Soft Computing
https://doi.org/10.1007/s00500-023-07873-y

OPTIMIZATION

A fruitfly-based optimal resource sharing and load balancing for the better cloud services

B. Edward Gerald[1] · P. Geetha[2] · E. Ramaraj[1]

Accepted: 23 January 2023

Abstract
Cloud is an important smart platform for digital applications to enrich communication services. However, massive data has made sharing resources and scheduling tasks a complex score. Also, any fault in one connected VM has degraded the entire cloud system. Considering these issues, the load balancing objective was taken into consideration. Moreover, the current study has implemented a novel Fruitfly-based transfer learning for sharing the required resources to each Virtual Machine (VM) for the task execution. Initially, the Virtual Machines (VMs) were created with different user tasks then the load was balanced by equally sharing the burden with other connected VMs. Consequently, the task ordering function was performed based on priority, and the required resources were assigned. Moreover, the planned model was tested in the python platform, and the metrics were measured. Also, the improved score for enhancing cloud services is validated by performing a comparative analysis. The shortest duration for the job scheduling process is 180 s, execution time 700 s and response time 15 s. These outcomes are better than the compared conventional models. Hence, this model efficiently balances the data load in the cloud computing environment, improving the cloud services.

Keywords Cloud computing · Virtual machine · Resources sharing · Load balancing · Task scheduling · Makespan time

1 Introduction

In web technology, cloud computing is becoming famous due to its high accessibility (Dalal et al. 2022). It is a well-known technique that provides private and public services to users (Karthiban and Raj 2020). The cloud service includes retrieving information and the program from system files over an internet connection (Reshmi and Saravanan 2020); it also offers storage resources to reserve data online rather than storing it in computer files (Seth et al. 2019). In addition, the cloud services were provided with broad storage areas to save large documents based on the user's needs. Hence, cloud computing facilities were enriched in all fields like healthcare, industry, education and E-business. However, maximizing the cloud servers' performance is a much-needed task for better cloud services.

As the demand for the cloud is enlarging, servers are becoming overloaded with data (Baucas and Spachos 2020). In cloud services, data overload causes high resource loss and has reduced the data sharing rate. Therefore, to satisfy users' needs, a load-balancing technique was introduced (Liu et al. 2019). It is the process of distributing the load among many servers to reduce the processing time taken by each server (Deng et al. 2021). Thus, load balancing provides greater usage of accessible sources to users (Balaji et al. 2021). Cloud computing integrates technologies like virtualization, network computing, etc., and improves user services (Therese et al. 2021). Virtualization defines the enveloping of physical computer resources while dealing with applications to minimize the complexity of users (Haseeb-Ur-Rehman et al. 2021). Network computing is a technique that permits

✉ B. Edward Gerald
geraldmay18@gmail.com

P. Geetha
geeth.ganesan@gmail.com

E. Ramaraj
ramrajrediff@gmail.com

[1] Department of Computer Science, Alagappa University, Karaikudi, Tamil Nadu 630003, India

[2] PG Department of Computer Science, Dr. Umayal Ramanathan College for Women, Karaikudi, Tamil Nadu 630003, India

Published online: 07 February 2023

users to access system files by distributed computing over the internet (Sabireen and Neelanarayanan 2021).

The load balancing in cloud computing is shown in Fig. 1. In the beginning stage, cloud computing is used in online-related business administration. Still, later it is widely used by different service companies to provide shared services among many users (Bello et al. 2021). All cloud units control the cloud environment (de Carvalho et al. 2021). Moreover, the steady connection between users and servers is verified by cloud carriers in the environment (Karthick et al. 2021). In addition, in the case of public and private clouds, the data centre is inside the cloud service providers and network administration, respectively (Helali and Omri 2021). Generally, a cloud environment has two elements: frontend and backend (Ansari et al. 2021). The front end is for users to use the information over the internet. At the same time, the backend deals with cloud service (Saldamli et al. 2021). As cloud-based applications are becoming popular and used worldwide by huge populations, they face several issues because of poor load balancing, such as security and resource management (Ranapana and Jayasena 2021). However, these cloud services were implemented in different applications like energy management in the cloud by wind-based firefly model (Swarna Priya et al. 2020), diabetes prediction framework (Rajput et al. 2022), etc.

So, load balancing is considered in this present work. If the loads were balanced properly, then the cloud computing process was enriched by optimal task scheduling. Also, the proper load balancing system has reduced the makespan and task execution delay (Kodli and Terda 2021). Hence, cloud computing is the required paradigm, but it is vulnerable to threats. So, threat mitigation is the Task needed for the cloud computing model. This blockchain model was implemented as the threat mitigation model (Alouffi et al. 2021). In addition, to execute the assigned job, the VM system was introduced in the cloud computing strategy

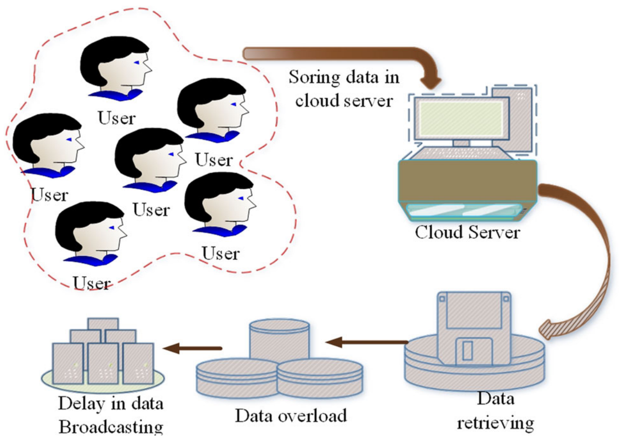

Fig. 1 Cloud computing framework

(Gabhane et al. 2021). The tasks were scheduled and implemented based on the VM load balancing capacity. In addition, better resource-sharing models were required to minimize the makespan duration of the cloud computing strategy, so a deep network was introduced in the cloud computing environment for a better resource-sharing process (Siddesha et al. 2022). The emotional features were upgraded to the cloud memory to perform the cloud computing task in emotional intelligence (Kouatli 2018). Besides, benchmarking techniques were utilized to process more data in the cloud paradigm (Kouatli 2019). However, a high makespan time was recorded because of the high data load. Different load balancing techniques, such as load balancing technique along with autonomous task allocation (Ebadifard and Babamir 2021), load balancing model based on metaheuristic optimization (Ziyath and Senthilkumar 2021), Trust-based agent (Li et al. 2019), Trust-based clustering (Zanbouri and Jafari Navimipour 2020), dynamic cloud resource allocation (Mireslami et al. 2019), vehicular cloud resource allocation using multi-objective model (Wei et al. 2021), and metaheuristic based task scheduling (Houssein et al. 2021) were introduced recently to improve the cloud services. However, the finest outstanding was not gained because of high data overload. Also, the highly overloaded data can damage the VM's performance if the process is continued. Hence, they faced some issues like high cost, complexity, etc. Thus, a novel, the cost-efficient load-balancing technique was developed to provide better cloud services.

So the present work has introduced the optimized transfer learning for this cloud application for tuning the performance of cloud services by balancing the data load. Here, the best solution of optimization is incorporated into the transfer learning to offer outstanding results. Hence, the novel technique is named novel FbTL. In addition, the load was balanced by the optimal task scheduling and resource-sharing process.

