Sridaran Rajagopal · Priti Sajja ·
Rohit Thanki · Ajay Kumar (Eds.)

Communications in Computer and Information Science 2823

Artificial Intelligence Based Smart and Secured Applications

4th International Conference, ASCIS 2025
Gujarat, India, September 11–13, 2025
Revised Selected Papers, Part V

Part 5

Springer

Contents

Smart Computing

Enhanced Time Stamp Virtualized Load Balancer in Cloud Computing Web Server

Daniel Raja Singh[1] and R. Durga[2(✉)]

[1] Department of Computer Science, VISTAS, Chennai, India
[2] Department of Advanced Computing and Analytics, VISTAS, Chennai, India
drrdurgaresearch@gmail.com

Abstract. Cloud computing on the Internet is significant because it allows for managing data and applications over the Internet without needing personal devices. Users' jobs are scheduled to run on cloud resources to improve efficiency. The range of energy consumption and efficiency relative to the allocated resources should be considered. Allocating resources, such as CPU, memory, and storage, to various tasks or processes is done through task scheduling. Load balancing divides incoming traffic or workload among several servers or resources to prevent any resource from becoming overburdened. Existing approaches have drawbacks, including difficulty distributing workloads evenly among servers, workload latency, and time-consuming task scheduling. To overcome the issue, the proposed method called the Demand Aware Elastic Load-Priority Queuing Algorithm (DAEL-PQA), allocates the task based on the workload priority. Focuses on task scheduling using a Time Stamp Virtualized load balancer to optimize energy and scheduling time. First Timeline Job Completion Behaviour Rate (TJCBR) allocates the resource task based on the user behavior and feedback rate for the processing timeline. The second step is the Multi-Tenant Spider-Ant Colony Feature Scaling Rate (MT_SACFSR), which selects feature weight and estimates the scaling feature rate for multiple tasks. Demand Aware Elastic Load-Priority Queuing Algorithm is used to analyze a Load Request priority allocation. The final step is the Energy Efficient Distributed Task Scheduling Technique (EEDTST) to optimize the task scheduling level energy in the cloud web server and respond to cloud user requests. The outcomes were finally contrasted to discovered that the proposed algorithm (DAEL-PQA) provides an optimal balance result.

Keywords: cloud computing · resources · Load balancing · scheduling time · Request · energy · web server · Queuing · response

1 Introduction

Technology for cloud computing, the ability to manage numerous services over the Internet, has been revolutionized by virtualization. Its primary goals are high performance, scalability, fault tolerance, high availability, usability, dependability, ease of use, monitoring and management, efficiency, and economy [1]. The ability of cloud computing to supply resources and services on demand via fast computer networks is one of its

R. Sridaran et al. (Eds.): ASCIS 2025, CCIS 2823, pp. 129–146, 2026.
https://doi.org/10.1007/978-3-032-17846-6_8

most significant benefits. As network technologies and infrastructure continue to grow and advance, cloud computing has shown to be more effective at handling a variety of complicated and large-scale end-user (customer) jobs.

Load balancing and resource scheduling strategies are essential for cloud computing systems to efficiently use computational resources [2, 3]. These techniques manage and distribute resources, including CPU, memory, and storage, to guarantee efficient and fair utilization [4]. Resource planning tactics determine how and when to allocate resources among different jobs.

The QoS requirements of cloud applications determine the proper resource scheduling for cloud workloads [5, 6]. Resource allocation in cloud systems is complicated by resource dispersion, diversity, and uncertainty, which current resource allocation techniques cannot resolve. The cloud and load balancer distribute the incoming traffic/load [7]. Set up and implement load balancing in your network system. Load balancing is seen as one technique to attain power optimization. In the cloud, a queuing system solves load balancing and efficiently allocates energy for various servers. Load balancers, software-based, virtual appliances, or hardware, are positioned strategically throughout the network [8].

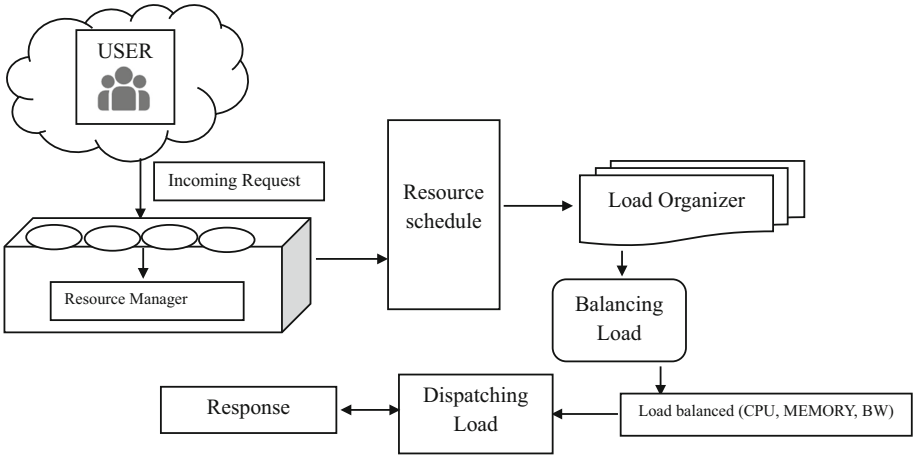

Fig. 1. Basic Flow of Introduction diagram

As shown in Fig. 1, Cloud users prioritize incoming requests by cutting down on vector-based processing time and optimizing resource consumption. Second, the resource manager assigns a higher priority to user requests to increase resource scheduling efficiency. After that, the load balancer balances the web server's load priorities. Finally, a web server queuing system should be developed to schedule different types of resources and expedite response times. Trust management strategies can be used to increase trust between cloud users and the cloud environment. Trust management systems make it easier to identify service providers who are dishonest or compromised. Determining the reliability of cloud users is also essential.

