, enhances the system's stability and reliability, future modification is accommodated, and for small jobs avoids prolonged starvation. This also helps for fail-over implementation, scalability, avoidance of bottleneck conditions and over-provisioning, reduction of waiting time and response time, etc. Apart from these issues, a load balancing mechanism in the cloud is also required to obtain the green computing [1] that can be accomplished taking the help of the below-mentioned factors:

- Reduction in consumption of energy.
- Reduction in emission of carbon.

Various Parameters Considered in Load Balancing

Various existing load balancing mechanisms [6, 19, 31, 63] in cloud environment consider different types of metrics such as throughput, response time, migration time, performance, resource utilization, scalability, associated overhead, and fault tolerance. These parameters are described about the provided service quality in the cloud system. The detail descriptions of these parameters are presented in Table 1.

The main goal of the proposed algorithm is to design such a task scheduling approach which not only minimize the response time but also make beneficial to the service consumer by reducing the service cost. For this reason, some more parameters are considered, as memory cost, CPU cost, storage cost, and data cost these are described about the amount of cost is needed to accomplish a task. The mathematical formulae of the load balancing parameters are presented in Listing 1.

```
Data Transfer Cost (DTC) = ( total_data / (1024 * 1024) ) *
    cost_per_data_GB ;

CPU Cost (CC) = ( total_time_in_ms / (1000 * 3600) ) *
    cost_cpu_per_hour ;

Memory Cost (MC) = (total_time_in_ms / 1000 ) * cost_per_memory ;

Storage Cost (STC) = ( total_time_in_ms / 1000 ) *
    cost_per_storage ;

Total Cost (TC) = DTC + CC + MC + STC ;

Throughput= Total accomplished task / time duration

Resource Utilization = processing time/ (processing time+Idle time
    )

Migration time=Total traverse time/(number of nodes * number of
    tasks)

Energy Consumption= Energy consumption in idle time + Energy
    consumption in processing time
```

Research Issues Related to Load Balancing

Various research issues should be taken into account in developing a load balancing algorithm that can help obtain an optimal solution. These are mentioned below.

- In implementing a load balancing algorithm, the distance between the cloud nodes should be considered. The algorithm developed should work effectively and efficiently in case nodes are far away from such as the Internet and for nodes close to each other such as the Intranet.
- Implementation mechanism and operational mechanism of algorithm development should not be complicated, since it may cause degradation in its performance.
- Single point of failure and deadlock condition should be avoided in the proposed algorithm of load balancing.
- Algorithm should take care of all possible scenarios in the cloud environment that can enhance the load balancing parameters as Table 1.

Table 1 Description of load balancing parameters involved in cloud environment

Parameter name	Description	Value to be
Overhead associated	Specifies the amount of overhead associated with the execution of a load balancing algorithm	Minimized
Throughput	Used for calculating the no. of tasks that completed execution successfully	Maximized
Resource utilization	Used for checking the amount of utilization of resources	Optimally maximized
Scalability	The capability to distribute the tasks of load balancing within a limited number of resources	Improved for better performance
Response time	The total time is needed to respond by a particular load balancing mechanism in a system	Minimized
Fault tolerance	The capability to achieve load balancing uniformly despite of arbitrary node or link failure	Maximized
Migration time	The time is taken for the migration of a task or resource from one node to another node	Minimized
Performance	Used for checking the efficiency of the system	Maximized at a cost that is reasonable
Energy consumption	The total amount of energy is taken by all the resources of a system. It helps to avoid overheat generation and reduce consumption of energy	Minimized
Carbon emission	Calculates the amount of carbon emission by all existing resources in the system	Minimized

Problem Formulation

The primary goal of the cloud computing system [10, 13, 14] is to provide the services in such way that it makes the economic benefits of cloud service consumers by utilizing the available resources in optimized manner. The resource allocation and deallocation mechanism is needed for avoiding underutilized or over-utilized conditions of the resources which affects the cloud services. Random selection of the resources for handling the load makes some resources are over-loaded or under-loaded or idle. Evenly distribution of load enhances performance by migrating load from over-loaded server to under-loaded server. Thus, it leads to the development of many algorithms for scheduling and load balancing as mentioned next in Section "Related Work".

Some authors just focused on the node's accessibility, and only a few factors are considered, such as the node's memory, the capacity of processing, etc. [6, 40]. Thus, some more factors are added here, such as the bandwidth of VM, VM computing capacity that is calculated concerning millions of instructions per second, count of processors in a VM, and VM's storage capacity. All these factors will be used to quickly get the most relevant and available resources for the considered task. Several research papers [25, 27, 29, 73] consider the concept of priority which could cause the increment of the response time. Therefore, Shortest Job Criteria are considered to minimize the average response time of cloudlets. The honey bee foraging concept is to distribute the nodes' workload and search for an optimal path toward the most relevant resource in the cloud. In contrast, the double threshold-based power-aware concept is applied to reduce energy consumption. Performance of the system is calculated based on efficient scheduling mechanism and resource allocation characteristics of cloud computing. These considered characteristics impact cost optimization that can be achieved by enhanced response time and data center processing time.

Related Work

In this section, we describe different cloud load balancing algorithms. Here, the primary focus of this study is on allocating all incoming requests across the available virtual machines that have a minimum response time. Extensive research works will develop the power-aware data center by keeping energy-performance trade-off in the cloud environment. Different studies show that the load balancing is main objectives of optimized scheduling compared to the emission of CO_2, processing power, usage of the fan, and others. The optimization of tasks consists of only the initial allocation of a VM to a host in most studies. Few more research works have been aimed at the issue of rescheduling VMs running on an over-loaded host. There are different algorithms and techniques for the efficient and effective utilization of cloud resources by the consolidation of servers.

The static consolidation process [46] is not a feasible option at the time of VMs live migration. At the initial static mapping, the server consolidation is not done for an extended period of time. Thus, a dynamic VM consolidation is a better option, as shown in article [12]. Live migration of VMs helps switch off hosts when they are under-loaded or over-loaded or both and thus minimize power consumption. DRSQ [49] properly utilizes the resources by assigning tasks to the corresponding practical resource. First work on energy management for the virtualized data center was proposed [50]. Here, the proposed architecture to the data center is the separation of resources at local and global levels. However, the author does not properly mention about the automatic resource allocation mechanism at a global level.

The authors of the article [42] considered the load and suggested a power-aware load balancing algorithm applied to the VM with an upper and lower threshold. Migration occurs when the load crosses the threshold boundary. Here, authors did not consider the CPU utilization and real data center data to run their experiment. Similar work has been done in [20] using a K-nearest neighbor regression mechanism for predicting resource usage of each host. Here, the authors do not describe the no. of VM migration sat the algorithm which is a crucial part for deciding the residual bandwidth availability in a data center. Another work in the same domain was done in [43] where authors mainly concentrated on a lower and upper threshold for minimizing power consumption at a data center and VM migrations. Since a static value is used here, it is not the right solution for increasing CPU utilization and dynamic workload within the data center.