The key contribution of this present study is exposed as follows,

- Initially, the required number of VMs is designed with different user tasks in the python environment.
- Consequently, a novel FbTL has been developed with the required load balancing and resource-sharing parameters.
- Moreover, the data-overloaded VMs are predicted by the fitness of fruitfly and half of the data is shared with other VMs.
- Then the tasks are scheduled on a priority basis, and the required resources are shared.
- Finally, flexibility and scalability have been measured based on resource usage, execution time makes pan time, and scheduling time.

The present research study is structured as follows; recent associated works are described in Sect. 2, the usual cloud service system with the problem is described in Sect. 3, solutions for the issues discussed are explained in Sect. 4, and Sect. 5 has described the outcome of the proposed solution. Finally, the research arguments are concluded in Sect. 6.

2 Related works

Some of the recent works related to loading balancing in cloud computing are described below:

The problem with the dynamic load balancing technique is that it enlarges inter-VM transmission overheads. Thus, to control transmission overheads, Ebadifard and Babamir (2021) developed a load-balancing method and an autonomous task allocation system in cloud computing. It shares the workload evenly and distributes the resources to users. The task allocation reduces the network traffic and enhances the performance of applications. However, this method concentrated only on Central Processing Unit (CPU) requests alone.

In the beginning stage, the task allocation strategy is used to reduce workload and minimize the traffic in the application. But, in a cloud environment, task allocation has several issues. Hence, Ziyath and Senthilkumar (2021) presented a novel load-balancing model based on a metaheuristic optimization method to improve cloud services and drain allocation problems in cloud applications. The complexity of this model is low and provides better services. But, when workloads are increased in the network, it takes more time.

In cloud architecture, to distribute a file virtually, cloud computing utilizes load balancing design and allocation design. Hence, an optimal task scheduling model and load balancing are necessary to disseminate information/files optimally. Thus, Priya et al. (2019) presented a multidimensional resource allocation design based on fuzzy logic to achieve better Task and resource scheduling efficiency. This hybrid approach incorporates the merits of the allocation model and load-balancing design. However, the execution time is more than the neural networks.

In cloud architecture, balancing the workloads between VMs plays a significant role. As the tasks are scheduled between VM owning different execution times, different sizes, etc., an optimal technique is required to obtain optimal balancing among VMs. Hence, Pradhan and Bisoy (2020) presented an optimal load-balancing method using an improved PSO task allocation model version. This technique enhanced the accessibility of resources and reduced execution time. However, the implementation cost is high in this technique.

Due to the wide usage of cloud servers, several issues arise in task and resource allocations and load balancing. Thus, optimal control over resources is required to provide efficient scheduling of resources to users in networks. Hence, Pourghaffari et al. (2019) developed an integral technique that reduces the computational time in resource scheduling. This method incorporates fuzzy logic, task allocation models, and different algorithm. This model minimizes energy consumption and computational time in networks. However, it does not provide efficient resource management in application.

The Markov model was implemented by Sefati and Navimipour (2021) for the cloud application to validate the service computation's resources. Here, the validation process was tuned by the optimization approach. Moreover, the application platform considered for this performance validation process is the Internet of Things. Hence, the Quality of Service (QoS) is analyzed for each functional level. But, it has required more computational time. In addition, Tong et al. (2021) have presented the task selection strategy in the cloud computing field for managing the data load during the data transmission process. The model that was utilized for the task selection paradigm is reinforcement learning. The accurate task selection outcome was gained based on priority. However, it takes more resources for task selection. Besides, Asghari et al. (2021) have executed the genetic model for job scheduling by the genetic fitness solution. In addition, for training the system, intelligent models were utilized. Here, the tasks were scheduled based on the deadline priority. But, it has needed additional space to implement the process.

The resource allocation model called the First-fit algorithm was introduced for the cloud computing application by Fathalla et al. (2022). The main objective of this work is to measure the target range and allocate the required resources for the particular VMs. After analysis, the required resources were given to the VMs for executing the assigned tasks. This model is not suitable for the dynamic cloud computing environment. To offer uninterrupted cloud services, service selection is a major concern. Based on the service selection, the cloud task execution function became easier, which minimized the algorithm's complexity score. So, Liu et al. (2022) have designed Kuhn–Munkres model for the service selection process in the cloud computing paradigm. Hence, the service was selected more optimally. However, this model is not suitable for resource allocation.

The overview of the discussed literature is exposed in Table 1. Considering these problems in the conventional cloud computing model, the present work has implemented an efficient strategy with the principle of transfer learning and fruitfly optimization. In addition, the cloud computing

Table 1 Overview of Discussed literature

Authors and references	Methods	Merits	Demerits
Ebadifard and Babamir (2021)	The distributed load-balancing approach	It has reduced the data traffic during the data transferring process	But, the manual request is not supported in this distributed framework
Ziyath and Senthilkumar (2021)	Metaheuristic load balancing	Resource allocation has been improved that has tended to increase the cloud services	It has recorded more time
Priya et al. (2019)	Fuzzy-based multidimensional resources allocation	This model is suitable for balancing the load in cloud computing	It has required more time
Pradhan and Bisoy (2020)	the optimal load-balancing method	Particle swarm fitness is utilized for allocating resources. So, the resource allocation process was executed exactly	It runs several iterations. So, it has required more memory resources
Pourghaffari et al. (2019)	integral technique	Resource scheduling is utilized in a sequence pattern, which mitigates the makespan time	However, this model has provided the same resources for every job. So, high resource wastage was reported
Sefati and Navimipour (2021)	Markov model	The cloud service performance was computed at every level	High time consumption was reported because of the data traffic
Tong et al. (2021)	reinforcement learning	It has offered accurate outcomes in the task-scheduling process	It has needed additional resources to complete the process
Asghari et al. (2021)	Genetic algorithm	Tasks were scheduled optimally by genetic fitness. Also, to train the tasks, neural networks were utilized	It needs more memory to execute the process
Fathalla et al. (2022)	First-fit algorithm	This model is implemented for the proper resource allocation based on task priority. Hence, the data load was balanced	It is not suitable for the dynamic cloud services
Liu et al. (2022)	Kuhn–Munkres	This model is good in the cloud service selection process	However, it is not suitable for the resource allocation process. Hence, the job execution percentage became very low

framework often causes issues because of data overloading. So, the load-balancing objective for this present research work is to provide better cloud services. Here, the load was balanced by the finest resource allocation strategy. Here, the resource-sharing features were updated in the fruitfly memory. Initially, the required resources of each Task were monitored and updated in the fruitfly memory then the Task was scheduled based on priority. Hereafter, the needed resources that the fruitfly estimates were shared with each VM. Besides, the fruitfly function was incorporated in the classification layer of the transfer learning, which provides optimal performance at each run. This process has been executed repeatedly, which helps to avoid data overload in the cloud computing paradigm.

3 System model with problem

Due to the vast wide amount of data managing the cloud services are much more difficult. Moreover, the data in the cloud system are not balanced, which has tended to cause VM damage. During the cloud processing module, if one VM is damaged, it collapses all functions between the cloud users. Considering these issues, the optimization model has implemented several load-balancing models. But, those have resulted in high execution time and resource usage. Hence, the present study has attempted to design a novel optimized load-balancing model based on the intelligent approach.