A behavioral feedback-based Timeline Job Completion Operation Rate (TJCBR) is suggested to gauge cloud service providers' (CSPs') dependability in cloud environments. As previously stated, the CSP supplies the CU with cloud services. As a result, you must make sure that only reliable CSPs are offering services in a cloud environment. Trust level is intended to assess the CSP's credibility and guarantee that cloud users receive secure services. In the proposed work, the following are the significant contributions:

- The suggested TJCBR method uses several variables to dynamically assess the cumulative trust of CSPs in cloud environments.
- Suggest using criteria other than the service level agreement to determine feedback trust.
- The Energy Efficient Distributed Task Scheduling Technique (EEDTST) computes the weights, prioritizing the qualities in question.

2 Literature Survey

External web requests are treated as chaotic inputs in the model of the web server system, which is a discrete-time linear time-invariant (LTI) system. Designing and implementing real-time event-triggered control (ETC) presents significant issues when dealing with such disturbance inputs [9]. Text files called web server access log files include crucial information about server activities, client requests sent to the Server, server answers, and more. Numerous improvements in various areas of interest will result from a thorough review of this data. The primary challenge is keeping these files in their original structure for extended periods while conducting analytical procedures whenever necessary to retrieve data that can be used as a foundation for superior decision-making [10].

As a result, a user's computer performance is heavily influenced by how well the services they use work. Consumers must carefully assess and contrast the various options to select the finest cloud service. However, performance studies and comparisons of cloud services are difficult due to different cloud service architectures, a broad range of feature and pricing plans, disparate performance attributes utilized by service providers, and the ambiguity of some performance attributes [11].

Cloud resources must be distributed effectively to respond to customer demands at a fair price. Most current research focuses on costly solutions using on-demand resources or low-cost solutions using allocated resources, which could result in over- or under-provisioning. Because ineffective cloud resource allocation can result in high infrastructure costs and variations in cloud demand, resource allocation is complicated to expose [12]. Needs for allocating resources can differ significantly. These elements can impact schedule performance, resource utilization, and load imbalances. Suggested the Dynamic Resource Allocation Load Balancing (DRALB) scheduling scheme [13, 14].

MEC allocates resources in a quality of experience (QoE-oriented)-oriented manner, considering resource constraints, user variety, and the intricate link between user experience and allotted resources [15]. This challenge is addressed by proposing a closed-loop online resource allocation (CORA) paradigm [16]. The services that comprise DRIVE are administered by the resources of the participants in a cooperative (cooperative) infrastructure that is suggested through market ownership and activity. Employ

cryptographic, safe, and privacy-preserving distribution protocols to build trust in the distribution infrastructure and thwart malevolent activity [17] (Table 1).

Table 1. Existing Survey for Cloud Computing Resource Scheduling

Author Name	Year	Proposed Method	Drawbacks
Y. Xie et al. [18]	2020	Cloud-Priority Imperialist Competitive Algorithm (GICA-CP)	Interoperability between various resources is a crucial concern.
E. Makridis et al. [19]	2022	Maximum Correntropy Criterion Kalman filter (MCC-KF)	Low CPU allocating resources
P. Lai, et al. [20]	2021	1/2 factor approximation algorithm	Assign and Allocate problem
A. Pandey et al. [21]	2023	k-nearest neighbors (k-NN)	lack expert guidance to handle the problem
Z. Xiang et al. [22]	2023	RDC-NeP	cost-effective service provisioning problem

Effective task allocation is crucial in cloud computing because of the limited resources and virtual machines. One of the models of this technology that oversees the backend of servers, data centers, and virtual machines is Infrastructure as a Service (IaaS). Because host overload or underload might result in lengthy processing times or machine malfunctions, cloud service providers must guarantee high service delivery efficiency in this architecture [23]. The dynamic cloud load balancing technique optimizes virtual machine load based on system state and allocates submitted tasks to virtual machines during task execution. After a failover has effectively occurred, cloud fault tolerance is initiated in response to system program failures. Reactive Cloud fault tolerance handles failures after failures [9]. Earlier methods typically phrase this problem as a job allocation optimization problem and consider the present virtual machine (VM) load. However, because they fail to consider a balanced VMH load, these methods only achieve limited performance in real-world scenarios. The research presented here suggests combining two approaches based on genetic algorithms (Table 2).

One of the most critical and challenging problems is the methodical distribution of system resources for batch operations, guaranteeing load balancing of dynamic clusters and satisfying user requests [28]. Reliable cloud results depend on load balancing (LB) and resource scheduling (RS) mechanisms. Tasks submitted by users are computed by the cloud platform using its virtual machines (VMs) [29].

Table 2. Existing approaches for Cloud Computing web server load balancing

Author Name	Year	Proposed Method	Drawbacks
R. Zhanuzak et al. [10]	2024	Enhanced Dynamic Load Balancing (EDLB)	restricted availability of virtual machines (VMs) and resources.
Y. Dong et al. [24]	2021	Deep Reinforcement Learning	The existing joint "cloud-edge" data center is exploited
H. Shen et al. [25]	2020	Resource Intensity Aware Load Balancing Method (RIAL).	Low scalability and high time complexity.
M. S. R. Krishna [26]	2025	Load Optimization Algorithm (LOA), Reinforcement Learning (RL), and Falcon Optimization Algorithm (FOA)	Lower response time
S. Singhal *et al*. [27]	2024	Rock Hyrax-based load balancing algorithm	essential for reducing processing time and utilizing resources

3 Proposed Method

Cloud-based virtual machines and data centers were utilized to apply the algorithm and comprehend how, when properly implemented, the resource scheduling method impacts the load balancing of various computers. Resource scheduling and load balancing are two essential strategies for enhancing performance and resource usage in contemporary computer systems. Algorithms for resource scheduling divide up resources like CPU, memory, and storage among various jobs or applications. A detailed comparison of unique algorithmic approaches such as TJCBR, MT_SACFSR, and DAEL-PQA algorithms is presented.

Figure 2 defines cloud users who request the resource manager allocate requests based on the Timeline Job Completion Behaviour Rate using resource task allocation. The behavior-based task allocation selects the maximum feature weight allocation based on the Multi-Tenant Spider-Ant Colony Feature Scaling Rate. It includes the load balancer that sets the priority queuing Request based on the Demand Aware Elastic Load-Priority Queuing Algorithm and, finally, energy-efficient distributed task scheduling, dispatching load, and responding to user requests.