VMware Vsphere distributed energy management [68] operates on an upper and lower threshold that is set at 81% and 45%. It is not acceptable as the utilization may vary differently for various data centers. A mechanism was proposed in [8] that deals with the power and efficient dynamic VM migration problem. Here, the authors proposed several algorithms to detect host over-loading but only a general algorithm for detecting the host under-loading. The consolidation of VM and Shingo can make a reduction of power consumption, and Toshhinori proposed a rank-based method for VM consolidation [62] for this. Here, the VM migration to a suitable destination host is based on host rank.

In [65], authors use backfilling with first-come-first-serve mechanism a combination of runtime and kill times, scheduled shorter tasks before their time if determined not to interfere with other tasks deadlines. Errors exist in run time, and kill times are estimated, which were found to be effective. Suresh et al. in [61] enhanced a backfilling mechanism that places similar length next to each other tasks to ensure

that they complete approximately simultaneously and make free the computation power which larger sized jobs will utilize. These kinds of scheduling mechanisms optimize the make-span. Garg et al. [25] suggested several techniques to find appropriate task placements across the heterogeneous type of hosts with different workloads for minimizing energy consumption. Voltage scaled dynamically was emerged as adjusting the host's voltage for saving energy depending on load [38]. Another power-aware approach like [39] is consolidating VMs to the server for turning off others that had been freed off.

Heuristics attempt for learning some characteristics of a solution for being able to generate better solutions later on. Ant Colony Optimization (ACO) is used a no. of times for scheduling purposes in the cloud. Zhu et al. [73] proposed an ACO-based algorithm in which tasks have metrics of quality of service related to bandwidth, task completion time, cost, and reliability. These properties help for choosing placement among heterogeneous resources. Similarly, Feller et al. [21] modified ACO for a problem of workload balancing. The system allows the same tasks overtime to record the maximum requirements of resource for each job, making their deployment easier once the system has learned the near-optimal solution. Similarly, minimizing both wastages of resource and energy consumption using ACO was proposed in [23].

Various researchers focus on designing distinct algorithm that minimize the energy consumption at task execution time. Some of them are described here. Genetic Algorithms (GA) have also been used in formulations with multi-objectives, as shown in [71]. Here, the authors proposed an algorithm where resource wastage is optimized, consumption of power, and thermal dissipation costs are also optimized. Among other rescheduling techniques, Mi et al. [44] used GA for adjusting the data center as per dynamically changing requests to minimize power consumption. Kusic et al. in [37] suggested an algorithm for minimizing energy consumption by handling the small clusters' applications of servers to ensure SLA levels. This algorithm considered the expected request demand at the time of application allocation and the hosts for VM allocation to avoid the later switching and dynamically adjusts of the no. of VMs. Chaotic Darwinian Chicken Swarm Optimization (CDCSO) algorithm [33] assigned tasks to the virtual machine based on certain multi-objective parameters, namely energy, cost, task completion time, response time, throughput, and load balancing index. This technique reduces the cost, energy consumption, and make-span. Our proposed approach points out another direction for reducing the energy consumption of the data center by shutting down the unallocated or idle VMs.

Additionally, setting rules without any flexibility to learn may not give a response to system changes or requirements of tasks. An adaptive mechanism of rescheduling that minimizes energy without depending on previous knowledge of resource requirements is required to dynamically solve the power efficiency issue. Such a technique can learn and respond to changes in an adaptive way. Sran et al. proposed a load balancer in [60] which controls the payload flow based on static or dynamic thresholds. The author analyzed the existing algorithms like throttled, round-robin, biased random sampling, and equally spread and proposed a new algorithm that increases the performance while decreasing the overall time of requesting and processing. Another load balancing mechanism is presented in [57] for balancing load and task's priorities which are removed from over-loaded VMs. This is based on honey bee foraging behavior which reduces response time and enhances the overall throughput, but the power consumption is not investigated here.

Authors in [69] suggested a green scheduling algorithm in a cloud environment that can optimize the power consumption. Here, for rescheduling of the services an adapted bee colony algorithm and managing of power consumption an ant colony algorithm are used. Contrary, this work use modified bee colony algorithm and other double threshold-based power-aware mechanism for detecting over-utilized hosts, and VM selection. Similarly, in [17], an architectural principle for managing clouds in an energy-aware way was proposed. They also proposed power-efficient resource allocation and scheduling strategies. However, since they used static thresholds of utilization, this technique may not be effective and efficient in cloud environments.

Recent work in [27] presented an adaptive approach for dynamic load balancing of VMs based on analytical, heuristics, and historical data about VM's usage of the resource. Authors proposed mechanisms like Interquartile Range (IQR), Robust Local Regression (RLR), Median Absolute Deviation (MAD), and Local Regression (LR) for the arrangement change over-loading detection host. Additionally, to select the VM, one of the Minimum Migration Time technique [53], Random Choice technique [7], and Maximum correlation technique [32] is used.

Nature inspired in designing the efficient load balancing algorithm in cloud domain. Yao et al. [24] suggested a load balancing technique by concerning an artificial bee colony (ABC) algorithm and presented an enhanced artificial bee colony algorithm for increasing the system's throughput. A load balancing technique using modified PSO task scheduling (LBMPSO) [54] considered the execution time, and starting time variation of the tasks and assigned them to the ideal VMs. It minimized the make-span and maximized the resource utilization value. LBMM [58] designed a task scheduling approach to overcome the load imbalance drawback of the max–min algorithm. As a result, it increased the turnaround and throughput of the system. Binary bird Swarm Optimization-Based Load Balancing (BSO-LB) algorithm [45] applied the concept of the mimicking behavior of a

Table 2 Parametric comparison of a list of parameters, viz., 1 -> fault tolerance, 2 -> migration time, 3 -> overhead, 4 -> performance, 5 -> resource utilization, 6 -> response time, 7 -> scalability, 8 -> throughput

Algorithm	Mechanism	Parameters							
		1	2	3	4	5	6	7	8
RR [29]	Static and centralized	X	X	✓	✓	✓	✓	✓	✓
ESCE [63]	Dynamic and distributed	✓	✓	X	✓	✓	✓	✓	X
TLB [66]	Dynamic and distributed	X	✓	X	✓	✓	X	✓	X
HBF [56]	Dynamic and distributed	X	X	X	X	✓	X	X	X
ACO [51]	Dynamic and distributed	✓	✓	X	✓	✓	X	✓	X
DTPAHBF (proposed)	Dynamic and distributed	✓	X	X	✓	✓	✓	✓	X

flock of birds in cloud domain for load balancing and task scheduling purpose in such a way that minimized the make-span and maximized the resource utilization and throughput value of the system. However, the authors did not consider the energy consumption issue or the SLA violation issue which is considered in our proposed approach.

In the recent era, researchers focused on hybrid algorithms to handle the huge workload in the cloud computing system. Some of them are discussed below. Hybrid Max–Min Genetic Algorithm (HMMGA) [34] combination of max–min algorithm and the genetic algorithm used to balance the load of the available VMs and scheduled the task in such a manner that it reduced the task completion time in between heterogeneous VMs. Authors of the article [18] proposed a load balancing method consists of two approaches, namely autonomous technique for avoiding the assignment of extra request to the VM which creates load redundancy, and prediction technique for predicting the future state of the VMs to minimize the request transfer cost from over-loaded VM to under-loaded VM for reducing the inter-VM communication overhead. Nowadays, the cloud domain faces a big challenge that is huge energy consumption with a huge workload.