The conventional cloud computing paradigm has the following issues: poor scheduling, poor resource sharing, and data overloading in the connected VM's, described in Fig. 2. These issues were addressed in this present study by implementing the optimized deep networks.

4 Proposed methodology

In cloud computing, arranging the user's request and balancing the data load is the most requested Task in the cloud environment. Hence, the present study has introduced a novel Fruitfly-based transfer learning (FbTL) for sharing

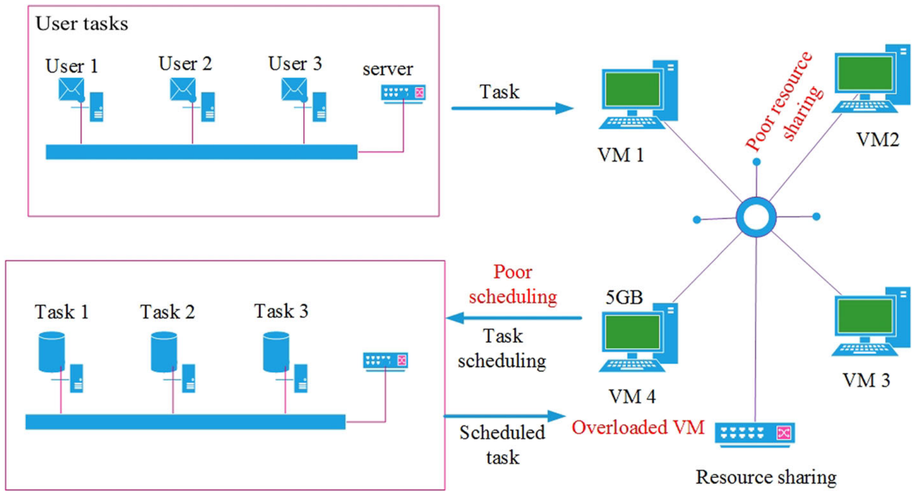

Fig. 2 Issues in cloud services

the required resources to the cloud users based on priority. Initially, the cloud environment was generated with four Virtual machines (VM). Each VM has specific tasks for different users. If any VM task has been overloaded, it is predicted by ant fruitfly fitness, then half load is shared to other VM, and then the functions in each VM have been scheduled priority-wise. Finally, the resources are allocated, and the task execution parameters have been noted.

The proposed architecture is described in Fig. 3. Total jobs considered in this study are 1200, the number of VM processors is 4, RAM 8 GB, and the operating system is windows 10. Subsequently, the percentage of improvement score is measured by the comparative analysis. Moreover, the tasks in Fig. 3 are data broadcasting, application download, E-business, surveying the products reviews, mailing and accounting. After training the task data and designing the required VMs. The proposed model was developed then further processes like load balancing, task scheduling, resource estimation and sharing process were executed.

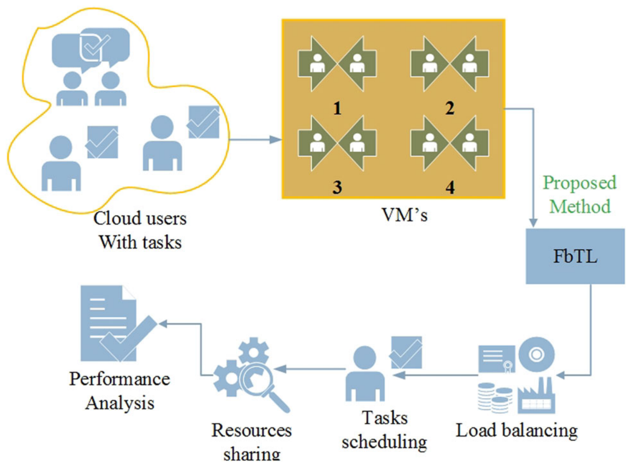

Fig. 3 Proposed architecture

In addition, the conventional data load monitoring model has required more time and resources for the data load monitoring process. Using the fruitfly fitness, these complexities were reduced by fixing the optimal load balancing capabilities of the VM. During the data overload monitoring process, this module was activated to find the data overloaded VM, which gives the optimal best solution as an accurate load monitoring process.

4.1 Process of FbTL

Primarily, the required number of VM's was created then a novel FbTL was designed. The working principle of the designed model is based on transfer learning (Yu et al. 2019) and fruitfly optimization (Hu et al. 2021).

$$F(T) = T\{1,2,3,4\} \tag{1}$$

The VM designing process is defined in Eq. (1). Here, T represents the user tasks, and the VM training or designing function is determined as $F(T)$ and {1, 2, 3, 4} determines the 4 number of VM's. The steps in the presented design are described in Fig. 4.

The stepwise process of the designed model is described in Fig. 4. Here, four phases were considered: initialization layer, in which VM and tasks were generated. The second layer is the feature analysis phase; task size and priority have been measured here. Once the feature is analyzed, then the function of the tuning layer has been activated for finding the free VM's and to forward the half load to the free VM's. Furthermore, the parameters were calculated in the final output phase.

4.1.1 Load monitoring phase

Before initiating the resource sharing and task allocation process, the designed virtual system has to be optimized.

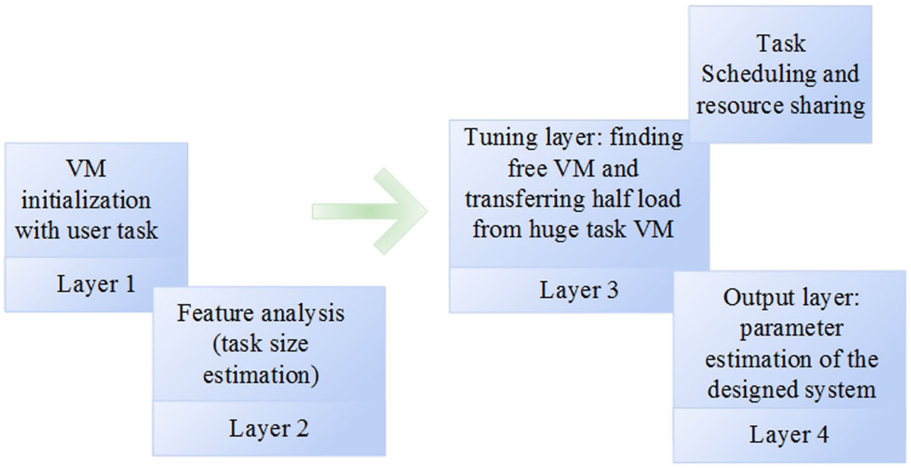
Task Scheduling and resource sharing
Tuning layer: finding free VM and transferring half load from huge task VM
VM initialization with user task
Layer 1
Feature analysis (task size estimation)
Layer 2
Layer 3
Output layer: parameter estimation of the designed system
Layer 4

Fig. 4 Steps of the proposed FbTL

Hence, the load monitoring and balancing module have been executed. Here, the loads that are present in each VM have been predicted by fixing the maximum optimal load balancing range in the fitness of fruitfly. Fixing load balancing capacity in VM is described in Eq. (2).

$$\text{Load}(\text{VM}) = |F(S) - 2[(S)]| \tag{2}$$

Here, the task loading is determined as $F(S)$, the Task is denoted as S, and 2 represents the size of Task in the VM, i.e., 2 GB. Fixing the load migration condition is defined in Eqn. (3), and the migration process is executed by Eq. (4).

$$M_c = F(S) < 2\ \text{GB} \tag{3}$$

Moreover, the load migration parameter is described as M, if the condition M_c is satisfied, then the load balancing process has been executed by migrating the loads. Here, M_c is the migration condition checking parameter.

$$M = \frac{1}{2}(S) \rightarrow T < 2\ \text{MB} \tag{4}$$

Once the migration condition is satisfied, half of the load is shared with other free hubs. The VM's tasks that are less than 2 MB then the VM are considered free VM. Moreover, half of the load from the overloaded VM is shared with the particular VM. This is defined the optimized status of VM.