3.1 Cloud Resource Manager

Cloud Resource Management (CRM) provides application-based services by efficiently managing runtime resources. Predictive models identify the best resources for each job, and models and tools that generate consumption profiles are essential for efficient

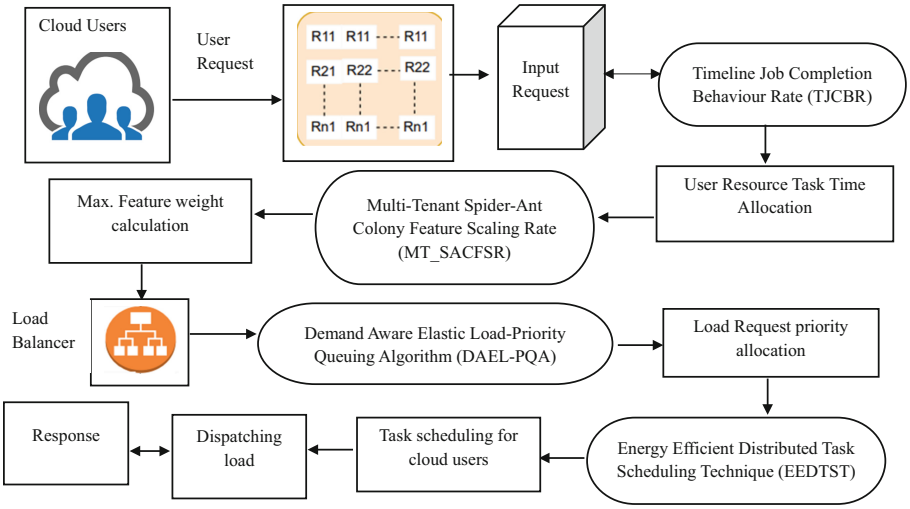

Fig. 2. Proposed Method

application and resource management. These models and tools can be used to apply certain forecasting techniques, estimate the amount of resources required by a workload, and create utilization profiles.

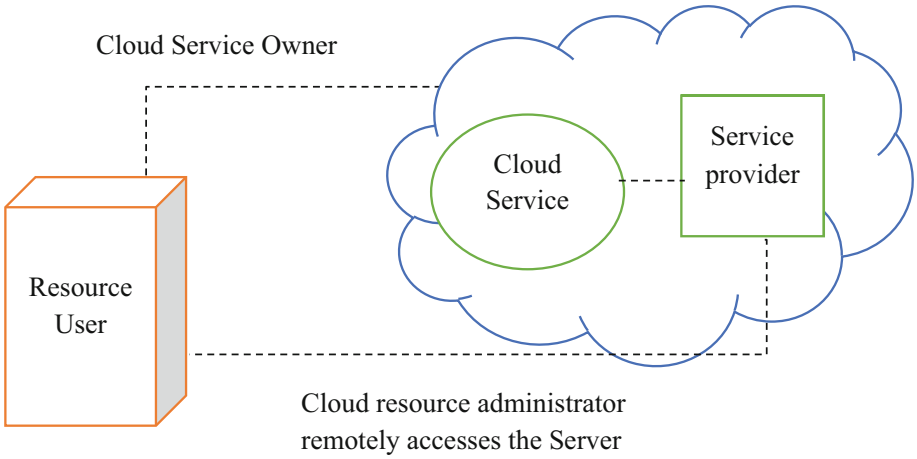

Fig. 3. Resource Management

Figure 3 defines cloud resources as serving millions of users in different ways based on customer requirements. Therefore, resource management (RM) techniques are needed to monitor dynamics and scalability during planning. A resource user who is the cloud service owner requesting the service provider can access the Server as a remote administrator.

3.2 Timeline Job Completion Behaviour Rate (TJCBR)

Propose a behavioural and opinion-based trust calculation scheme to calculate the overall trust process of behavioral trust forverification. Before offering services to cloud consumers (CUs), a specific CSP's trust value is estimated using the Timeline Completion Rate (TCR). Behavioral trust is calculated as part of CSP's trust evaluation in the cloud environment. The behavior of the CSP when offering CU services is referred to as behavioral trust. The focus is on the impartiality of resources and tasks, load balancing of computing resources, and task scheduling in cloud computing environments.

Task Completion Time

Assume that the cloud computing environment is denoted as m resources $Rs = \{Rs_1, Rs_2, \ldots Rs_n\}$, , number of task (n) represents as, $T = \{T_1, T_2, \ldots T_n\}$, so the cloud computing system can be described as Cloud = (Rs, T).

A set of m resource loads can be expressed as: $L = \{L_1, L_2, \ldots L_n\}$. The predicted shortest job completion time when m resources are used to perform n tasks can be written as an m*n matrix.

$$TCR = \begin{bmatrix} Ct_1 & \cdots & Ct_m \\ \vdots & C_{xy} & \vdots \\ Ct_{n1} & \cdots & Ct_{nm} \end{bmatrix}$$

The predicted element C_{mn} represents the job completion task time T_x is done on the resources. Rs_y represents the computing task queuing. Users are assured they can complete their tasks within the predicted minimum job completion time. The ideal outcome of task scheduling is that each task can be scheduled to the resource with the shortest predicted completion time and the smallest load.

The first step is to initialize the team TCR and L package. The second step is to generate a timeline of all the elements in the matrix to find the minimum expected completion time.

$$C_{min} = Min\left(C_{xy}\right), 1 \le m \le x, 1 \le n \le y,$$

Find the lowest load of the resource in the task (T), $T_{min} = Min\left(l_{xy}\right)$

$$\frac{T_{min}}{T^{Rs}_{xy\,min}} \ge \frac{L_{min}}{l^{Rs}_{y}min}$$

$$\frac{T_{min}}{T^{Rs}_{xy\,min}} < \frac{L_{min}}{l^{Rs}_{y}min}$$

$l^{Rs}_{y}min$ is the load value of the resource with the minimum completion time in the all elements in the matrix TCR. $T^{Rs}_{xy\,min}$ predicts the completion time of a task run by the resource with the lowest load value.