Existing Load Balancing Algorithms

In this section, we present five well-known load balancing algorithms of cloud environments that are used for comparison with the proposed load balancing algorithm. However, in this section, we have also done theoretical comparisons on these five load balancing algorithms in terms of their

Table 3 Comparison of pros and cons of different algorithms considered for performance comparison

Algorithm	Pros	Cons
RR [29]	Process of allocation of job, Response time, Resource utilization	No earlier information about any process
	Equal distribution of workload	Job processing time is not considered
ESCE [63]	Enhanced response time	No fault tolerance for the presence of a single point of failure
	Maximum throughput	
	Minimized data transfer cost	
TLB [66]	Distributes the load equally across the VMs	Does not consider the present load on VM that can result in the increased response time for a task
	The load balancer maintains VMs and their corresponding states (Available or Busy)	Higher maintenance is needed for VMs
HBF [56]	VM's waiting time and response time are reduced	Throughput does not increment with the size of the system
		Without VM, high-priority tasks cannot work here
ACO [51]	Under-loaded nodes are identified in the best case	Network overhead is created
	Decentralized	A number of ants and points of beginning can not be perceived easily
	Ensures availability and provides efficient resource utilization	After visiting by ants, status changing of nodes is not considered
	Increases the no. of requests processed and decreases the multiple requests' processing time	Availability of node is only taken into consideration
	Throughput is maximized	
DTPAHBF (proposed)	Efficient	This may initiate unnecessary migration of VMs
	Minimal power consumption	

implementation environment, parameters, and their pros and cons, as shown in Tables 2 and 3, along with the proposed algorithm of this article.

There are various mechanisms that exist for balancing the load in multiple situations. Static mechanisms [6, 46, 55] are very efficient in the case of a stable environment. Whereas the dynamic algorithms uninterruptedly observe the resources during runtime [2, 36, 60]. The dynamic algorithm provides a much better solution to adjust the workload dynamically at runtime. Tables 2 and 3 and a short explanation of each of the algorithms in this section give a better understanding of these algorithms.

Round Robin (RR) Load Balancing Algorithm This algorithm is considered the most fundamental and the least complicated scheduling algorithm. Here, on the concept of time quantum is used, where a quantum of time is assigned for each processor. This technique is distributed among all processors. Moreover, each process is allocated in the processor in the way of rounded ordering. This mechanism of load balancing uses the round-robin concept [29]. If any process does not finish its execution within a given time, then the process will be put at the waiting queue's end position. Round Robin Algorithm [29] selects the load randomly, thus resulting in a condition where some nodes are over-loaded and others are under-loaded, which is the main drawback of the approach. Also, there is an extra load to the scheduler for deciding the quantum size. This technique has a larger average waiting time, longer turnaround time, longer context switches, and lower throughput.

Equally Spread Current Execution (ESCE) Load Balancing Algorithm This algorithm [63] considers the allocation size and distributes the load randomly. After that, the task load is transferred to that VM that is loaded lesser or will handle the task efficiently; taking less time gives maximum throughput. This is the spread spectrum mechanism where the load balancer spreads the job load to multiple VMs. In this technique, the load balancer keeps a VMs table and the currently allocated request numbers to the corresponding VMs. In this Equally Spread Current Execution mechanism, there is a communication between the DataCenterController and the load balancer for keeping the index table updated, leading to overhead. Moreover, this overhead creates a delay to respond to the incoming requests [63].

Throttled Load Balancing (TLB) Algorithm TLB [66] keeps the index table of all the VMs with the maintaining of each VM's state (available or busy). At the algorithm's starting point, all the VM's state is available. The Throttled VM load balancer analyzes the VM allocation table from start to end till the currently accessible VM is identified. The table must be searched entirely.

Honey Bee Foraging (HBF) Load Balancing Algorithm Effective and efficient implementation of load balancing mechanism will construct cloud computing more empirical and jointly enhances user satisfaction. Among the mechanisms, a honey bee forage mechanism [41] is utilized for task allocation and load balancing. When tasks get allocated to the VMs, current load has been calculated. Whenever the VM gets over-loaded, then the VM migration is done among of those VMs whose load is below the threshold value [73]. Honey Bee forage mechanism uses task migration and sub-urbanized load balancing technique in this circle. This mechanism ensures system performance and avoidance of system imbalance.

Ant Colony Optimization (ACO) Load Balancing Algorithm ACO [21] derives from the natural behavior of the real ants. In this load balancing, algorithm [51], the head node is selected randomly that is responsible for generating the ants. The task of these ants is to identify the positions of the under-loaded or over-loaded nodes in the cloud system by traveling the entire cloud network. At the time of traversing, ants update a pheromone table by that each cloud system node can monitor the utilization of resources.

Materials and Methods

The proposed work aims to make an effective and efficient scheduling and uniform workload distribution across the resources of cloud with an excellent performance rate. This paper is based on the classical swarm intelligent technique [52], i.e., a metaheuristic algorithm, honey bee foraging load balancing algorithm [30] along with double threshold-based power-aware load distribution concept to reduce energy consumption. We implemented the scheduling and load balancing algorithms within a cloud scenario to determine which virtual resource is over-loaded or under-loaded.

There exist several data centers and virtual machines. These virtual machines have their ID, CPUs, bandwidth capacity, memory, and processing power. VM's over-loading and under-loading situations are determined by the number of active tasks in the VM. It decides by a threshold value. When the active task is above 90%, then an over-loading situation occurs, and when it is below 20%, then an under-loading situation occurs. Here, these 90% and 20% are called threshold values for deciding over-loading and under-loading situations of VMs, respectively. These threshold values are considered in our approach for experiment as input. It is decentralized to avoid the bottlenecks situation and single point of failure. It uses the following parameters to execute:

- Maximum number of job allocation in VMs.
- Number of currently allocated job in VMs.

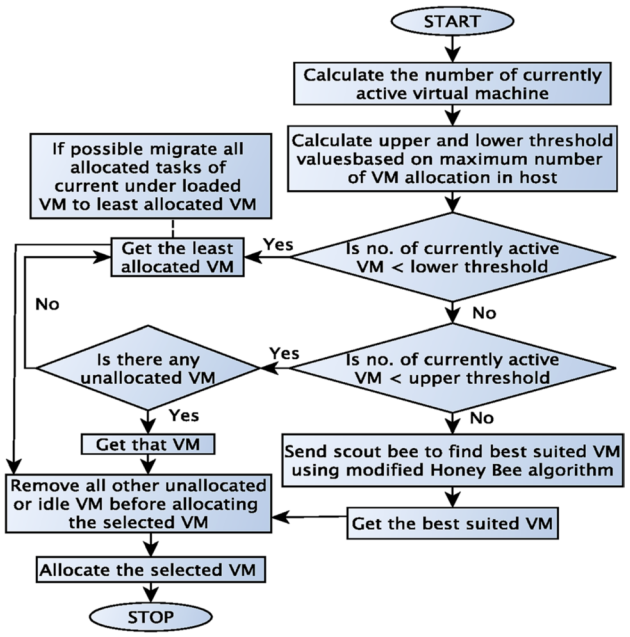

Fig. 1 Flowchart of proposed double threshold-based power-aware honey bee (DTPAHBF) load balancing algorithm

- Number of currently active VMs.
- Virtual machine states list.
- Lower and upper cut-off value of modified honey bee algorithm.