4.1.2 Task scheduling and resource sharing

To avoid the data overloading and VM damage, task scheduling is an important module in the cloud computing and job execution environment. In cloud computing and the task execution environment, the jobs have been scheduled in different ways that are based on deadline, based on the priority of user login, or task important priority range. Here, the present study has scheduled the Task based on the user's login priority. The tasks were scheduled for each VM based on the max load balancing rate. After scheduling the tasks in the VM's, the required resources for each Task have been shared with the particular VM's.

$$J = \text{schedule}(1 - P[(S)]) \tag{5}$$

The task ordering function is described in Eq. (5). Here, the scheduling parameter is determined is j and the task scheduling process based on priority is denoted as 1. Moreover, the priority estimation of each Task is described as $P[F(S)]$. Here, p denotes priority.

$$R_S = D_r\sqrt{S_1 + S_2 + \cdots S_n} \tag{6}$$

Here, the resource sharing parameter is described as R_S, the data rate is represented as D_r, and $S_1 + S_2 + \cdots S_n$ defined as the n number of scheduled tasks. Hence, the resource sharing process is explained in Eq. (6).

Algorithm: 1 FbTL

```
Start
{
  int $t = 1,2,3,4$;
  // Initializing virtual machines
  Load monitoring ()
  {
      int $S$;
      // Task initialization
      $Monitoring(VM) \to VM(load) > 2GB$
      // if the VM's load is greater than 2GB, then it is considered an overloaded VM
  }
  Load Balancing ()
  {
      int $M$;
      //initialize load balancing variable
      $if(F(s) > 2GB)$
      {
          $migrate(S) = 1/2(S) \to free \quad VM$
      }
      $free \quad VM = load > 2MB$
      // updating the condition of the free VM search
  }
  Task scheduling ()
  {
      int $P, J$;
      // initializing job scheduling parameter
      $schedule \quad job \to P(S)$
      // tasks have been scheduled based on priority
  }
  Resource sharing ()
  {
      int $R_S, D_r$;
      // initialize resource sharing variables
      $D_r \to P(S)$
      // resources were shared
  }
Performance parameter validation
Stop
```

The procedures considered for balancing the load in the cloud environment are described stepwise as defined in Fig. 5. Here, the task scheduling function was processed based on the task priority, which is predicted by the fruitfly function. Then the free VM selection is found by estimating the data load percentage of every VM; the condition of a free VM's selection is data should be less than 2 MB, and then it is considered a free VM. This condition was also updated in the fruitfly memory; during execution, the free VM was selected by enabling that free VM selection condition.

The pseudo-code of the developed algorithm is described in algorithm 1. The modules included in the present model are load monitoring, balancing, task scheduling, and resource sharing. Here, these modules were executed to enrich the cloud server performance.

5 Results and discussion

The planned model is tested in the python framework executed in the windows 10 platform. Primarily, the required numbers of VMs were created with different user tasks, then to bring the optimal status, each VM's load. Also, the overloaded VM is identified by fixing the max optimal load range.

The chief testbed parameters that were required for this execution process are RAM, bandwidth, VM's power operating system, memory and storage capacity. Those parameters are defined in Table 2.

5.1 Case study

This case study has been conducted to measure the working function of the designed model. The key objective addressed in this study is load balancing and optimizing the cloud servers by enhancing the cloud data transmission and Task executing services. After the feature analysis process of task size, the recorded task size is 2.9 GB, i.e., $F(S) = 4$. When substituting this in Eq. (2),

$$M_c = 2.9\text{ GB} > 2\text{ GB}$$

Here, 2.9 GB is greater than the 2 GB, hence the migration condition is satisfied, and load sharing has been initiated. Moreover, half of the load is shared with another free VM; the load-transferring process has been executed using Eq. (3). Here, the calculated half of the load from the previous calculation is 1.45, i.e., half of 2.9. Hence, 1.45 is substituted in Eq. (3).

$$M = 1.45 \to T < 2\text{ MB}$$

Now the half load is migrated to another VM, which has the low data loading rate that is described as $T < 2$ MB. Consequently, the Task is scheduled based on the user request or login priority in the cloud server. Finally, the required resources were shared with the scheduled tasks. The time taken to schedule the Task is defined in Fig. 6d), and the response time is described in Fig. 6f).

The metrics throughput has been measured to measure the data transfer speed and delivery capacity. Hence, the recorded throughput measure is 5000 bps, which is quite good for the cloud environment to share the data. Here, the jobs count has varied between 200 and 1200, illustrated in Fig. 6a). The presented model has recorded the maximum resource usage rate as 4.6% for a maximum delay limit of 0.6. Also, the gained minimum resource utilization score is