3.3 Multi-tenant Spider-Ant Colony Feature Scaling Rate (MT_SACFSR)

Multi-tenant spider ant colony feature expansion rate planning to resolve feature expansion rate. To identify the optimal path to the food source, ants can follow some of these foods to the colony's nest. To select the same path if the pheromone value on that path is higher (Porb).

For processing, the operators send in n tasks/scaling rate $\{CT1; CT2; \ldots\ldots : CTn\}$. . Each cloudlet's scaling length (CTi) is expressed in millions of instructions (MI). The scheduler first determines each task's operation time (ET) in each VMJ using the formula below:

$$ET(\text{CTi, vmj}) = \frac{(\text{CTi})}{Tot\_MIPS(\text{vmj})}$$

where, $Tot\_MIPS(\text{vmj}) = \text{vmj} * \text{vmmips}$. The scheduler then determines how much energy the virtual machine needs to complete all of the tasks that have been assigned to it and how much it will cost to operate them. The scheduler chooses the virtual machine that best suits each task's requirements and has a fair processing cost to reduce the cost of processing jobs.

Depending on social and environmental variables, a parent group may split into smaller subgroups (fission process) or reintegrate into a larger group (fusion process).

The Initialization Phase

A random uniformly distributed population of spider Ant is first created by SAC, with each spider Ant SA_{xl} (x = {0, 1, 2, 3…. A}) being a D-dimensional vector. The value of SA_{xl}, which denotes the xth spider monkey in the population, can be computed as follows:

$$SA_{xl} = SA_{min} + ran(0, 1) * (SA_{maxl} - SA_{min})$$

Local Leader and Global Leader Phase

Every spider ant colony in LLP advances its standing by drawing on the knowledge of its local leader and other group members. After evaluating the new state based on its fitness value, the SM updates its public with the newly produced state. If the new value is superior to the previous one, the SM changes its state.

$$SA_{newx} = \begin{cases} SA_{xl} + ran(0, 1) * \left(LL_{gx} - SA_{xl}\right) + ran(1, -1) * (SA_{rl} - SA_{xl}) ran(0, 1) \geq P \\ SA_{xl} Otherwise \end{cases}$$

$$SA_{newx} = \begin{cases} SA_{xl} + ran(0, 1) * \left(GL_{gx} - SA_{xl}\right) + ran(1, -1) * (SA_{rl} - SA_{xl}) ran(0, 1) \geq Pi \\ SA_{xl} Otherwise \end{cases}$$

where l is the xth dimension of the xth spider, LL_{gx} is, and GL_{gx} is the lth dimension of the GL state, whose values are randomly selected from {0,1, 2…. x]. The likelihood of present position confusion and its range of values [0.1, 0.9] make up the x dimension of the local location of group k, pr. While random p is a consistently random value in the range, SA_{xl} is the lth dimension of SA chosen randomly from the group ($x \neq 1$).

The cooperative behavior of actual ants during foraging is replicated by the ant colony scaling ratio algorithm.

$$p = 0.9 * \frac{f(SA_{xl})}{maxi_{FIt}} + 0.1$$

$$Prob_{xy}^{k}(r) = \begin{cases} \dfrac{\partial_{xy}(t)\vartheta_{xy}()}{\sum_{Z\epsilon x^{i}} \tau\delta xt(i)\ \vartheta_{tx}(i)} \\ 0 \end{cases}$$

The pheromone $\delta_{xy}(t)$ value on the path xy is the captivation of the pheromone in task x^{i}, the virtual machine (VM). The pheromone's initial value is set to a tiny positive constant. δ_0. $\vartheta_{tx}(i)$ Indicates that the Ant has visibility of k in the current iteration

$$\vartheta_{tx}(i) = 1[MI(CT_{xl})\backslash MIPS(vm_x)]$$

$$MIPS(vm_x) = Pe\_No * MIPS\_pe$$

where *Pe_No* is the processing elements of the VM and the number of processing units, *MIPS_pe* represents each processor's MIPS. After receiving a job request from the task manager, the cloud scheduler allocates incoming jobs to available virtual machines using the specified MT_SACFSR scheduling mechanism. The cloud network server's task scaling rate and virtual machine data are the basis for MT_SACFSR's scheduling decisions.

3.4 Demand-Aware Elastic Load-Priority Queuing Algorithm (DAEL-PQA)

The proposed algorithm, DAEL-PQA, aims to achieve task scheduling under deadline constraints and utilizes load balancing across virtual machines to optimize resource utilization and minimize downtime. It prioritizes high-priority traffic while ensuring efficient use of resources. The queuing algorithm reduces the latency by establishing a virtual queue that is supplied at a fraction of the real service rate and utilizing the virtual queue length value during the application process.

Queue evolution for a link l is given by

$$\theta_t(l) = (x_t(l) + y_t(l) - \alpha_1 q_l)_{\theta_t(l)}$$

where $y_t(l)$. . Is the aggregated elastic, and (t) is the continuous time index. Virtual queues α1 and α2 regulate the overall load $x_t(l)$, the link l ∈ L's capacity. Participatory queuing priority concepts—the foundation of priority-based queuing—must be closely tied to packet flow priorities.

L_{xy}^{min} and L_{xy}^{max} are the min and max values of the packet flow priority, and l_{xy} is a Total number of packet flow priorities for the lth sub-queue.

$$. l_{xy} = L_{xy}^{min} - L_{xy}^{max} + \left(y = \overline{1, N}, t = \overline{1, L}\right)$$

Input: Req R_{l1},...R_{lN}, Server S, $L_{priority}\,(L_P)$&$L_{priority}\,(L_{CPU})$

Output: Resource task allocation results in load-balancing priority

Foreach R_{l1} in queue Q^1 do //Q^1 –*queue*

If not, first req, then

$L_{priority}$ ←available state of Server by last n req

If $L_{priority} > R_{l1}$ *then*

Return $L_{priority}$ ←$Q^1(R_{lN})$

Else

Return Q^2←$Q^1(R_{lN})$

End of $L_{priority} > R_{lN}$

Else

Return $L_{priority}$ ← Available load request queue.