In this work, specific intelligent behavior of a honey bee swarm called foraging behavior is considered. Through the foraging mechanism, bees are continuously searches for the food sources with food quality, quantity, and direction [28]. Also, bees are communicated with each other through the waggle dance. In such way, it reduces the chance of occurring over-loading and under-loading situation of VMs and increases the resource utilization.

A new Double Threshold-based Power-Aware Honey Bee (DTPAHB) Load Balancing algorithm simulates this behavior of real honey bees is discussed for solving optimization problems. The flowchart of the proposed double threshold-based power-aware honey bee (DTPAHBF) load balancing algorithm is shown in Fig. 1 and pseudo-code of the proposed approach is given as a supplementary material. This honey bee-inspired load balancing is based on a dynamic approach on double threshold values depending on the maximum number of virtual machine counts. The upper threshold value is 90% of the maximum count, and the lower threshold value is 20% of the maximum count. Once the tasks are allotted to the VMs, the current load is calculated. If the VM becomes over-loaded, the task is transferred to the VM based on the currently active VM count with respect to the lower and upper threshold. Suppose the currently active VM count is less than the lower threshold value. In that case, the least allocated VM is chosen for task allocation instead of using a modified honey bee algorithm to reduce migration time and cost, storage cost, CPU cost, and memory cost and thus save energy. If the currently active VM count is greater

Fig. 2 Flowchart of the modified double threshold honey bee foraging algorithm

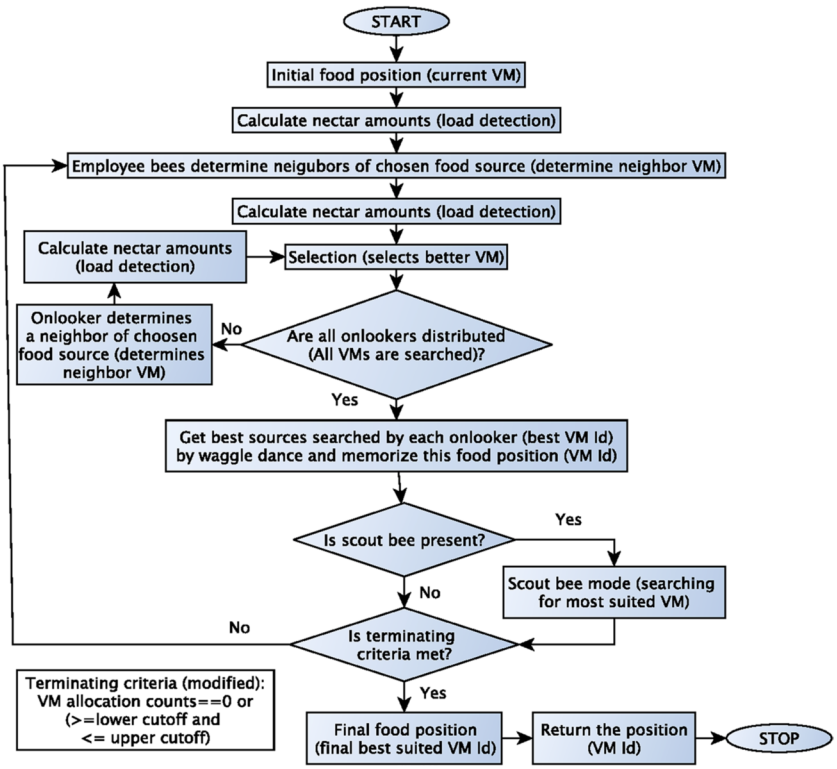

Table 4 Parameter settings for CloudAnalyst simulation

Sl No.	Parameter	Value
1.	VM Memory	512 MB
2.	Data Center OS	Linux
3.	Data Center VM	Xen
4.	Data Center Architecture	x86

than the upper threshold value and an unallocated VM is available, choose that VM; otherwise, choose the least allocated VM instead of using the modified honey bee algorithm to save time, cost, and energy.

When currently active VM count is within the lower and upper threshold value, i.e., within the normal range, then a double threshold-based modified honey bee algorithm is followed to get the optimized VM allocation of the task. The flowchart of the modified double threshold honey bee foraging algorithm is shown in Fig. 2. In the modified honey bee algorithm, if the number of VM allocation is within the normal range, i.e., within 20–90% of the maximum allocation count, then scout bees will not be sent further searching food source; otherwise, it will go for it. After the VM for task allocation is chosen, all unallocated or idle VMs are detected and removed from the host to reduce unnecessary energy consumption. If any host contains VMs that are all unallocated or idle, then the host must be shut down or sent to sleep mode for reducing colossal energy consumption. An idle VM consumes about 70% of energy concerning a fully utilized VM. Honey Bee forage technique hires suburbanized load balancing methodology, and task transfer is blow on the fly.

Simulation and Results Analysis

The experimental setup, results, and discussions of the experiment for this research work are presented in this section. The CloudAnalyst simulator is used for this experiment presented in this section. The results are obtained by comparing the proposed algorithm with five existing load balancing algorithms (Round Robin, ESCE, Throttled, Honey Bee Foraging, and ACO).

Simulation Configuration

We have performed the simulation followed by performance analysis using the specific configuration in the CloudAnalyst toolkit [70]. CloudAnalyst is designed to model the resource scheduling algorithm, cloud service brokers, and cloud data centers. vmLoadBalancer component of the CloudAnalyst is used to implement the load balancing mechanism. This simulator provides an user-friendly GUI to remove all complexities for the programming aspect. It allows parameter sweep to do the experiments by users. This CloudAnalyst framework can allow users to set the regions for cloud-based user bases and data centers. Several other parameters can also be configured like: number of requests generated by per user per hour, number of user bases, number of VMs and number of CPUs, amount of storage and bandwidth of the network, and some other significant parameters, as shown in Table 4.

Based on the parameters mentioned in Table 4, cloud analyst judges the simulation and its results are presented in a graphical format. Following are some statistical metrics based on which we have derived the output of the simulation and compare the performance

1. Average response time of the system.
2. Average processing time of the data center.
3. CPU cost of the virtual machine.
4. Storage cost of the system.
5. Memory cost of the system
6. Total data transfer cost.
7. Energy consumption of the overall process.

The CloudAnalyst enables to repeatedly do the simulation experiments with parameter variations quickly and easily. This is a tool using which testing and simulation are done with different metrics. It is used to examine the behavior of huge applications, present in cloud.

Experimental Setup

The parameters for the configuration of User Bases, Application Deployment, Data center, and Physical Hardware details of any Data Center are defined as given in Tables 5, 6, 7, and 8, respectively.

