European Journal of Prosthodontics and Restorative Dentistry (2026) 34(5s), 332–348

EDLB-R: Enhancing Cloud Computing Performance Through Intelligent Load Balancing and Resource Allocation

Abstract— Cloud computing has become a widely adopted paradigm for providing scalable and on-demand computing resources. However, efficient task scheduling and load balancing remain major challenges due to dynamic workloads and heterogeneous virtual machine (VM) resources. Conventional load balancing algorithms primarily focus on processor utilization while often neglecting memory availability, leading to increased execution time, poor resource utilization, and uneven workload distribution. This paper proposes an Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R) algorithm that incorporates available RAM into the VM selection process to improve scheduling efficiency and resource allocation. The proposed algorithm was implemented using the CloudSim simulation toolkit and evaluated in the Eclipse IDE using 2 virtual machines with workloads ranging from 10 to 100 cloudlets. Performance was assessed using conventional metrics, including total makespan, average makespan, average execution time, resource utilization, and load balancing, along with additional Quality of Service (QoS) metrics comprising response time, throughput, and deadline success rate. Experimental results demonstrate that EDLB-R consistently outperforms the benchmark algorithm. At 100 cloudlets, the proposed EDLB-R algorithm reduces total makespan by 29%, average makespan by 7%, and average execution time by 14% compared with the baseline EDLB algorithm. Furthermore, resource utilization improves by 2.01%, increasing from 65.57% to 66.89%, while load balancing is enhanced by 34%, increasing from 90.34% to 121.06%. Overall experimental results demonstrate that the proposed EDLB-R algorithm consistently outperforms the baseline across all evaluated workloads by reducing scheduling overhead, improving resource utilization, and achieving better workload distribution. These findings indicate that EDLB-R provides an efficient, scalable, and reliable scheduling solution for Infrastructure-as-a-Service (IaaS) cloud environments.

Keywords
Cloud Computing, Load Balancing, Task Scheduling, RAM-Aware Scheduling, Quality of Service

Authors
Meenakshi Saini[1]
Designation: Research Scholar Institution: Galgotias University, Greater Noida Department: Electrical, Electronics and Communication Engineering
Email: meenakshisaini929@gmail.com
Address: Plot No. 2, Sector 17-A, Yamuna Expressway, Greater Noida, Gautam Buddh Nagar, Uttar Pradesh, India, PIN – 203201
ORCID iD: 0009-0006-5274-8784
https://orcid.org/0009-0006-5274-8784

Prabhakar Agarwal[2] (Corresponding Author)
Designation: Associate Professor
Qualification: PhD Institution: Galgotias University, Greater Noida Department: Electrical, Electronics and Communication Engineering
Email: prabhakarvirgo15@gmail.com
Address: Plot No. 2, Sector 17-A, Yamuna Expressway, Greater Noida, Gautam Buddh Nagar, Uttar Pradesh, India, PIN – 203201
ORCID iD: 0000-0003-4818-9351
https://orcid.org/0000-0003-4818-9351

Received:16-06-2026
Revised:20-07-2026
Accepted: 24-07-2026
Doi: 10.1922/ejprd.v34i5s.1561

ejprd.org- Published by Riset Publishing Services LLC. Copyright © 2026 by Riset Publishing Services LLC

I. Introduction

Cloud computing has revolutionized the way computing resources are delivered by providing scalable, on-demand, and cost-effective services over the Internet. Organizations increasingly rely on cloud infrastructures to execute computationally intensive applications without investing in expensive hardware resources. Through virtualization technologies, cloud service providers can dynamically allocate processing power, memory, storage, and network resources to users based on workload demands. Despite these advantages, efficient resource management remains one of the most significant challenges in cloud computing environments.

designing intelligent load balancing algorithms capable of adapting to dynamic cloud environments has become an active research area.

Numerous static and dynamic load balancing algorithms have been proposed to improve cloud performance. Static algorithms generally allocate tasks based on predefined resource information and are suitable for predictable environments. However, they cannot effectively adapt to dynamic workload variations. Dynamic load balancing algorithms overcome this limitation by continuously monitoring system resources and redistributing workloads according to current resource availability. Although these approaches improve scheduling efficiency, many existing algorithms primarily focus on CPU utilization while giving limited consideration to memory availability during task allocation. As a result, cloudlets may be assigned to virtual machines with insufficient RAM, causing execution delays, excessive waiting time, and inefficient resource utilization.