End if

End

In the proposed cloud resource allocation mechanism, every provider has container scheduler software running in its cloud center. All the host state information has been sent to the compute nodes to make scheduling decisions periodically.

3.5 Energy Efficient Distributed Task Scheduling Technique (EEDTST)

To achieve the service level agreement for cloud users and increase the system's energy efficiency, the cloud computing system is optimized using an energy-saving job scheduling approach. The cloud dynamically modifying resource allocation in response to workload demands, EEDTST is for lower energy consumption. Tasks are service requests that users submit to the cloud system.

The scheduler performs collective scheduling whenever a user task enters the cloud system. One way to model the amount of energy needed to complete a task (t) is as follows:

$$S_{xy} = E_{xy} * T_{xy}$$

where E_{xy} j - power consumption of task x on core y, $T_{xy}-$ execution time of job x on core y. Energy-saving task scheduling.

$$Mini \sum_{x=1}^{i} \sum_{y=1}^{j} E_{xy} * T_{xy}$$

$$\sum_{x=1}^{i} t_{xy} = 1, \forall x \in 1, 2 \ldots, i$$

$$\sum_{y=1}^{j} t_{xy} \leq 1, \forall x \in 1, 2 \ldots, i$$

where t_{xy} is a binary variable that indicates whether task x is assigned to core y, and n, m, and t_{xy} are the number of tasks, centers, and centers, respectively.

Input: Task list (Tl), cloud web server (W)

Output: Task schedule (Ts)

Task (T) based on dependencies priority (P)

For each task (t) in T, do

For each Server (s) in P do

Compute the score of energy level scheduling t to P

End for

Assign the highest priority score level

Update the scheduling task based on the time

End for

Predict energy efficiency in the web server for task scheduling

Select the task minimizing the deadlines

End for

Applying selected and executing the responsive tasks

Return task schedule

The task prioritizing stage ensures that high-priority activities are scheduled first, in accordance with their dependencies and deadlines. When comparing the number of jobs to the time problem, an energy-efficient task scheduling algorithm scales well. Each step's performance determines the algorithm's approximation rate, and additional research and experimental evaluation are needed to compare the algorithm's performance.

4 Experimental Setup

Evaluate the algorithm's performance on task sets, including 50–500 tasks with execution times ranging from 10 to 100 ms. A project's deadline is defined by its execution time plus a random factor ranging from 1.2 to 2.0. Also create a set of task dependencies to simulate limits in real-world scenarios. Compare the suggested method to four alternative scheduling algorithms: Cloud-Priority Imperialist Competitive Technique (GICA-CP), Maximum Correntropy Criterion Kalman Filter (MCC-KF), Enhanced Dynamic Load Balancing (EDLB), and Resource Intensity Aware Load Balancing Method (RIAL).

4.1 Results and Discussion

Response time, which is the amount of time required to complete a task or Request and is measured in milliseconds or seconds from task submission to completion, is one of the performance indicators used for evaluation. Resource usage, calculated as (Resources Consumed/Total Resources Available) * 100%, is the percentage of available resources consumed by currently running processes. Energy consumption, measured in Watt-hours (Wh) or Joules (J), is the total energy required to execute a task.

The suggested method outperformed other methods when storage optimization was evaluated using a confusion matrix to measure performance in terms of accuracy, time complexity, cost-effectiveness, and energy efficiency.

Table 3. Simulation Processing Tools

Parameters	Values
Size of data	25 Gb
Process Rate	[250, 1500] MIPS
Number of tasks	500
Bandwidth	100,500 and1024 Mbs
Tool	Visual Studio/c#

Table 3 defines the simulation parameters, including the number of tasks, processing time, and bandwidth used in the Visual Studio tool, which is introduced based on the proposed technique to provide better performance under different levels of testing (Table 4).

Table 4. Comparison of Various Methods

Methods	Accuracy %	Storage %	Success Rate %	False Rate%	Energy efficient (J)	average task response time (ms)	Time Complexity ms
GICA-CP	85	38.4	68	6.8	74	0.35	0.33
MCC-KF	88	49.7	72	5.7	79	0.30	0.29
EDLB	92	52.1	75	4.1	84	0.27	0.24
RIAL	93.5	68.7	78	3.9	89	0.20	0.18
DAEL-PQA	96.7	77.9	81	3.5	91	0.18	0.15

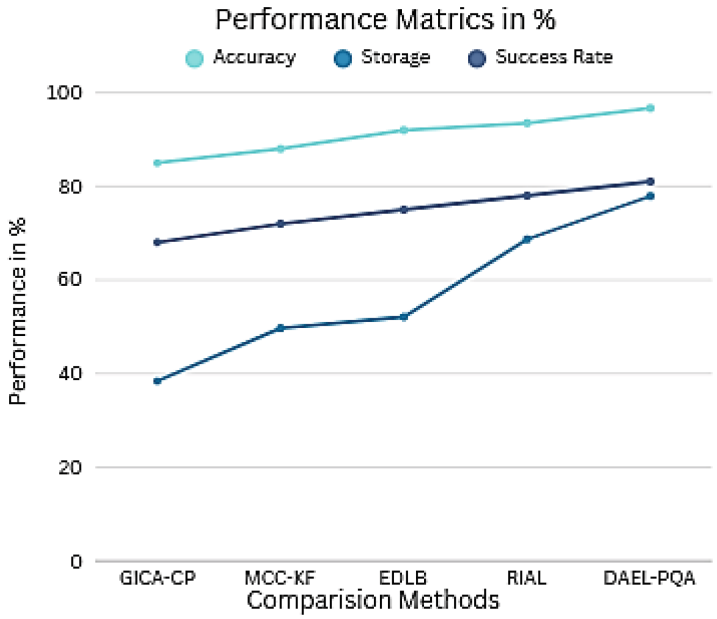

Fig. 4. Performance Matrices

Figure 4 clearly illustrates data sizes to observe the performance of the tested data analysis. GICA-CP accuracy is 85%, MCC-KF is 88%, EDLB is 92%, RIAL is 93.5%, and DAEL-PQA is 96.7%. Then, testing the performance storage rate is GICA-CP is 38.4%, MCC-KF is 49.7%, EDLB is 52.1%, RIAL is 68.7%, and DAEL-PQA is 77.9%. Compared with GICA-CP, the task request success rate is 68%, MCC-KF is 72%, EDLB is 75%, RIAL is 78%, and DAEL-PQA is 81%. The proposed DAEL-PQA performance can be associated with the tested data in 20 GB size.