Table 5 shows that the six distinct regions of the cloud are selected to set up the user bases' locations. The service requests of these user bases are handled by the four data centers. The data centers are located in such way, first one in region 0, second one in region 2, third one in region 1, and fourth one in region 5. The number of allocated VMs is presented in data centers (DC) like, 2 VMs in DC1, 5 VMs in DC2, 10 VMs in DC3, and 4 VMs in DC4. The user bases select the optimized response time-based data center for the application execution purpose under the applied load balancing policy that considers the Optimized Response Time broker policy.

The different no. of virtual machines are defined in the data center. Virtual machines have 512 Mb of RAM memories and 10 Mb bandwidth. Simulated hosts have Xen as virtual machine monitor, Linux operating system, and x86 architecture. The hosts have 100 GB storage, and 2 GB RAM. Others, each machine has the same number CPUs

Table 5 Configuration of user bases used in the experiment

Name	Region	Requests/ User/Hr.	Data Size/ Req. (Bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg. Peak Users	Avg. Off Peak Users
UB1	2	60	100	3	9	1000	100
UB2	0	60	100	3	9	1000	100
UB3	1	60	100	3	9	1000	100
UB4	3	60	100	3	9	1000	100
UB5	4	60	100	3	9	1000	100
UB6	1	60	100	3	9	1000	100
UB7	3	60	100	3	9	1000	100
UB8	5	60	100	3	9	1000	100
UB9	4	60	100	3	9	1000	100
UB10	0	60	100	3	9	1000	100
UB11	1	60	100	3	9	1000	100
UB12	4	60	100	3	9	1000	100
UB13	5	60	100	3	9	1000	100
UB14	2	60	100	3	9	1000	100
UB15	0	60	100	3	9	1000	100
UB16	3	60	100	3	9	1000	100

Table 6 Configuration of application deployment used in experiment

Data center	# VMs	Image size	Memory	BW
DC1	2	10000	512	1000
DC2	5	10000	512	1000
DC3	10	10000	512	1000
DC4	4	10000	512	1000

and speed. The grouping is done in such way, users by a factor of 10, and requests by a factor of 10. 100 instructions corresponding of each user request are executed. CPU takes nearly 45 watts and other units take nearly 28 watts to process each request. The simulation duration took 10 min. Used the response time and processing time metrics and also CPU cost, storage cost, memory cost, data transfer cost, and energy consumption to compare the proposed algorithm with other existing algorithms.

Table 7 Configuration of data center used in the experiment

Name	Region	Arch.	OS	VMM	Cost / VM ($/ Hr)	Memory Cost ($/s)	Storage cost ($/s)	Data Transfer Cost ($/GB)	Physical HW units
DC1	0	x86	Linux	Xen	0.1	0.005	0.01	0.1	2
DC2	2	x86	Linux	Xen	0.1	0.005	0.01	0.1	5
DC3	1	x86	Linux	Xen	0.1	0.005	0.01	0.1	10
DC4	5	x86	Linux	Xen	0.1	0.005	0.01	0.1	4

Table 8 Configuration of physical hardware details of one data center (e.g., DC2) used in the experiment

Id	Memory (MB)	Storage (MB)	Available BW	# Processors	Processor speed	VM policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED
2	204800	100000000	1000000	4	10000	TIME_SHARED
3	204800	100000000	1000000	4	10000	TIME_SHARED
4	204800	100000000	1000000	4	10000	TIME_SHARED

Table 9 Results of response time (RT) and processing time (pt) considering optimized response time service broker policy

Algorithms	Avg. (ms)	Min (ms)	Max (ms)
RR [29]			
RT	148.29	37.60	381.13
PT	0.42	0.01	0.88
ESCE [63]			
RT	148.46	37.60	520.10
PT	0.42	0.01	0.88
TLB [66]			
RT	148.47	37.60	520.10
PT	0.43	0.01	0.86
HBF [56]			
RT	148.41	37.62	367.63
PT	0.47	0.01	3.51
ACO [51]			
RT	148.30	37.60	367.61
PT	0.43	0.01	0.95
DTPAHBF			
RT	148.22	37.48	520.10
PT	0.46	0.01	3.51

Results and Discussion

The obtained result set from our experiments are displayed in Tables 9 and 10. Also, the corresponding graphs are shown in Figs. 3, 4, 5, 6, and 7.

The results shown in Table 9 and graphs presented in Figs. 3 and 4 showed that the proposed algorithm has the same results with respect to overall response and processing time with the other existing algorithms when considering Optimized Response Time Service Broker policy. Also, it can be seen that Round Robin, HBF, and ACO are better than ESCE, TLB, and the proposed algorithm considering the maximum response time. Whereas, Round Robin, ESCE, TLB, and ACO have better maximum processing time than HBF and the proposed algorithm. However, if one considers the other cost measures like CPU cost, data transfer cost, storage cost, and memory cost, then the proposed algorithm is giving better results, as shown in Figs. 5 and 6. Also, as per the energy consumption shows in Fig 7, our proposed algorithm consumes less energy compared to other considered algorithms.

Table 10 CPU cost, storage cost, memory cost, data transfer cost, and energy consumption results considering dynamic service broker policy

Algorithms	CPU cost (/$)	Storage cost (/$)	Memory cost (/$)	Data transfer cost (/$)	Energy consumption (/ Watts)
RR [29]	0.50	181.80	90.90	0.18	4.15926
ESCE [63]	0.53	191.40	95.70	0.18	4.16402
TLB [66]	0.48	174.60	87.30	0.18	4.16447
HBF [56]	0.48	174.60	87.30	0.18	4.16347
ACO [51]	0.50	179.40	89.70	0.18	4.15971
DTPAHBF (proposed)	0.48	173.40	86.70	0.18	4.15798