To address these limitations, this paper proposes an Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R) algorithm. The proposed approach extends conventional dynamic load balancing by incorporating available RAM as a scheduling parameter before assigning cloudlets to virtual machines. During scheduling, EDLB-R evaluates the memory capacity and current workload of each VM, ensuring that tasks are allocated only to resource-capable virtual machines. This memory-aware scheduling strategy reduces resource contention, prevents VM overloading, improves workload distribution, and enhances overall scheduling efficiency.

The proposed algorithm is implemented and evaluated using the CloudSim simulation framework under varying workloads ranging from 10 to 100 cloudlets across multiple virtual machine configurations. Performance is assessed using widely accepted cloud computing metrics, including total makespan, average makespan, execution time, resource utilization, and load balancing efficiency. Experimental results demonstrate that the proposed EDLB-R algorithm consistently outperforms the baseline approach by reducing makespan and execution time while significantly improving workload distribution. Although resource utilization exhibits a slight decrease due to increased parallelism, the overall system performance is substantially enhanced through efficient and balanced resource allocation.

Among the various resource management issues, task scheduling and load balancing play a crucial role in improving the overall performance of cloud systems. Load balancing aims to distribute incoming tasks uniformly among available virtual machines (VMs) to maximize resource utilization, minimize task completion time, and prevent resource overloading. Inefficient scheduling may result in some VMs becoming overloaded while others remain underutilized, leading to increased execution time, longer makespan, higher response time, and reduced Quality of Service (QoS). Therefore,

The main contributions of this paper are summarized as follows:

- Proposes the Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R) algorithm for intelligent cloud task scheduling.
- Introduces memory-aware virtual machine selection to improve resource allocation decisions.
- Implements and evaluates the proposed algorithm using the CloudSim simulator under different workload scenarios.
- Performs a comprehensive performance comparison using total makespan, average makespan, execution time, resource utilization, and load balancing metrics.
- Demonstrates consistent improvements in scheduling efficiency, workload distribution, and overall cloud system performance compared with the baseline load balancing algorithm.

II. Literature Survey

Cloud computing has transformed the delivery of computing services by enabling scalable, on-demand, and virtualized access to computational resources. However, efficient load balancing and task scheduling remain major challenges because they directly affect resource utilization, response time, throughput, makespan, and Quality of Service (QoS). Numerous researchers have proposed static, dynamic, heuristic, metaheuristic, and intelligent scheduling algorithms to address these challenges [1], [2].

Early cloud scheduling approaches mainly relied on **static load balancing algorithms**, including Round Robin (RR), Min-Min, Max-Min, and Opportunistic Load Balancing (OLB). These algorithms allocate tasks based on predefined scheduling rules without considering the current status of virtual machines. Although static approaches have low computational complexity, they cannot adapt to changing workloads, leading to poor scalability and uneven workload distribution in heterogeneous cloud environments [3], [4].

Dynamic load balancing algorithms were later introduced to overcome these limitations by continuously monitoring virtual machine status before task allocation. Compared with static scheduling, dynamic algorithms provide better adaptability, lower response time, and improved resource utilization. However, many existing dynamic schedulers primarily consider processor utilization while overlooking memory availability, bandwidth, and other resource constraints that significantly influence cloud performance [5], [6].

Copyright © 2026 by Riset Publishing Services LLC

CloudSim has become one of the most widely used simulation platforms for evaluating cloud resource allocation and scheduling algorithms. Developed by Calheiros et al., CloudSim enables researchers to model data centers, virtual machines, cloudlets, and resource provisioning policies under realistic cloud environments. Due to its flexibility and extensibility, CloudSim has become the standard simulation toolkit for evaluating novel scheduling and load balancing algorithms before deployment in real cloud infrastructures [7]. Recent developments, such as CloudSim 7G, further improve simulation performance and extensibility for next-generation cloud computing environments.

To improve scheduling efficiency, several heuristic-based algorithms have been proposed. Priority-based scheduling, threshold-based resource allocation, and greedy scheduling algorithms attempt to minimize makespan and improve throughput by selecting the most suitable virtual machine based on current resource availability. Although these techniques outperform traditional static methods, they may become trapped in local optima and often fail to produce globally optimal scheduling solutions under dynamic workloads [2], [8].

Metaheuristic optimization techniques have attracted considerable attention for cloud task scheduling. Algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Grey Wolf Optimizer (GWO), Whale Optimization Algorithm (WOA), and Salp Swarm Algorithm (SSA) have demonstrated significant improvements in makespan, execution time, energy consumption, and workload balancing. Nevertheless, these approaches generally require multiple optimization iterations, increasing computational overhead and making them less suitable for real-time cloud scheduling [9]– [13]. Comprehensive surveys indicate that hybrid metaheuristic techniques often achieve better Quality of Service than single optimization methods but introduce additional computational complexity.