Figure 5 illustrates the various methods used to calculate false ratio yields. Recommended values are 6.8% for GICA-CP, 5.7% for MCC-KF, 4.1% for EDLB, 3.9% for

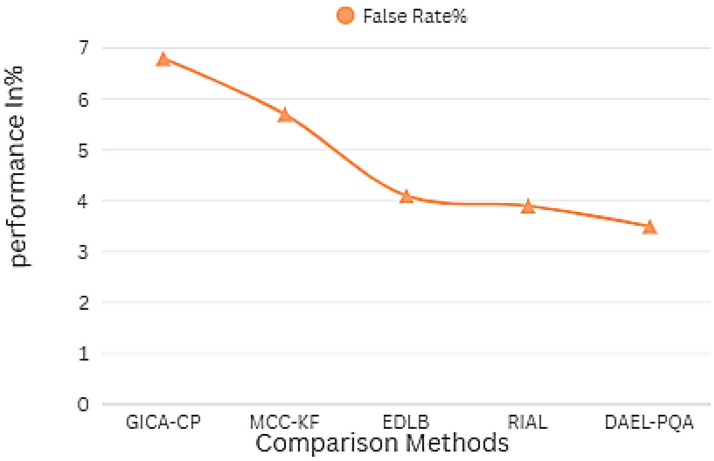

Fig. 5. Analysis of False Rate

RIAL and 3.5% for DAEL-PQA. Compared to strategies tested with 20 GB of data. The statistics show that the suggested DAEL-PQA channel produces fewer false positives than the other approaches.

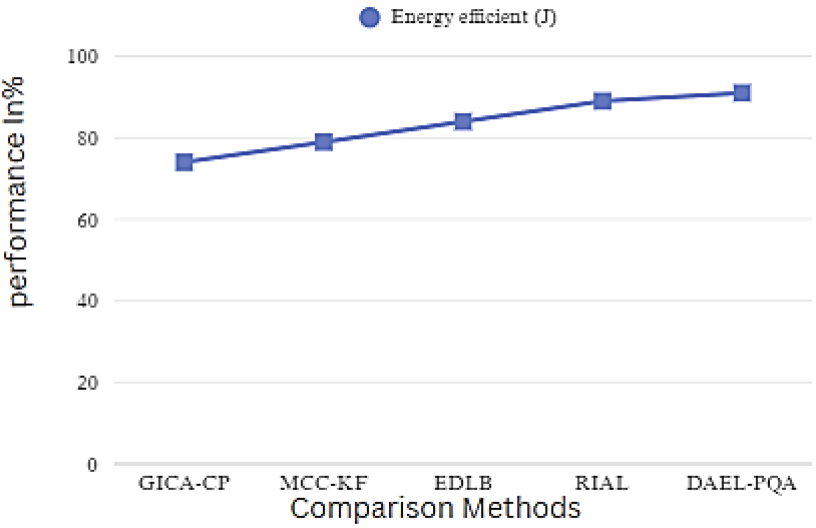

Fig. 6. Energy-efficient

Figure 6 defines task scheduling as assigning virtual machines to available servers and routing user requests or tasks to the correct virtual machine. The suggested GICA-CP is 74 J, MCC-KF is 79 J, EDLB is 84 J, RIAL is 89 J and DAEL-PQA is 91 J (Fig. 7).

The average time a cloud service takes to process a request and return a response. This time is typically measured in milliseconds. The Effect of Limited Bandwidth on

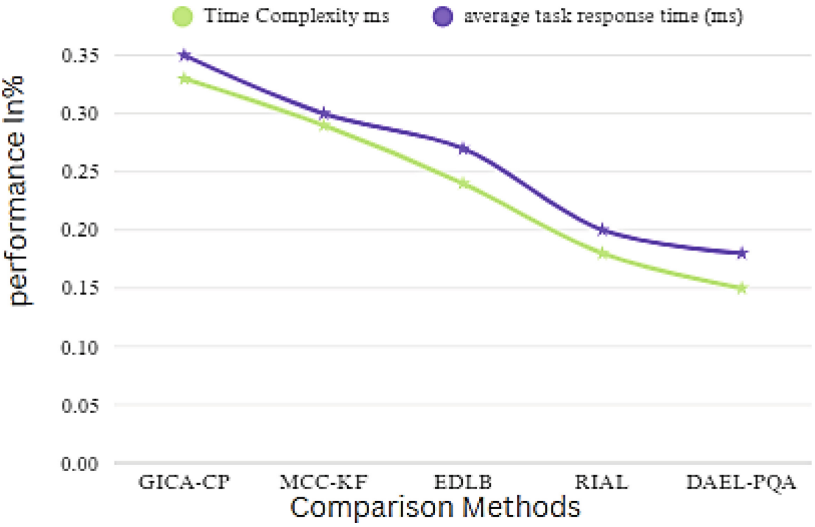

Fig. 7. Analysis of task time complexity and response time

Cloud Systems' Average Response Time. It establishes the maximum bandwidth allotted to requests and answers from cloud network users. And the time process of several ways of measuring production. When compared to alternative approaches, the suggested technique saves time. When evaluated with 10 GB of data, the suggested DAEL-PQA performs better than the current algorithms.