Fig. 5 Flow of the design FbTL

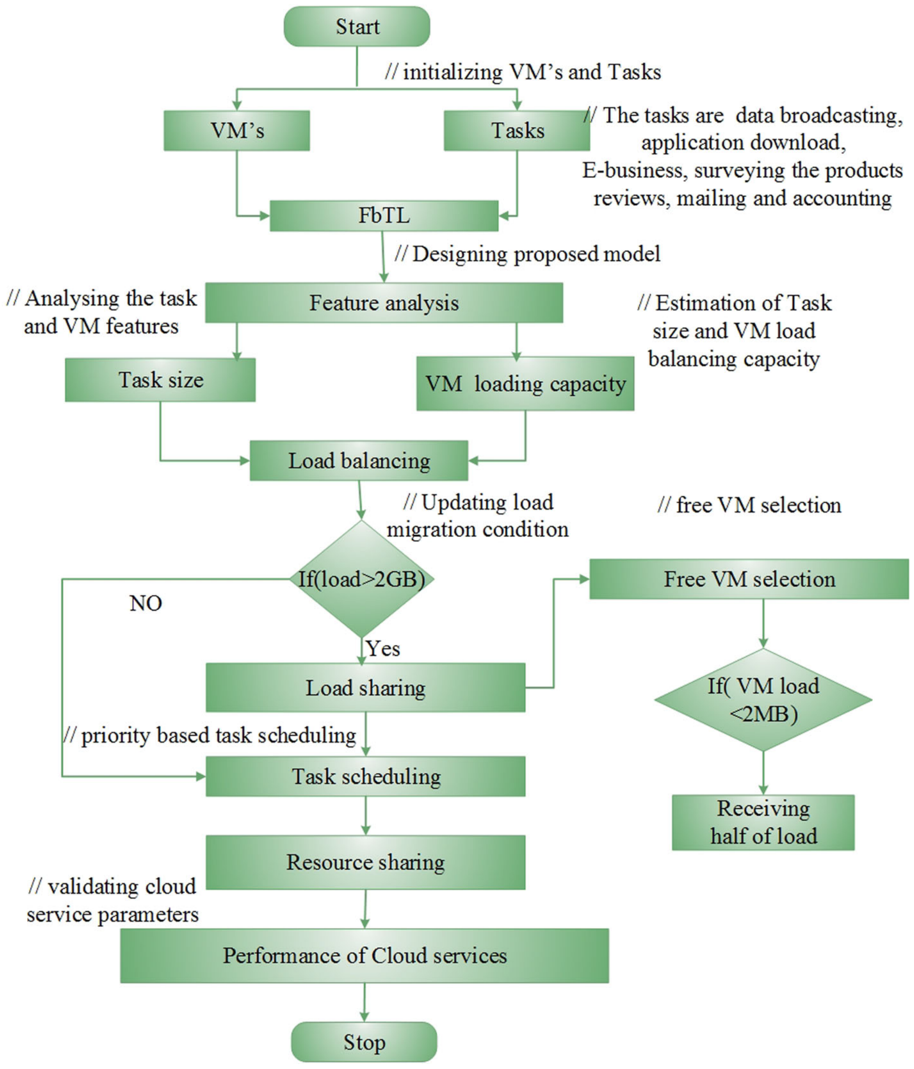
Start
// initializing VM's and Tasks
VM's
Tasks
// The tasks are data broadcasting, application download, E-business, surveying the products reviews, mailing and accounting
FbTL
// Designing proposed model
// Analysing the task and VM features
Feature analysis
// Estimation of Task size and VM load balancing capacity
Task size
VM loading capacity
Load balancing
// Updating load migration condition
// free VM selection
If(load>2GB)
NO
Free VM selection
Yes
Load sharing
If(VM load <2MB)
// priority based task scheduling
Task scheduling
Receiving half of load
Resource sharing
// validating cloud service parameters
Performance of Cloud services
Stop

Table 2 Tested parameters

VM parameters	Description
RAM (GB)	3.75
Bandwidth	1Gbps
VM's power (MIPS)	250 to 1400
Operating system	Windows 10
Architecture	X64
Host parameters	6821 MIPS
CPU	2 × Intel Xeon E5-2630 2.3 GHz
Memory (DDR3)	128G
Storage capacity	4.81 TB

2.3% for the delay limit of 0.1. These statistics are described in Fig. 6b).

The starting and ending duration of works have been measured concerning the delay limit. The delay limit range considered for measuring the work completion process is 0.1–0.6. These statistics are exposed in Fig. 6e).

The execution period of the presented model is measured based on the distributed functions. The distribution range taken to measure the execution duration is 0.3–0.85. Moreover, the recorded high execution period of the developed framework is 700 s. These details are exposed in Fig. 6c).

5.2 Performance validation

To measure the performance maximizing score of the designed model, some of the existing works were considered that is Maximum Reduction Model (MRM) (Malik et al. 2022), Clustering and Checkpoint replication (CCR) (Malik et al. 2022), heterogeneous Earliest Completion time (HECT) (Malik et al. 2022) and Hybrid Wolf-Ant lion Model (HWALM) (Malik et al. 2022).

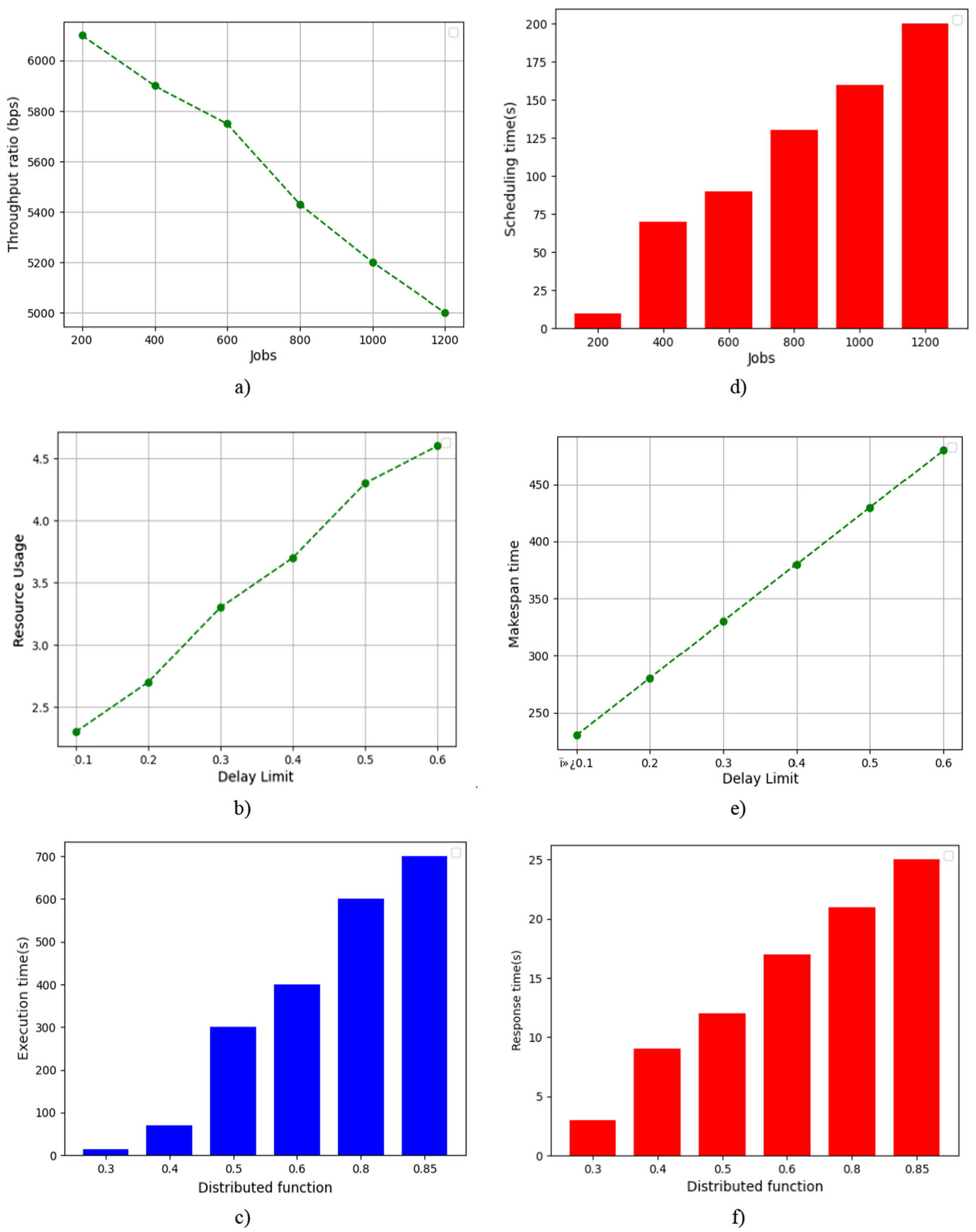

Fig. 6 Performance Outcome: **a** Throughput ratio, **b** Resource usage, **c** Execution time, **d** Scheduling time, **e** makespan time, **f** Response time

5.2.1 Resource usage

Usually, the cloud environment is rich in vast data content, so more resources are required to execute the specific process in the cloud application. Hence, measuring the resource utilization rate is more important. Here, the resources have included memory features, execution time, and disk occupying space.