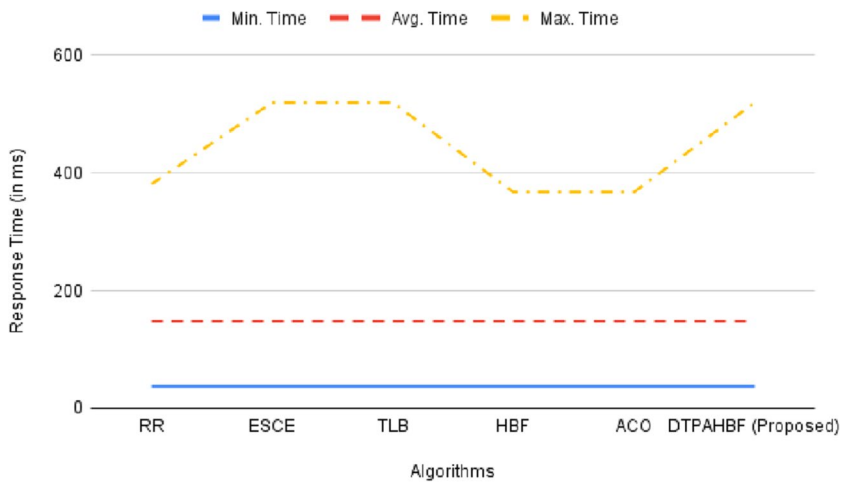

Fig. 3 Performance comparison of minimum, average, and maximum response time

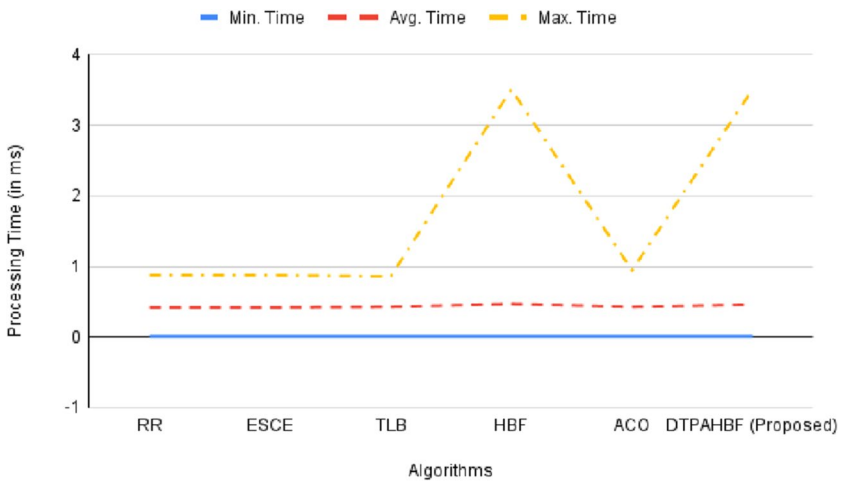

Fig. 4 Performance comparison of minimum, average, and maximum processing time

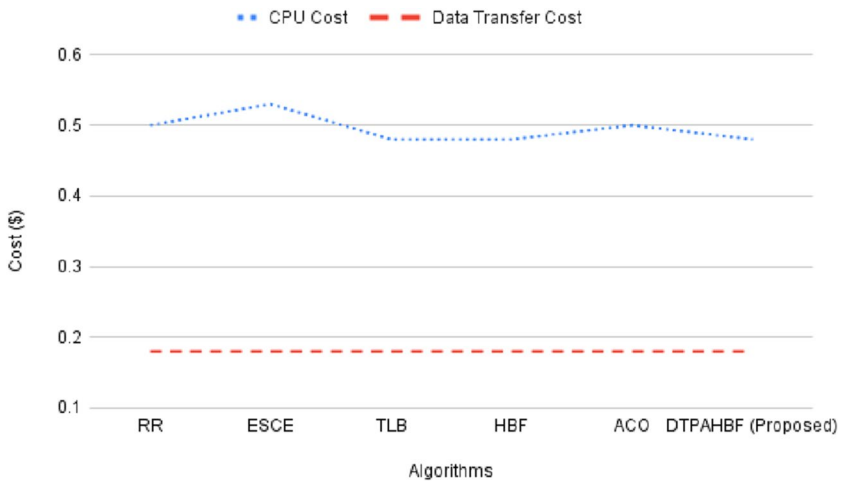

Fig. 5 Comparison of CPU and data transfer cost by each algorithm

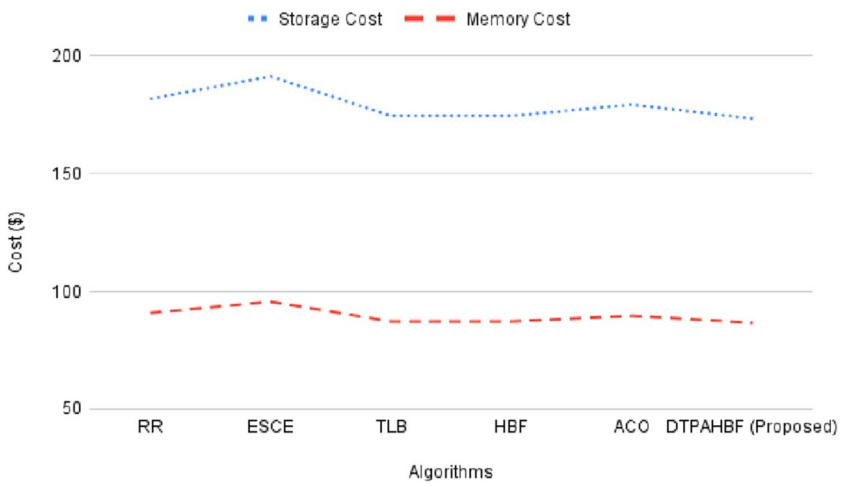

Fig. 6 Comparison of storage and memory cost requirements by each algorithm

Conclusion

Cloud computing becomes an extensively adopted IT service. Although, there exist several challenges, such as load balancing, virtual machines migration, etc. Load balancing is a mechanism for distributing the workload efficiently and effectively. All existing algorithms in the literature focus mainly on overhead reduction, migration time reduction, performance enhancement, etc. The response time is a challenging issue to create the application that would maximize

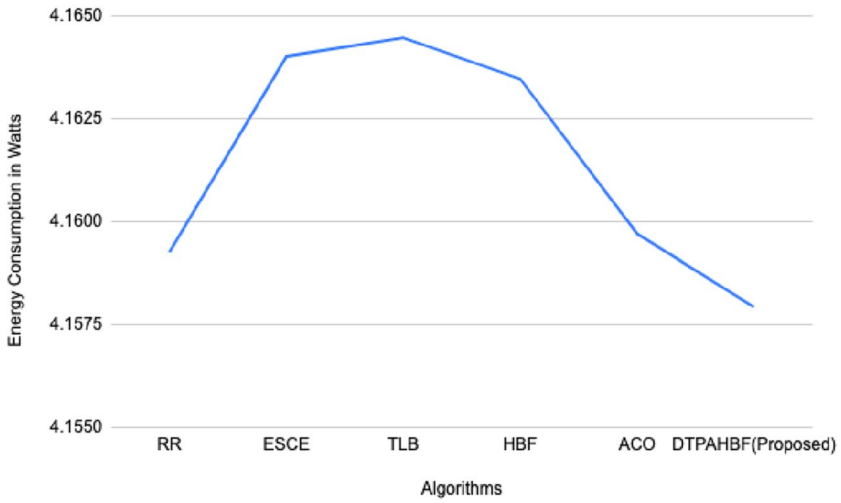

Fig. 7 Comparison between algorithms with respect to energy consumption per request

the system's overall throughput in a cloud environment. The proposed algorithm in this work provides balancing the workload in the cloud environment. Also, the CloudAnalyst simulator is used to reduce the response time for a given number of cloud requests. The generated simulation results show that the proposed algorithm DTPAHBF performs better than the other widely known existing algorithms namely, RR, ESCE, TLB, HBF, and ACO to different aspects.

This proposed double threshold-based power-aware mechanism has notable enhancements comparing the traditional load balancing algorithms with respect to average data center processing time, average overall response time, energy consumption, and total cost. With respect to the data center processing time, DTPAHBF is less efficient compared to RR, ESCE, TLB, and ACO, but presents better result than HBF. In future, our target is to make DTPAHBF efficient in the data center processing task. Our proposed algorithm may initiate unnecessary migration of VMs and this should also be resolved in the future. In the future, considering various load parameters and user requirements, we can also perform this proposed implementation over a real-time cloud setup.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s42979-021-00771-w.