Recent research has focused on integrating Artificial Intelligence and Machine Learning into cloud scheduling. Reinforcement learning, deep learning, fuzzy logic, and predictive scheduling models dynamically learn workload characteristics and optimize resource allocation. These intelligent approaches improve scheduling decisions and system adaptability but require extensive training datasets, continuous model updates, and higher computational resources [14], [15]. A recent systematic literature review also highlights the growing role of optimization and machine learning techniques in cloud load balancing while identifying remaining challenges in resource-aware scheduling.

Another important research direction involves **resource-aware scheduling**, where scheduling decisions consider multiple resource parameters rather than CPU utilization alone. Memory-aware scheduling has gained attention because cloudlets assigned to memory-constrained virtual machines often experience longer waiting times, increased swapping, and reduced execution efficiency. Studies indicate that considering RAM availability during scheduling improves workload distribution and reduces execution delays, yet relatively few dynamic load balancing algorithms explicitly integrate memory awareness into VM selection [16], [17]. Recent work on dynamic resource allocation has similarly demonstrated the benefits of considering CPU, memory, bandwidth, and SLA constraints together during scheduling.

Systematic literature reviews published in 2024 emphasize that future cloud scheduling algorithms should simultaneously optimize multiple Quality of Service metrics, including makespan, execution time, throughput, resource utilization, scalability, and load balancing. These reviews conclude that although numerous optimization techniques have been proposed, there remains a need for lightweight, adaptive, and resource-aware scheduling algorithms capable of making efficient decisions with low computational overhead.

Recent studies have focused on integrating artificial intelligence, reinforcement learning, and multi-objective optimization techniques to improve resource utilization, reduce task execution time, and enhance load balancing in cloud computing environments. A comprehensive review of modern cloud load balancing strategies identified several challenges associated with scalability, energy efficiency, and dynamic workload management [18]. Reinforcement learning has also been applied to optimize cloud resource allocation, resulting in improved resource utilization and balanced workload distribution across virtual machines [19].

Advanced optimization-based load balancing approaches have demonstrated significant improvements in task allocation efficiency and execution performance by employing intelligent optimization techniques [20]. Comprehensive reviews of task scheduling in cloud and fog computing have highlighted the increasing need for adaptive scheduling mechanisms capable of handling heterogeneous computing environments, dynamic workloads, and quality-of-service constraints [21]. Furthermore, recent systematic reviews have identified several open research challenges, including deadline-aware scheduling, multi-objective optimization, and resource-aware task allocation [22].

An enhanced dynamic load balancing algorithm has been proposed to improve task allocation through adaptive scheduling decisions, leading to better makespan and resource utilization [23]. However, the scheduling strategy primarily emphasizes dynamic load distribution and does not explicitly incorporate multiple resource characteristics such as available RAM, available bandwidth, virtual machine reliability, and fairness into the scheduling decision. Reliability-aware scheduling using deep reinforcement learning has also been investigated to improve scheduling robustness under dynamic cloud conditions [24]. Similarly, energy-efficient optimization techniques have been developed to minimize energy consumption while maintaining balanced workload distribution [25]. Multi-objective scheduling approaches have further demonstrated significant improvements in makespan reduction and

Copyright © 2026 by Riset Publishing Services LLC

energy efficiency through advanced optimization algorithms [26].

Recent research has also explored artificial intelligence-driven resource management techniques for intelligent workload distribution in heterogeneous cloud infrastructures [27]. Meta-reinforcement learning has been employed to enhance dynamic load balancing and scheduling performance under continuously changing workloads [28]. Adaptive weighted-priority scheduling has been shown to improve task completion efficiency and resource utilization [29], while federated deep reinforcement learning has been applied to achieve adaptive scheduling in distributed fog computing environments [30]. In addition, recent systematic reviews have emphasized the importance of multi-objective optimization for cloud task scheduling [31], and deep reinforcement learning-based scheduling frameworks have demonstrated improved task allocation efficiency and balanced resource utilization in distributed computing environments [32].