The existing model MRM has earned the maximum resource usage score is 10%, the CCR approach has yielded the highest percentage of resource utilization is 10.1%, the HECT model has employed 9% resource for performing the task scheduling process, and the model HWALM has earned the resource utilization score as 5%. Compared to these previous studies, the designed model has earned a resource usage score of 4.6%. The statistics of resource utilization comparison are determined in Fig. 7.

5.2.2 Makespan duration

Validation of work assigning time and the completion time is considered as Makespan time. Hence, the Makespan of each Task and VM might be differed based on the target of a specific task and delay. This delay is often caused because of the occurrence of data overloading in the connected VM's. Hence, this present study has addressed this issue by implementing the load migration strategy.

The recorded work span duration for the model MRM is 1000, the time taken by CCR framework for initiating and ending the Task is 1010 s, the model HECT has reported the makespan duration as 900 s, and the model HWALM has reported the makespan duration as 500 s. Moreover, the designed model has recorded the makespan duration as 480 s. This comparison measure is defined in Fig. 8.

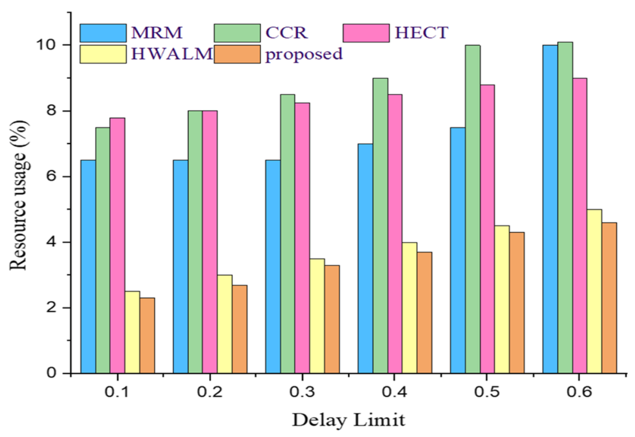

Fig. 7 Comparison of resource usage

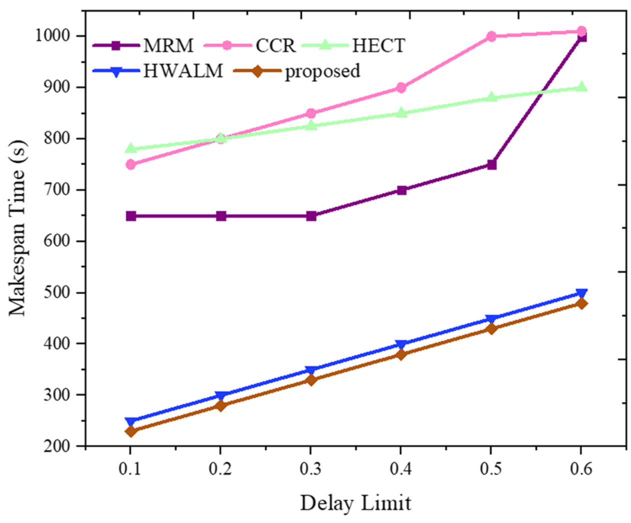

Fig. 8 Makespan time comparison

5.2.3 Execution time

Task execution time has been varied based on the task target. Considering this, the Task is calculated for the distributed functions. The cloud environment has included the distributed users at several ends; hence, the execution time is measured for the distribution function and the comparison is described in Fig. 9.

While performance analyzing, the execution time earned for the approach MRM is 800 s, CCR model has yielded 600 s of execution duration, the scheme HECT has recorded the execution period as 400 s, HWALM model reported the execution duration as 200 s. Considering all these existing studies, the designed model has recorded the execution duration as 180 s, which is very less than the previous studies.

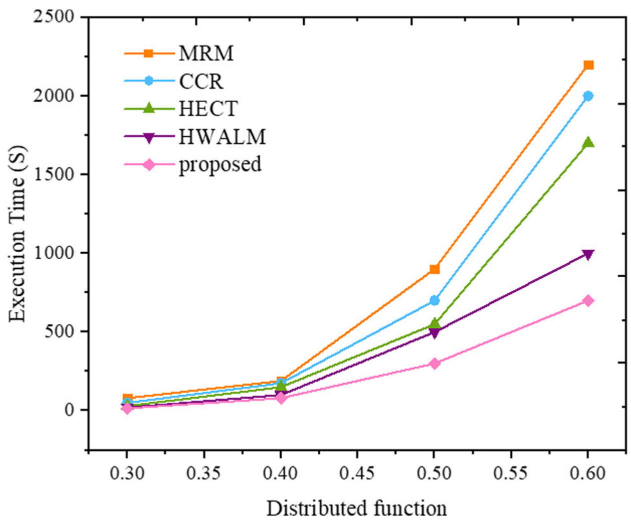

Fig. 9 Execution time comparison

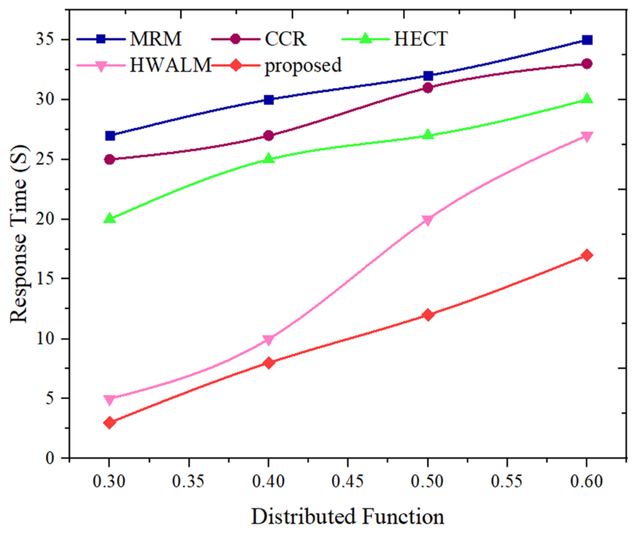

Fig. 10 Response time comparison

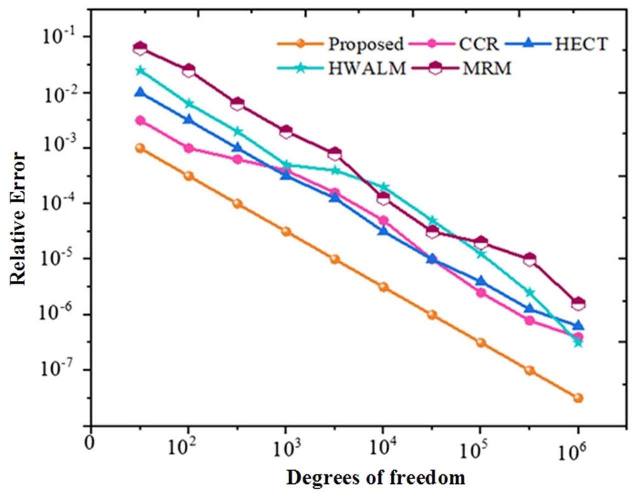

Fig. 11 Relative error comparison

Table 3 Scheduling time validation

Time complexity assessment	
Methods	Time complexity (s)
MRM	800
CCR	600
HECT	400
HWALM	200
Proposed	180