Author Contributions Not applicable.

Funding Not applicable.

Declarations

Conflicts of interest We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. On behalf of all authors, the corresponding author states that there is no conflict of interest.

Availability of data and materials Not applicable.

Code availability Code and experiment if available in YouTube as video material and will be made available for the public after the acceptance of the article, if required.

References

1. Ajit M, Vidya G. VM level load balancing in cloud environment. 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013, pp. 1–5, https://doi.org/10.1109/ICCCNT.2013.6726705.
2. Alakeel AM. A guide to dynamic load balancing in distributed computer systems. Int J Comput Sci Inf Secur. 2010;10(6):153–60.
3. Armbrust M, Fox A, Griffith R, Joseph AD, Randy K, Andy K, Gunho L, David P, Ariel R, Ion S, Matei Z. A view of cloud computing. Commun ACM. 2010;53(4):50–8.
4. Arnold J. Openstack swift: using, administering, and developing for swift object storage. O'Reilly Media Inc; 2014.
5. Bahrami M, Singhal M. The Role of Cloud Computing Architecture in Big Data. In: Pedrycz W., Chen SM. (eds) Information Granularity, Big Data, and Computational Intelligence. Studies in Big Data, vol 8. Cham: Springer; 2015. https://doi.org/10.1007/978-3-319-08254-7_13.
6. Bakde KG, Pati BM. Survey of techniques and challenges for load balancing in public cloud. Int J Tech Res Appl. 2016;4(2):279–90.
7. Bala A, Chana I. Prediction-based proactive load balancing approach through vm migration. Eng Comput. 2016;32(4):581–92.
8. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Exp. 2012;24(13):1397–420.
9. Beloglazov A, Buyya R. Energy Efficient Resource Management in Virtualized Cloud Data Centers. 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 826–831. https://doi.org/10.1109/CCGRID.2010.46.
10. Bhathiya W. Cloudanalyst: a cloudsim-based tool for modelling and analysis of large scale cloud computing environments. Project report, University of Melbourne, 2009.

11. Bilal K, Ur Rehman Malik S, Khan SU, Zomaya AY. Trends and challenges in cloud datacenters. Cloud Comput. 2014;1(1):10–20.
12. Bobroff N, Kochut A, Beaty K. Dynamic Placement of Virtual Machines for Managing SLA Violations. 2007 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007, pp. 119–28. https://doi.org/10.1109/INM.2007.374776.
13. Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp. 2011;41(1):23–50.
14. Calheiros RN, Ranjan R, De Rose CAF, Buyya R. Cloudsim: a novel framework for modeling and simulation of cloud computing infrastructures and services. Preprint arXiv:0903.2525.
15. Choi Y, Bone C, Zhang N. Sustainable policies and strategies in Asia: challenges for green growth. Technol Forecast Soc Change. 2016;112:134–7.
16. Collins E. Big data in the public cloud. IEEE Cloud Comput. 2014;1(2):13–5.
17. Dhinesh BLD, Krishna PV. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl Soft Comput. 2013;13(5):2292–303.
18. Ebadifard F, Babamir SM. Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Cluster Comput. 2021;24:1075–101. https://doi.org/10.1007/s10586-020-03177-0.
19. Elhady GF, Tawfeek MA. A comparative study into swarm intelligence algorithms for dynamic tasks scheduling in cloud computing. 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), 2015, pp. 362–69. https://doi.org/10.1109/IntelCIS.2015.7397246.
20. Farahnakian F, Pahikkala T, Liljeberg P, Plosila J. Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers. In: IEEE/ACM 6th International Conference on utility and cloud computing. IEEE, 2013; p. 256–9.
21. Feller E, Rilling L, Morin C. Energy-aware ant colony based workload placement in clouds. In: Proceedings of the 2011 IEEE/ACM 12th International Conference on grid computing. IEEE Computer Society, 2011; p. 26–33.
22. Fernández V, Méndez V, Pena TF. Federated big data for resource aggregation and load balancing with dirac. Proc Comput Sci. 2015;51:2769–73.
23. Gao Y, Guan H, Qi Z, Hou Y, Liu L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci. 2013;79(8):1230–42.
24. Garala K, Goswami N, Maheta PD. A performance analysis of load balancing algorithms in cloud environment. In: 2015 International Conference on computer communication and informatics (ICCCI), 2015, pp. 1–6.
25. Garg S, Yeo CS, Anandasivam A, Buyya R. Energy-efficient scheduling of hpc applications in cloud computing environments. 2009. Preprint ArXiv arXiv:0909.1146.
26. Garnier S, Gautrais J, Theraulaz G. The biological principles of swarm intelligence. Swarm Intell. 2007;1(1):3–31.
27. Gupta P, Ghrera SP. Load and fault aware honey bee scheduling algorithm for cloud infrastructure. In: Proceedings of International Conference on frontiers of intelligent computing: theory and applications (FICTA), 2015; volume 328, p. 135–43.
28. Gupta T, Handa SS, Panda S. A survey on honey bee foraging behavior and its improvised load balancing technique. Int J Res Appl Sci Eng Technol (IJRASET). 2017;5:2039–49.
29. Hahne EL. Round-robin scheduling for max-min fairness in data networks. IEEE J Sel Areas Commun. 1991;9(7):1024–39.
30. Hashem W, Nashaat H, Rizk R. Honey bee based load balancing in cloud computing. KSII Trans Internet Inf Syst. 2017;11(12):5694–711.
31. Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Appl Math Comput. 2009;214(1):108–32.
32. Khichar GS, Gupta G, Singh R, Rathi R. Maximum correlation with migration control based on modified knapsack (mc_mc) approach for vm selection for green cloud computing. In: 2018 8th International Conference on Cloud computing, data science & engineering (Confluence). IEEE, 2018; p. 1–6.
33. Kiruthiga G, Mary Vennila S. Energy efficient load balancing aware task scheduling in cloud computing using multi-objective chaotic Darwinian chicken swarm optimization. Int J Comput Netw Appl (IJCNA). 2020;7:82–92.
34. Kodli S, Terdal S. Hybrid max-min genetic algorithm for load balancing and task scheduling in cloud environment. Int J Intell Eng Syst. 2021;14(1):63–71.
35. Kulkarni G, Sutar R, Gambhir J. Cloud computing-infrastructure as service-amazon ec2. Int J Eng Res Appl. 2012;2:117–25.
36. Kumar M, Sharma SC. Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing. Proc Comput Sci. 2017;115:322–9.
37. Kusic D, Kephart JO, Hanson JE, Kandasamy N, Jiang G. Power and performance management of virtualized computing environments via look ahead control. In: International Conference on autonomic computing, 2008, p. 3–12.
38. Lee YC, Zomaya AY. Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In: 9th IEEE/ACM International Symposium on cluster computing and the grid. IEEE, 2009; pp. 92–9.
39. Lee YC, Zomaya AY. Energy efficient utilization of resources in cloud computing systems. J Super Comput. 2012;60(2):268–80.
40. Leelipushpam GJP, Sharmila J. Live vm migration techniques in cloud environment—-a survey. In: IEEE Conference on Information & Communication Technologies, IEEE, 2013, pp. 408–13.
41. Li L, Liu F, Li WF, SongKun S, et al. Characterization and mechanism of honeybee foraging behavior. Chin J App Entomol. 2012;49(5):1354–9.
42. Maurya K, Sinha R. Energy conscious dynamic provisioning of virtual machines using adaptive migration thresholds in cloud data center. Int J Comput Sci Mob Comput. 2013;2(3):74–82.
43. Metkar G, Agrawal S, Singh DS. A live migration of virtual machine based on the dynamic threshold at cloud data centres. Int J Adv Res Comput Sci Softw Eng. 2013;3(10):401–5.
44. Mi H, Wang H, Yin G, Zhou Y, Shi D, Yuan L. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: IEEE International Conference on services computing, 2010; p. 514–21.
45. Mishra K, Majhi SK. A binary bird swarm optimization based load balancing algorithm for cloud computing environment. Open Comput Sci. 2021;11(1):146–60.
46. Mishra SK, Sahoo B, Parida PP. Load balancing in cloud computing: a big picture. J King Saud Univ Comput Inf Sci. 2020;32(2):149–58.
47. Mondal AS, Chattopadhyay S, Neogy S, Mukherjee N. Object based schema oriented data storage system for supporting heterogeneous data. In: International Conference on advances in computing, communications and informatics, 2016; p. 1025–32.
48. Mondal SA, Neogy S, Mukherjee N, Chattopadhyay S. Performance analysis of an efficient object-based schema oriented data storage system handling health data. Innov Syst Softw Eng. 2019;16:1–15.
49. Sarkar A, Pant K, Chattopadhyay S. DRSQ - A Dynamic Resource Service Quality Based Load Balancing Algorithm. In: Mandal J., Mukhopadhyay S., Dutta P., Dasgupta K. (eds) Computational Intelligence, Communications, and Business Analytics. CICBA 2018. Communications in Computer and Information Science, vol. 1031. Singapore: Springer; 2019. https://doi.org/10.1007/978-981-13-8581-0_8.