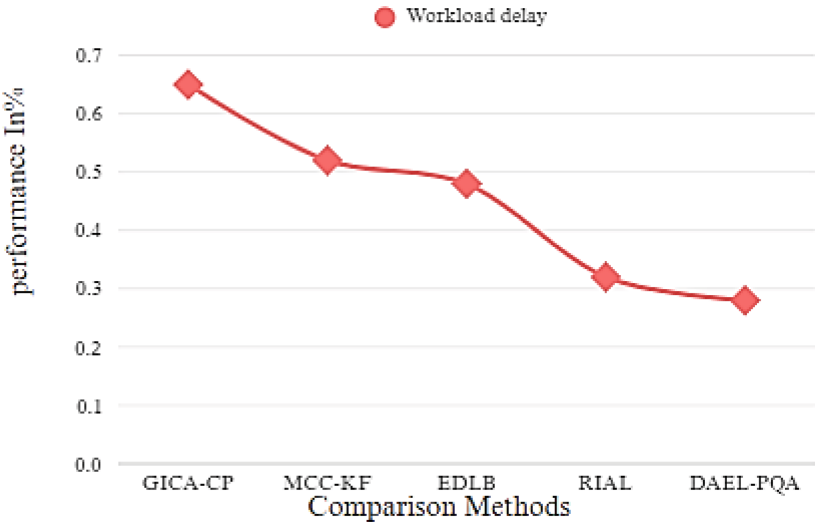

Fig. 8. Workload delay

Figure 8 defines the average reduction time for the accepted workload under different delay constraints. The results showed that GICA-CP 0.65 s, MCC-KF 0.52 s, EDLB 0.48 s, RIAL 0.32 s, and DAEL-PQA 0.28 s. Latency is significantly reduced when workload latency is limited to 50 to 100 ms. This is because the resource allocation is adjusted for each interval as the amount of work to be reassigned increases. These algorithms aim to optimize the resource utilization of fog and cloud nodes, i.e., to maximize the information processing capabilities of nodes.

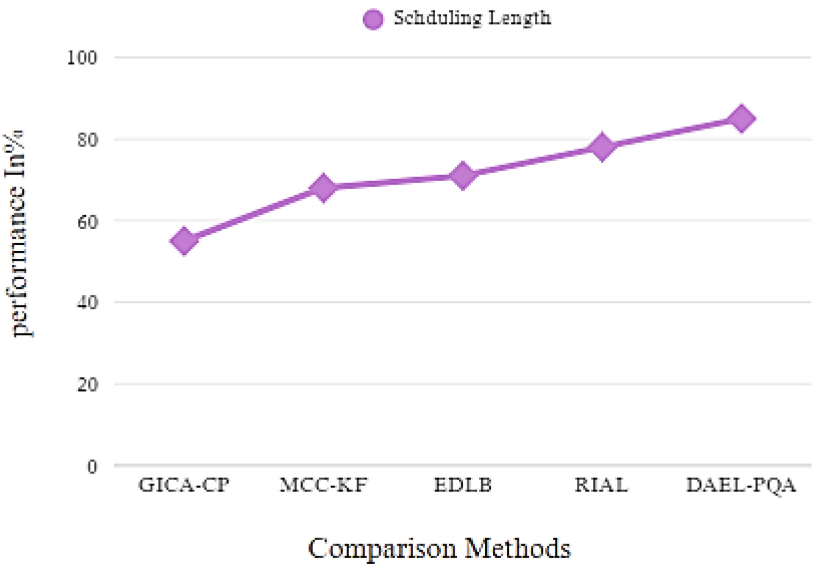

Fig. 9. Scheduling Length

Figure 9 defines the planning length as, depending on the workload, 55% for GICA-CP, 68% for MCC-KF, 71% for EDLB, 78% for RIAL, and 85% for DAEL-PQA. From this analysis, it can be seen that ECBTSA-IRA achieves shorter schedule lengths compared to other algorithms.

5 Conclusion

In conclusion, task scheduling in cloud computing data centers preserves deadline-based service level agreements while lowering energy usage by decreasing the number of servers. Use integer programming optimization to reduce task reaction time and energy usage in data center processing jobs. The performance of the suggested approach is compared with a random-based task scheduling strategy using a Visual Studio simulation with independent exponentially distributed job arrivals. According to simulation studies, a data center that uses the suggested work scheduling scheme has an average lower server energy consumption than one that uses a task scheduling system—suggested energy efficiency at the expense of lengthier task deadlines and response times.

Cloud computing resources must be used as efficiently as possible. Because cloud-based distributed algorithms may effectively use computer resources, minimize energy consumption, and optimize spacing, they are appropriate for web server environments. According to experimental data, the suggested approach (DAEL-PQA) outperforms the maximum utility scheduling optimization model. More reliable mechanisms should be emphasized, and the fundamental goals (scheduling rate, load balance, cost, etc.) should be covered.

References

1. Wu, C., Toosi, A.N., Buyya, R., Ramamohanarao, K.: Hedonic pricing of cloud computing services. IEEE Trans. Cloud Comput. **9**(1), 182–196 (2021). https://doi.org/10.1109/TCC.2018.2858266qw
2. Motai, Y., Henderson, E., Siddique, N.A., Yoshida, H.: Cloud colonography: distributed medical testbed over cloud. IEEE Trans. Cloud Comput. **8**(2), 495–507 (2020). https://doi.org/10.1109/TCC.2015.2481414
3. Wang, B., Wang, C., Huang, W., Song, Y., Qin, X.: A survey and taxonomy on task offloading for edge-cloud computing. IEEE Access **8**, 186080–186101 (2020). https://doi.org/10.1109/ACCESS.2020.3029649
4. Danquah, W.M., Altilar, D.T.: UniDRM: unified data and resource management for federated vehicular cloud computing. IEEE Access **9**, 157052–157067 (2021). https://doi.org/10.1109/ACCESS.2021.3127521
5. Domanal, S.G., Guddeti, R.M.R., Buyya, R.: A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment. IEEE Trans. Serv. Comput. **13**(1), 3–15 (2020). https://doi.org/10.1109/TSC.2017.2679738
6. Su, X., Li, X.: Elastic performance test method of web server in cloud computing environment. J. Web Eng. **20**(5), 1641–1658 (2021). https://doi.org/10.13052/jwe1540-9589.20514
7. Wen, X., Zheng, Y.: The application of artificial intelligence technology in cloud computing environment resources. J. Web Eng. **20**(6), 1853–1866 (2021). https://doi.org/10.13052/jwe1540-9589.2067
8. Yan, C., Zhang, Y., Zhong, W., Zhang, C., Xin, B.: A truncated SVD-based ARIMA model for multiple QoS prediction in mobile edge computing. Tsinghua Sci. Technol. **27**(2), 315–324 (2022). https://doi.org/10.26599/TST.2021.9010040
9. Tawfeeg, T.M., et al.: Cloud dynamic load balancing and reactive fault tolerance techniques: a systematic literature review (SLR). IEEE Access **10**, 71853–71873 (2022). https://doi.org/10.1109/ACCESS.2022.3188645
10. Hung, L.-H., Wu, C.-H., Tsai, C.-H., Huang, H.-C.: Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods. IEEE Access **9**, 49760–49773 (2021). https://doi.org/10.1109/ACCESS.2021.3065170
11. Singh, D., Dwarakanath, K., Pasumarthy, R.: Event-triggered control design for systems with exogenous inputs: application for auto-scaling of cloud-hosted web servers. IEEE Trans. Syst. Man Cybernet. Syst. **52**(8), 5201–5211 (2022). https://doi.org/10.1109/TSMC.2021.3121681
12. Zagan, E., Danubianu, M.: Data lake architecture for storing and transforming web server access log files. IEEE Access **11**, 40916–40929 (2023). https://doi.org/10.1109/ACCESS.2023.3270368
13. Nadeem, F.: A unified framework for user-preferred multi-level ranking of cloud computing services based on usability and quality of service evaluation. IEEE Access **8**, 180054–180066 (2020). https://doi.org/10.1109/ACCESS.2020.3027775