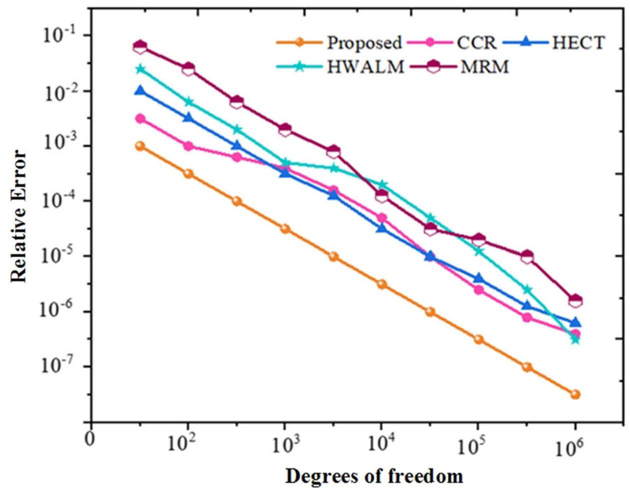

Fig. 12 Comparison of RMSE

5.2.4 Response time

The metrics response duration is measured to justify the performance of the proposed model. During the task process, if there are no issues based on overloading, then the acceptable time is very short. If any interruption occurs between the task scheduling and resource sharing, the response time is too long.

Here, when the distribution function is increased, the response time has been maximized because of the multiple users in the cloud environment. Response duration achieved for the model MRM is 35 s, CCR 33 s, HECT 30 s, and the hybrid model HWALM has recorded the task assignment response duration as 27 s. Besides, the presented novel scheme FbTL has recorded the maximum response time as 25 s.

However, this increased response time is in the average state only. It cannot affect the cloud servers more. The assessment of response duration comparison is determined in Fig. 10.

5.3 Discussion

The performance assessment has shown the finest performance of the designed model. In addition, the scheduling time is measured and compared with other models to measure the task scheduling capacity. Those statistical outcomes are determined in Table 3.

The model MRM has yielded the task scheduling duration as 800 s, the approach CCR has reported the scheduling duration as 600 s, the HECT framework has reported the scheduling duration as 400 s, and the model HWALM has gained the scheduling duration as 200 s. Considering these models, the developed framework has reported the scheduling duration as 180 s. Besides, the delay limit is set based on the memory and disk space resources. The Makespan duration was optimized if the memory spaces were not overloaded.

To measure the proposed model's fall ratio, the relative error and the Root Mean Square Error (RMSE) were validated. Besides, to verify the system's robustness, recent existing approaches were tested in the same proposed platform, and the performance was measured. Hence, the comparison is described in the convergence plot elaborated in Figs. 11 and 12.

Here, the relative error is measured concerning degrees of freedom and the RMSE was measured based on the total count of file samples representing the tasks.

These outcomes have verified the necessity of the designed model in cloud application services. In addition, the traffic-free cloud environment will help the network users to easily access the data at any time during any data retrieving applications.

6 Conclusion

A novel solution is presented for improving the cloud services that are FbTL. Hence, to enrich the resource-sharing process, the load-balancing model has been conducted with the help of fruitfly fitness. Here, the load has been balanced by the equal separation of the assigned tasks. Moreover, the task scheduling function has been executed based on priority, and the required resources have been shared. Furthermore, the presented model has reported the scheduling duration as 180 s compared to the existing studies; 20% of the scheduling duration has been reduced. Also, 15 s has been recorded as the maximum response time of the designed model, compared to other conventional approaches; it has minimized the response duration by 10%. In addition, a novel FbTL model has reported the maximum execution duration as 700 s; compared to other models, 30% of the time has been reduced. Hence, the designed resource-sharing scheme is more efficient for improving cloud services by reducing the computation cost. However, if the job counts are increased, then the execution time has been maximized because of insufficient memory. In future, designing the transfer learning with this proposed model will give better outcomes by allowing limited tasks for the execution at a single cycle.

Acknowledgements None.

Funding The authors have not disclosed any funding.

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare that they have no potential conflict of interest.

Ethical approval All applicable institutional and/or national guidelines for the care and use of animals were followed.

Informed consent For this type of analysis formal consent is not needed.

References

Alouffi B, Hasnain M, Alharbi A, Alosaimi W et al (2021) A systematic literature review on cloud computing security: threats and mitigation strategies. IEEE Access 9:57792–57807. https://doi.org/10.1109/ACCESS.2021.3073203

Ansari MD, Gunjan VK, Rashid E (2021) On security and data integrity framework for cloud computing using tamper-proofing. ICCCE 2020, Springer, Singapore, pp 1419–1427. https://doi.org/10.1007/978-981-15-7961-5_129

Asghari A, Sohrabi MK, Yaghmaee F (2021) Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm. J Supercomput 77(3):2800–2828. https://doi.org/10.1007/s11227-020-03364-1

Balaji K, Kiran PS, Kumar MS (2021) An energy efficient load balancing on cloud computing using adaptive cat swarm optimization. Mater Today Proc. https://doi.org/10.1016/j.matpr.2020.11.106

Baucas MJ, Spachos P (2020) Using cloud and fog computing for large scale IoT-based urban sound classification. Simulat Model Pract Theor 101:102013. https://doi.org/10.1016/j.simpat.2019.102013

Bello SA, Oyedele LO, Akinade OO et al (2021) Cloud computing in construction industry: Use cases, benefits and challenges. Autom Constr 122:103441. https://doi.org/10.1016/j.autcon.2020.103441

Dalal S, Seth B, Jaglan V et al (2022) An adaptive traffic routing approach toward load balancing and congestion control in Cloud–MANET ad hoc networks. Soft Comput 26:5377–5388. https://doi.org/10.1007/s00500-022-07099-4

de Carvalho PS, Siluk JCM, Schaefer JL, Pinheiro JR, Schneider PS (2021) Proposal for a new layer for energy cloud management: the regulatory layer. Int J Energy Res 45(7):9780–9799. https://doi.org/10.1002/er.6507

Deng S, Zhang C, Li C, Yin J, Dustdar S, Zomaya AY (2021) Burst load evacuation based on dispatching and scheduling in distributed edge networks. IEEE Trans Parallel Distrib Syst 32(8):1918–1932. https://doi.org/10.1109/TPDS.2021.3052236

Ebadifard F, Babamir SM (2021) Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Clust Comput 24(2):1075–1101. https://doi.org/10.1007/s10586-020-03177-0

Fathalla A, Li K, Salah A (2022) Best-KFF: a multi-objective preemptive resource allocation policy for cloud computing systems. Cluster Comput 25(1):321–336. https://doi.org/10.1007/s10586-021-03407-z

Gabhane JP, Pathak S, Thakare NM (2021) Metaheuristics algorithms for virtual machine placement in cloud computing environments—a review. Comput Netw Big Data IoT. https://doi.org/10.1007/978-981-16-0965-7_28