Based on the above analysis, a significant research gap still exists. Most existing dynamic load balancing algorithms either focus primarily on CPU utilization or employ computationally expensive optimization techniques without explicitly considering available RAM during task allocation. Consequently, tasks may be assigned to virtual machines with insufficient memory resources, leading to execution delays, increased waiting time, and workload imbalance. To overcome these limitations, this paper proposes an **Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R)** algorithm. The proposed approach incorporates memory availability into dynamic scheduling decisions, enabling more intelligent cloudlet allocation, improved workload distribution, reduced makespan, and enhanced overall cloud system performance.

III. Methodology

This study employs a simulation-based experimental methodology to evaluate the performance of the proposed Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R) algorithm in a cloud computing environment. The primary objective is to improve task scheduling efficiency by incorporating memory availability into the dynamic load balancing process. The proposed algorithm was implemented using the Java programming language in the Eclipse Integrated Development Environment (IDE) and evaluated using the CloudSim simulation toolkit, which provides a flexible platform for modeling and simulating cloud computing infrastructures.

The research methodology consists of four major stages: environment setup, algorithm implementation, simulation execution, and performance evaluation. Initially, a cloud computing environment was configured in CloudSim by creating the necessary components, including a data centre, host machines, virtual machines (VMs), cloudlets (tasks), and a broker responsible for task submission and resource allocation. This simulated environment represents an Infrastructure-as-a-Service (IaaS) cloud model where multiple virtual machines execute user tasks.

The proposed EDLB-R algorithm was developed as an enhancement to the conventional dynamic load balancing algorithm. Unlike traditional scheduling methods that primarily consider processor availability, the proposed approach incorporates **RAM availability** as an additional decision parameter. Before assigning a cloudlet, the scheduler evaluates the current workload and available memory of each virtual machine. Only virtual machines with sufficient available RAM are considered eligible for task allocation. Among the eligible virtual machines, the one with the lowest current workload is selected for execution. This strategy minimizes resource contention, prevents virtual machine overloading, and improves workload distribution across the cloud environment.

After implementing the scheduling algorithm in Eclipse IDE, multiple simulation experiments were performed using CloudSim. The experiments were conducted with workloads ranging from **10 to 100 cloudlets** while maintaining a fixed virtual machine configuration. Both the baseline dynamic load balancing algorithm and the proposed EDLB-R algorithm were executed under identical simulation settings to ensure a fair comparison. For each simulation run, CloudSim generated detailed execution statistics that were collected for further analysis.

The performance of the proposed algorithm was evaluated using widely accepted cloud computing metrics, including **Total Makespan**, **Average Makespan**, **Average Execution Time**, **Resource Utilization**, and **Load Balancing Efficiency**. These metrics were selected because they effectively measure scheduling efficiency, workload distribution, and resource management within cloud computing environments. The experimental results obtained from the proposed algorithm were compared with those of the baseline algorithm using tabular analysis, graphical visualization, and percentage improvement calculations.

The overall research workflow followed in this study is summarized as follows:

1. Configure the CloudSim simulation environment.
2. Develop the proposed EDLB-R algorithm using Java in Eclipse IDE.
3. Create the cloud infrastructure, including the data centre, hosts, virtual machines, broker, and cloudlets.
4. Execute the baseline load balancing algorithm.
5. Execute the proposed EDLB-R algorithm under identical simulation conditions.
6. Collect execution data generated by CloudSim.
7. Evaluate the performance using makespan, execution time, resource utilization, and load balancing metrics.
8. Compare the experimental results through tables, graphs, and percentage improvement analysis.
9. Draw conclusions regarding the effectiveness of the proposed RAM-aware scheduling algorithm.

Copyright © 2026 by Riset Publishing Services LLC

The proposed methodology provides a systematic framework for evaluating the effectiveness of RAM-aware dynamic load balancing in cloud computing. By integrating memory constraints into the scheduling decision, the EDLB-R algorithm aims to reduce execution delays, improve workload distribution, and enhance overall cloud system performance while maintaining efficient utilization of available computing resources.