14. Mireslami, S., Rakai, L., Wang, M., Far, B.H.: Dynamic cloud resource allocation considering demand uncertainty. IEEE Trans. Cloud Comput. **9**(3), 981–994 (2021). https://doi.org/10.1109/TCC.2019.2897304
15. Chhabra, S., Singh, A.K.: Dynamic resource allocation method for load balance scheduling over cloud data center networks. J. Web Eng. **20**(8), 2269–2284 (2021). https://doi.org/10.13052/jwe1540-9589.2083
16. Fu, L., Tong, J., Lin, T., Zhang, J.: Data-driven online resource allocation for user experience improvement in mobile edge clouds. IEEE Trans. Wirel. Commun. **23**(10), 13707–13721 (2024). https://doi.org/10.1109/TWC.2024.3403996
17. Chard, K., Bubendorfer, K.: Cooperative resource allocation: building an open cloud market using shared infrastructure. IEEE Trans. Cloud Comput. **7**(1), 183–195 (2019). https://doi.org/10.1109/TCC.2016.2594174
18. Xie, Y., Guo, Y., Mi, Z., Yang, Y., Obaidat, M.S.: Loosely coupled cloud robotic framework for QoS-driven resource allocation-based web service composition. IEEE Syst. J. **14**(1), 1245–1256 (2020). https://doi.org/10.1109/JSYST.2019.2904098
19. Makridis, E., Deliparaschos, K., Kalyvianaki, E., Zolotas, A., Charalambous, T.: Robust dynamic CPU resource provisioning in virtualized servers. IEEE Trans. Serv. Comput. **15**(2), 956–969 (2022). https://doi.org/10.1109/TSC.2020.2966972
20. Lai, P., Fan, R., Zhang, X., Zhang, W., Liu, F., Zhou, J.T.: Utility optimal thread assignment and resource allocation in multi-server systems. IEEE/ACM Trans. Netw. **30**(2), 735–748 (2022). https://doi.org/10.1109/TNET.2021.3123817
21. Pandey, A., Calyam, P., Liu, Z., Wang, S., Chemodanov, D., Joshi, T.: Knowledge-engineered multi-cloud resource brokering for application workflow optimization. IEEE Trans. Netw. Serv. Manag. **20**(3), 3072–3088 (2023). https://doi.org/10.1109/TNSM.2022.3227767
22. Xiang, Z., Zheng, Y., Zheng, Z., Deng, S., Guo, M., Dustdar, S.: Cost-effective traffic scheduling and resource allocation for edge service provisioning. IEEE/ACM Trans. Netw. **31**(6), 2934–2949 (2023). https://doi.org/10.1109/TNET.2023.3265002
23. Shafiq, D.A., Jhanjhi, N.Z., Abdullah, A., Alzain, M.A.: A load balancing algorithm for the data centres to optimize cloud computing applications. IEEE Access **9**, 41731–41744 (2021). https://doi.org/10.1109/ACCESS.2021.3065308
24. Zhanuzak, R., Ala' Anzy, M.A., Othman, M., Algarni, A.: Optimizing cloud computing performance with an enhanced dynamic load balancing algorithm for superior task allocation. IEEE Access **12**, 183117–183132 (2024). https://doi.org/10.1109/ACCESS.2024.3508793
25. Shen, H., Chen, L.: A resource usage intensity aware load balancing method for virtual machine migration in cloud datacenters. IEEE Trans. Cloud Comput. **8**(1), 17–31 (2020). https://doi.org/10.1109/TCC.2017.2737628
26. Krishna, M.S.R., Khasim Vali, D.: Meta-RHDC: meta reinforcement learning driven hybrid lyrebird falcon optimization for dynamic load balancing in cloud computing. IEEE Access **13**, 36550–36574 (2025). https://doi.org/10.1109/ACCESS.2025.3544775
27. Singhal, S., et al.: Energy efficient load balancing algorithm for cloud computing using rock hyrax optimization. IEEE Access **12**, 48737–48749 (2024). https://doi.org/10.1109/ACCESS.2024.3380159
28. Li, T., Ying, S., Zhao, Y., Shang, J.: Batch jobs load balancing scheduling in cloud computing using distributional reinforcement learning. IEEE Trans. Parallel Distrib. Syst. **35**(1), 169–185 (2024). https://doi.org/10.1109/TPDS.2023.3334519
29. Lahande, P.V., Kaveri, P.R., Saini, J.R., Kotecha, K., Alfarhood, S.: Reinforcement learning approach for optimizing cloud resource utilization with load balancing. IEEE Access **11**, 127567–127577 (2023). https://doi.org/10.1109/ACCESS.2023.3329557