Haseeb-Ur-Rehman RMA, Liaqat M et al (2021) Sensor cloud frameworks: state-of-the-art, taxonomy, and research issues. IEEE Sens J 21(20):22347–22370. https://doi.org/10.1109/JSEN.2021.3090967

Helali L, Omri MN (2021) A survey of data center consolidation in cloud computing systems. Comput Sci Rev 39:100366. https://doi.org/10.1016/j.cosrev.2021.100366

Houssein EH, Gad AG, Wazery YM et al (2021) Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm Evol Comput 62:100841. https://doi.org/10.1016/j.swevo.2021.100841

Hu G, Xu Z, Wang G, Zeng B, Liu Y, Lei Y (2021) Forecasting energy consumption of long-distance oil products pipeline based on improved fruitfly optimization algorithm and support vector regression. Energy 224:120153. https://doi.org/10.1016/j.energy.2021.120153

Karthick G, Mapp G, Kammueller F, Aiash M (2021) Modeling and verifying a resource allocation algorithm for secure service migration for commercial cloud systems. Comput Intell. https://doi.org/10.1111/coin.12421

Karthiban K, Raj JS (2020) An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm. Soft Comput 24:14933–14942. https://doi.org/10.1007/s00500-020-04846-3

Kodli S, Terda S (2021) Hybrid max-min genetic algorithm for load balancing and task scheduling in cloud environment. Int J Intell Eng Syst 14(1):63–71. https://doi.org/10.22266/ijies2021.0228.07

Kouatli I (2018) Emotions in the cloud: a framework architecture for managing emotions with an example of emotional intelligence management for cloud computing organizations. Int J Work Organ Emot 9(2):187–208. https://doi.org/10.1504/IJWOE.2018.093317

Kouatli I (2019) People-process-performance benchmarking technique in cloud computing environment: an AHP approach. Int J Product Perform Manag 69(9):1955–1972. https://doi.org/10.1108/IJPPM-04-2017-0083

Li W, Cao J, Hu K, Xu J, Buyya R (2019) A trust-based agent learning model for service composition in mobile cloud computing environments. IEEE Access 7:34207–34226. https://doi.org/10.1109/ACCESS.2019.2904081

Liu L, Zhu H, Chen S, Huang Z (2022) Privacy regulation aware service selection for multi-provision cloud service composition. Futu Gen Comput Syst 126:263–278. https://doi.org/10.1016/j.future.2021.08.010

Liu Y, Zeng Z, Liu X, Zhu X, Bhuiyan MZA (2019) A novel load balancing and low response delay framework for edge-cloud network based on SDN. IEEE Internet Things J 7(7):5922–5933. https://doi.org/10.1109/JIOT.2019.2951857

Malik MK, Singh A, Swaroop A (2022) A planned scheduling process of cloud computing by an effective job allocation and fault-tolerant mechanism. J Ambient Intell Humaniz Comput 13(2):1153–1171. https://doi.org/10.1007/s12652-021-03537-7

Mireslami S, Rakai L, Wang M et al (2019) Dynamic cloud resource allocation considering demand uncertainty. IEEE Trans Cloud Comput 9(3):981–994. https://doi.org/10.1109/TCC.2019.2897304

Pourghaffari A, Barari M, Kashi SS (2019) An efficient method for allocating resources in a cloud computing environment with a load balancing approach. Concurr Comput Pract Exp 31(17):e5285. https://doi.org/10.1002/cpe.5285

Pradhan A, Bisoy SK (2020) A novel load balancing technique for cloud computing platform based on PSO. J King Saud Univ Comput Inf Sci 34(7):3988–3995. https://doi.org/10.1016/j.jksuci.2020.10.016

Priya V, Kumar CS, Kannan R (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. Appl Soft Comput 76:416–424. https://doi.org/10.1016/j.asoc.2018.12.021

Rajput DS, Basha SM, Xin Q, Gadekallu TR et al (2022) Providing diagnosis on diabetes using cloud computing environment to the people living in rural areas of India. J Ambient Intell Human Comput 13(5):2829–2840. https://doi.org/10.1007/s12652-021-03154-4

Ranapana R, Jayasena KPN (2021) Novel approach for load balancing in mobile cloud computing. In: 2021 6th international conference on information technology research (ICITR), IEEE. https://doi.org/10.1109/ICITR54349.2021.9657441

Reshmi R, Saravanan DS (2020) Load prediction using (DoG–ALMS) for resource allocation based on IFP soft computing approach in cloud computing. Soft Comput 24:15307–15315. https://doi.org/10.1007/s00500-020-04864-1

Sabireen H, Neelanarayanan V (2021) A review on fog computing: architecture, fog with IoT, algorithms and research challenges. Ict Express 7(2):162–176. https://doi.org/10.1016/j.icte.2021.05.004

Saldamli G, Doshatti A, Kapadia D, Nyati D, Bodiwala M, Ertaul L (2021) Enterprise backend as a service (EBaaS). Advances in parallel & distributed processing, and applications. Springer, Cham, pp 1077–1099. https://doi.org/10.1007/978-3-030-69984-0_78

Sefati S, Navimipour NJ (2021) A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm. IEEE Internet Things J 8(20):15620–15627. https://doi.org/10.1109/JIOT.2021.3074499

Seth B, Dalal S, Kumar R (2019) Hybrid homomorphic encryption scheme for secure cloud data storage. Recent advances in computational intelligence. Springer, Cham, pp 71–92. https://doi.org/10.1007/978-3-030-12500-4_5

Siddesha K, Jayaramaiah GV, Singh C (2022) A novel deep reinforcement learning scheme for task scheduling in cloud computing. Cluster Comput 25(6):4171–4188. https://doi.org/10.1007/s10586-022-03630-2

Swarna Priya RM, Bhattacharya S, Maddikunta PKR et al (2020) Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything. J Parallel Distrib Comput 142:16–26. https://doi.org/10.1016/j.jpdc.2020.02.010

Therese MJ, Dharanyadevi P, Harshithaa K (2021) Integrating IoT and cloud computing for wireless sensor network applications. Cloud IoT-Based Veh Ad Hoc Netw. https://doi.org/10.1002/9781119761846.ch7

Tong Z, Deng X, Chen H, Mei J (2021) DDMTS: a novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing. J Parallel Distrib Comput 149:138–148. https://doi.org/10.1016/j.jpdc.2020.11.007

Wei W, Yang R, Gu H, Zhao W et al (2021) Multi-objective optimization for resource allocation in vehicular cloud computing networks. IEEE Trans Intell Transp Syst 23(12):25536–25545. https://doi.org/10.1109/TITS.2021.3091321

Yu C, Wang J, Chen Y, Huang M (2019) Transfer learning with dynamic adversarial adaptation network. In: 2019 IEEE international conference on data mining (ICDM), IEEE. https://doi.org/10.1109/ICDM.2019.00088

Zanbouri K, Jafari Navimipour N (2020) A cloud service composition method using a trust-based clustering algorithm and honeybee mating optimization algorithm. Int J Commun Syst 33(5):e4259. https://doi.org/10.1002/dac.4259

Ziyath S, Senthilkumar S (2021) MHO: meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services. J Ambient Intell Humaniz Comput 12(6):6629–6638. https://doi.org/10.1007/s12652-020-02282-7