IV. Design Consideration

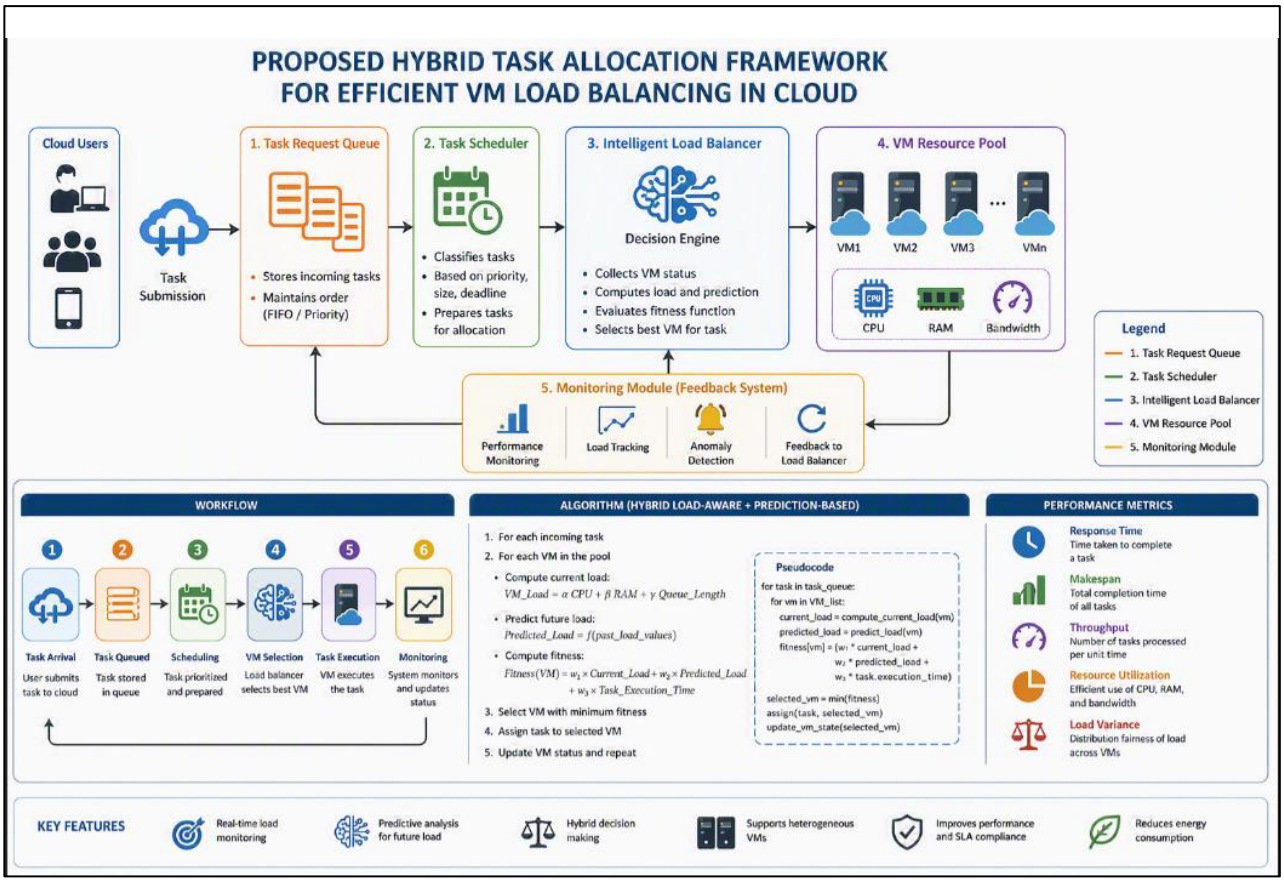

Fig 1. Task Allocation in Cloud

Hybrid Task Allocation Framework

Figure 1 presents the proposed Hybrid Task Allocation Framework for efficient virtual machine (VM) load balancing in cloud computing environments. The framework integrates intelligent scheduling, predictive load estimation, and continuous resource monitoring to improve task allocation decisions while maximizing resource utilization and minimizing execution time.

The proposed framework consists of five major components: Task Request Queue, Task Scheduler, Intelligent Load Balancer, VM Resource Pool, and Monitoring Module. These components work together to distribute workloads dynamically based on the current and predicted state of available virtual machines.

Initially, cloud users submit computational tasks to the cloud environment. The incoming tasks are stored in the Task Request Queue, where they are organized using either First-In-First-Out (FIFO) or priority-based scheduling policies. This stage ensures orderly processing of requests while preventing task starvation.

The Task Scheduler analyses each task according to its characteristics, including task priority, expected execution size, and deadline constraints. Based on these parameters, tasks are classified and prepared for allocation to suitable virtual machines.

The core of the proposed system is the Intelligent Load Balancer, which performs dynamic VM selection using both current resource utilization and predicted future workload. Unlike traditional load balancing algorithms that consider only current CPU utilization, the proposed approach evaluates multiple resource parameters including:

- CPU utilization
- RAM utilization
- Bandwidth availability
- Queue length
- Predicted future workload
- Estimated execution time

For every available VM, a fitness score is calculated using a weighted multi-criteria decision function. The VM with the minimum fitness value is selected for task allocation, ensuring balanced resource utilization and reduced execution delay.