50. Nathuji R, Schwan K. Virtualpower: coordinated power management in virtualized enterprise systems. SIGOPS Oper Syst Rev. 2007;41(6):265–78.
51. Nishant K, et al. Load Balancing of Nodes in Cloud Using Ant Colony Optimization. 2012 UKSim 14th International Conference on Computer Modelling and Simulation, 2012, pp. 3–8. https://doi.org/10.1109/UKSim.2012.11.
52. Panigrahi BK, Shi Y, Lim M-H. Handbook of swarm intelligence: concepts, principles and applications, 1st edn. Springer; 2011. pp. 0-544.
53. Pooja Tandel JS, Parmar Abhijit S. Vm migration using minimum migration time selection policy on virtual machines. J Emerg Technol Innov Res (JETIR). 2019;6:298–301.
54. Pradhan A, Bisoy SK. A novel load balancing technique for cloud computing platform based on pso. J King Saud Univ Comput Inf Sci. 2020.
55. Rastogi D, Bansal A, Hasteer N. Techniques of load balancing in cloud computing: a survey. In: International Conference on computer science and engineering (CSE), 2013.
56. Senthilkumar S, Brindha K, Rathi R, Angulakshmi J, Thirani YV. Honey-bee foraging algorithm for load balancing in cloud computing optimization. Int J Engg Sci Comput. 2017;7(12):2292–303.
57. Sheeja YS, Jayalekshmi S. Cost effective load balancing based on honey bee behaviour in cloud environment. 2014 First International Conference on Computational Systems and Communications (ICCSC), 2014, pp. 214–9. https://doi.org/10.1109/COMPSC.2014.7032650.
58. Shi Y, Qian K. Lbmm: a load balancing based task scheduling algorithm for cloud. In: Advances in information and communication. Springer International Publishing; 2020, p. 706–12.
59. Skourletopoulos G et al. Big Data and Cloud Computing: A Survey of the State-of-the-Art and Research Challenges. In: Mavromoustakis C., Mastorakis G., Dobre C. (eds) Advances in Mobile Cloud Computing and Big Data in the 5G Era. Studies in Big Data, vol 22. Cham: Springer; 2017. https://doi.org/10.1007/978-3-319-45145-9_2.
60. Sran N, Kaur N. Zero proof authentication and efficient load balancing algorithm for dynamic cloud environment. Int J Adv Res Comput Sci Softw Eng. 2013:7;2277–3218.
61. Suresh A, Vijayakarthick P. Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler. 2011 International Conference on Recent Trends in Information Technology (ICRTIT), 2011, pp. 624–7. https://doi.org/10.1109/ICRTIT.2011.5972255.
62. Takeda S, Takemura T. A rank-based vm consolidation method for power saving in data centers. Inf Media Technol. 2010;5(3):994–1002.
63. Tangang, Zhan R, Shibo, Xindi. Comparative Analysis and Simulation of Load Balancing Scheduling Algorithm Based on Cloud Resource. In: Patnaik S., Li X. (eds). Proceedings of International Conference on Computer Science and Information Technology. Advances in Intelligent Systems and Computing, vol 255. New Delhi: Springer; 2014. pp: 449–56. https://doi.org/10.1007/978-81-322-1759-6_52.
64. Teodorović D. Bee Colony Optimization (BCO). In: Lim C.P., Jain L.C., Dehuri S. (eds). Innovations in Swarm Intelligence. Studies in Computational Intelligence, vol 248. Springer, Berlin, Heidelberg, 2009. pp: 39–60. https://doi.org/10.1007/978-3-642-04225-6_3.
65. Tsafrir D, Etsion Y, Feitelson DG. Backfilling using system-generated predictions rather than user run-time estimates. IEEE Trans Parallel Distrib Syst. 2007;18(6):789–803.
66. Tyagi V, Kumar T. Ort broker policy: reduce cost and response time using throttled load balancing algorithm. Proc Comput Sci. 2015;48:217–21.
67. Unhelkar B. Green IT strategies and applications: using environmental intelligence. CRC Press; 2016.
68. VMware. Vmware distributed power management: Concepts and usage. White Paper VMW_10Q1_WP_VSPHERE_DPM_EN_P18_R3, VMware, Inc., 3401 Hillview Avenue Palo Alto CA 94304 USA, 2010.
69. Venkata Krishna PV, Dhinesh Babua LD. Honey bee behaviour inspired load balancing of tasks in cloud computing environments. Elseiver; 2013. p. 120–31.
70. Wickremasinghe B, Buyya R. Cloudanalyst: a cloudsim-based tool for modelling and analysis of large scale cloud computing environments. Distributed Computing Project, CSSE Dept., University of Melbourne, 2009; p. 433–659.
71. Xu J, Fortes JAB. Multi-objective virtual machine placement in virtualized data center environments. In: IEEE/ACM Conference on cyber, physical and social computing (CPSCom), IEEE/ACM, 2010; p. 179–88.
72. Zahariev A. Google app engine. Helsinki University of Technology; 2009. p. 1–5.
73. Zhu L, Li Q, He L. Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm. Int J Comput Sci Issues. 2012;9(5):1694–0814.