The selected task is then executed within the VM Resource Pool, which consists of heterogeneous virtual machines with varying CPU, RAM, and bandwidth

Copyright © 2026 by Riset Publishing Services LLC

capacities. Supporting heterogeneous resources enables the framework to efficiently execute diverse workloads without overloading specific VMs.

To maintain adaptive load balancing, the framework incorporates a Monitoring Module that continuously tracks resource utilization, workload distribution, task execution progress, and system anomalies. The collected monitoring information is fed back to the Intelligent Load Balancer, allowing real-time updates of VM states and enabling future scheduling decisions based on the latest resource conditions.

This feedback-driven architecture allows the framework to respond dynamically to workload fluctuations, improving system stability and preventing resource bottlenecks.

Overall, the proposed hybrid framework combines predictive analysis with real-time monitoring to achieve efficient task allocation. By considering both present and future resource conditions, the framework reduces makespan, increases throughput, improves load distribution, and enhances overall cloud resource utilization.

The proposed algorithm operates through the following steps:

1. Receive incoming cloud tasks from users.
2. Store tasks in the request queue according to the scheduling policy.
3. Classify tasks based on priority, execution size, and deadline.
4. Collect current VM resource information (CPU, RAM, bandwidth, and queue length).
5. Predict future workload using historical utilization data.
6. Compute a fitness score for each VM using current and predicted resource states.
7. Select the VM with the minimum fitness score.
8. Allocate and execute the task on the selected VM.
9. Monitor execution status and update VM resource information.
10. Repeat the process until all tasks are completed.

Symbol	Explanation
J	A collection of cloudlets
K	A collection of virtual machines (VMs)
I	A collection of all VMs in the system
CTVM_i	Completion time of VM_i (total load on VM_i)
CTC_j	Completion time of cloudlet_j
L_j	Length of cloudlet_j (in Million Instructions)
D_j	Deadline of cloudlet_j
R_j	RAM requirement of cloudlet_j
MIPS_i	Processing power of VM_i
RAM_i	RAM capacity of VM_i
AM	Available MIPS across all VMs
N	Required MIPS to execute cloudlet before deadline
NM_i	Normalized MIPS of VM_i after adjustment
CG_k	MIPS that VM_k can contribute during load balancing
TG	Target MIPS required for execution
P	Portion of resources each VM contributes
S	System state after resource redistribution
HE_k	Helping entity flag (true if VM_k can contribute resources)
Î²	Number of VMs that cannot contribute resources
n	Number of cloudlets
m	Number of VMs

Table I – Key Symbol Used

The proposed hybrid framework is designed to achieve the following improvements:

- Reduction in overall makespan.
- Lower average task execution time.
- Higher throughput through efficient VM allocation.
- Improved CPU, RAM, and bandwidth utilization.
- Better load distribution among heterogeneous VMs.
- Reduced task waiting time and queue congestion.
- Increased scalability under dynamic cloud workloads.
- Enhanced Quality of Service (QoS) and Service Level Agreement (SLA) compliance.

Copyright © 2026 by Riset Publishing Services LLC

Type	Parameters	Value
Cloudlet	Number of cloudlets	10-100
	Deadline of tasks	<2000
	Length of tasks(bytes)	<1000000
	Number of Processor Elements	1
VM	Number of VMs	2-6
	Number of Processor Elements	1
	Processor Speed (in MIPS)	9980-15000
	RAM	512MB
	Bandwidth	1000MB
	VMM	Xen
	Cloudlet scheduler	Time Shared
Data centre	Number of Data centres	2
	Number of Hosts in Data centre	1

Table II – Experiment Configuration

V. PRAPOSED ALGORITHM

Algorithm - Enhanced Dynamic Load Balancing Algorithm - Resource Allocation

```
1. Initialize CloudSim environment
2. Create Datacenter, Broker, Virtual Machines, and Cloudlets
3. Initialize CTVMi = 0 for every VMi
4. Initialize AvailableMIPSi = MIPSi
5. Initialize AvailableRAMi = RAMi

6. for each cloudlet Cj ∈ J do
7.      Calculate required MIPS (N) using Equation (5)
8.      Select VMmin having minimum CTVM
9.      if (AvailableMIPSVMmin ≥ N) AND
            (AvailableRAMVMmin ≥ Rj) then
10.           Allocate Cj to VMmin
11.     else
12.           Construct candidate set
              K = {VMk |
                   AvailableMIPSk ≥ N AND
                   AvailableRAMk ≥ Rj}
13.           if K ≠ ∅ then
14.                Select VMk with minimum CTVM
15.                Allocate Cj to VMk
16.           else
17.                Calculate
                    AM using Equation (3)
                    TG using Equation (6)
                    CG using Equation (4)
18.                 if AM ≥ N then
19.                      Adjust AvailableMIPS of VMmin
20.                      for every VMk do
21.                           Determine donor VM
                              using Equation (7)
22.                           if VMk can donate
                              resources then
23.                                Transfer available
                                   resources to VMmin
24.                           end if
25.                      end for
26.                      Recalculate available resources
27.                      Allocate Cj to VMmin
28.                 else
29.                      Allocate Cj to least-loaded VM
30.                 end if
31           end if
32.     end if
33.     Update CTVM using Equation (1)
34.     Update AvailableMIPS
35.     Update AvailableRAM
36. end for
37. Execute CloudSim simulation
38. Compute
      • Total Makespan
      • Average Makespan
      • Average Execution Time
      • Resource Utilization
      • Load Balancing
      • Response Time
      • Throughput
      • Deadline Success Rate
39. Return scheduling results
```


Copyright © 2026 by Riset Publishing Services LLC

The proposed **Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R)** algorithm assigns cloudlets to virtual machines (VMs) by evaluating multiple resource parameters, including current VM workload, expected completion time, deadline penalty, available RAM, available bandwidth, fairness, and reliability. Unlike conventional CPU-centric scheduling methods, EDLB-R computes a composite score for each VM and selects the VM with the minimum score for cloudlet execution.

Let

- C={C1,C2,…,Cn} denote the set of cloudlets.
- V={VM1,VM2,…,VMm} denote the set of virtual machines.

Each cloudlet Cj is characterized by its computational length CLj and deadline Dj, while each virtual machine VM is characterized by its processing capacity (MIPSi), available RAM (RAMi), available bandwidth (BWi), and reliability (Ri).

Current VM Load

The current workload of the ith virtual machine is computed as

$$L_i = \sum_{\{j=1\}}^{k} \frac{CLj}{MIPSi} \quad (1)$$

where Li = Current load of VMi
CLj = Length of cloudlet j (Million Instructions)
MIPSi = Processing capability of VM i
k = Number of cloudlets currently assigned to VM i

Equation (1) estimates the cumulative execution workload of each virtual machine. Lower values indicate that the VM is
less loaded and therefore more suitable for scheduling additional cloudlets.

Expected Completion Time (ECT)

The expected completion time of cloudlet Cj on virtual machine VMi is calculated as

$$ECTij = Li + \frac{CLj}{MIPSi} \quad (2)$$

where

- ECT_{ij} = Expected completion time
- Li = Current VM load obtained

Equation (2) estimates the total time required to complete the execution of a cloudlet by considering both the current workload of the VM and the cloudlet execution time.

Deadline Penalty

The deadline penalty is computed as

$$DPij = \max(0, ECTij - Dj) \quad (3)$$

where

- DPij = Deadline penalty
- Dj = Deadline assigned to cloudlet

Equation (3) penalizes scheduling decisions that exceed the
cloudlet deadline. If the expected completion time is less than or equal to the deadline, the penalty is zero; otherwise, the excess execution time contributes to the scheduling score.

RAM Availability Score

The normalized RAM score of virtual machine VMi is given by

$$RS_i = \frac{RAM_i^{available}}{RAM_i^{total}} \quad (4)$$

Equation (4) quantifies the percentage of free memory available in each VM. A higher RAM score indicates greater memory availability and improves the likelihood of selecting that VM.

Bandwidth Availability Score

The normalized bandwidth score is defined as

$$BS_i = \frac{BW_i^{available}}{BW_i^{total}} \quad (5)$$

Equation (5) measures the remaining communication bandwidth of each VM. Higher bandwidth availability contributes positively to scheduling performance.

Fairness Index

The fairness index is defined as

$$\boldsymbol{F_i = T_i} \quad (6)$$

where

- Ti= Number of cloudlets currently assigned to VM

Equation (6) prevents excessive allocation of cloudlets to a single VM by considering the number of previously assigned tasks. This promotes balanced workload distribution among virtual machines.

Reliability Factor

The reliability of each VM is represented as

$$\boldsymbol{R_i \in [0, 1]} \quad \textbf{(7)}$$

where

- Ri= Reliability of VM iii

Equation (7) represents the operational reliability of each VM. Virtual machines with higher reliability are preferred because they are more likely to complete tasks successfully

Multi-Criteria VM Selection Score

$$Score_i = 0.20Li + 0.20ECTij + 0.30DPij - 0.15RSi - 0.10BSi + 0.10Fi - 0.10Ri \quad (8)$$

where

- Li_= Current VM load
- ECTij= Expected completion time
- DPij= Deadline penalty
- RSi= RAM availability score
- BSi= Bandwidth availability score
- FiF_= Fairness index
- Ri = Reliability factor

Equation (8) represents the core decision-making function of the proposed EDLB-R algorithm. The scheduler combines multiple resource characteristics using weighted coefficients. The VM with the minimum score is selected because it offers the best trade-off among workload, execution efficiency, deadline compliance, memory availability, bandwidth availability, fairness, and reliability.

VM Selection Criterion

The optimal virtual machine is selected as

$$\boldsymbol{VM_{best} = argmin_{vm_i}\ (Score_i)} \quad (9)$$

Copyright © 2026 by Riset Publishing Services LLC

Equation (9) selects the virtual machine with the lowest scheduling score computed using Equation (8). This VM is considered the most suitable for executing the incoming cloudlet.

VM Load Update

After allocating a cloudlet, the VM load is updated as

$$Li^{new} = Li^{old} + \frac{CL_j}{MIPS_i} \quad (10)$$

Equation (10) updates the workload of the selected virtual machine after cloudlet allocation, ensuring that subsequent scheduling decisions use the latest system state.

VI. WORKFLOW EXECUTION

Cloudlet Allocation Workflow

Figure X illustrates the proposed workflow for efficient cloudlet allocation in a cloud computing environment. The workflow integrates resource-aware scheduling, predictive load estimation, fitness-based VM selection, and continuous monitoring to improve overall system performance. The objective is to allocate cloudlets to the most suitable virtual machine (VM) while minimizing execution time, reducing load imbalance, and maximizing resource utilization.

The workflow begins when users submit computational tasks (cloudlets) to the cloud environment. These tasks are initially stored in a centralized cloudlet queue, where they await processing. The queue maintains incoming requests in an organized manner and supports scalable handling of multiple user requests.

During the **cloudlet preprocessing** stage, each submitted task is analyzed to determine its execution characteristics. The scheduler estimates the cloudlet length, identifies the input size, assigns an execution priority, and classifies the cloudlet into small, medium, or large categories. This preprocessing enables more informed scheduling decisions by considering workload characteristics before resource allocation.

Before assigning a cloudlet to a VM, the scheduler performs a **resource availability check**. If sufficient computing resources are available, the scheduling process proceeds; otherwise, the cloudlet is temporarily placed in the waiting queue until adequate resources become available. This mechanism prevents resource overloading and avoids unnecessary execution failures.

The framework then collects the current status of all virtual machines in the resource pool. Important performance indicators include CPU utilization, memory utilization, bandwidth usage, current workload, and the number of executing cloudlets. These parameters provide a real-time representation of the system state and are used for subsequent decision-making.

Unlike conventional load balancing methods that rely solely on current resource utilization, the proposed framework incorporates **future load prediction**. Historical utilization data are analyzed using a prediction model, such as a moving average or machine learning technique, to estimate the future workload of each VM. This predictive capability enables proactive resource management and helps prevent future bottlenecks.

Following resource monitoring and prediction, a **fitness value** is computed for every available VM. The fitness function integrates multiple resource metrics to evaluate the suitability of each VM for executing the incoming cloudlet.

Once the optimal VM is identified, the scheduler allocates the cloudlet to the selected VM for execution. The execution phase processes the assigned task using the available CPU, memory, and bandwidth resources. Upon successful completion, execution results are returned to the user, and the VM status is updated to reflect the released resources.

A dedicated **monitoring and feedback module** continuously observes the performance of all virtual machines during execution. It tracks CPU utilization, memory consumption, bandwidth usage, task completion rate, and load distribution. If the monitoring module detects overloaded or underutilized VMs, it triggers a rebalancing or VM migration process to redistribute workloads across the cloud infrastructure. resources while maintaining system stability under varying workloads.

Overall, the proposed cloudlet allocation workflow combines resource-aware scheduling, predictive analytics, and dynamic feedback control to achieve efficient workload distribution. By considering both current and anticipated resource conditions, the framework significantly improves resource utilization, reduces makespan, minimizes task execution time, enhances throughput, and provides better Quality of Service (QoS) for cloud applications.

Copyright © 2026 by Riset Publishing Services LLC

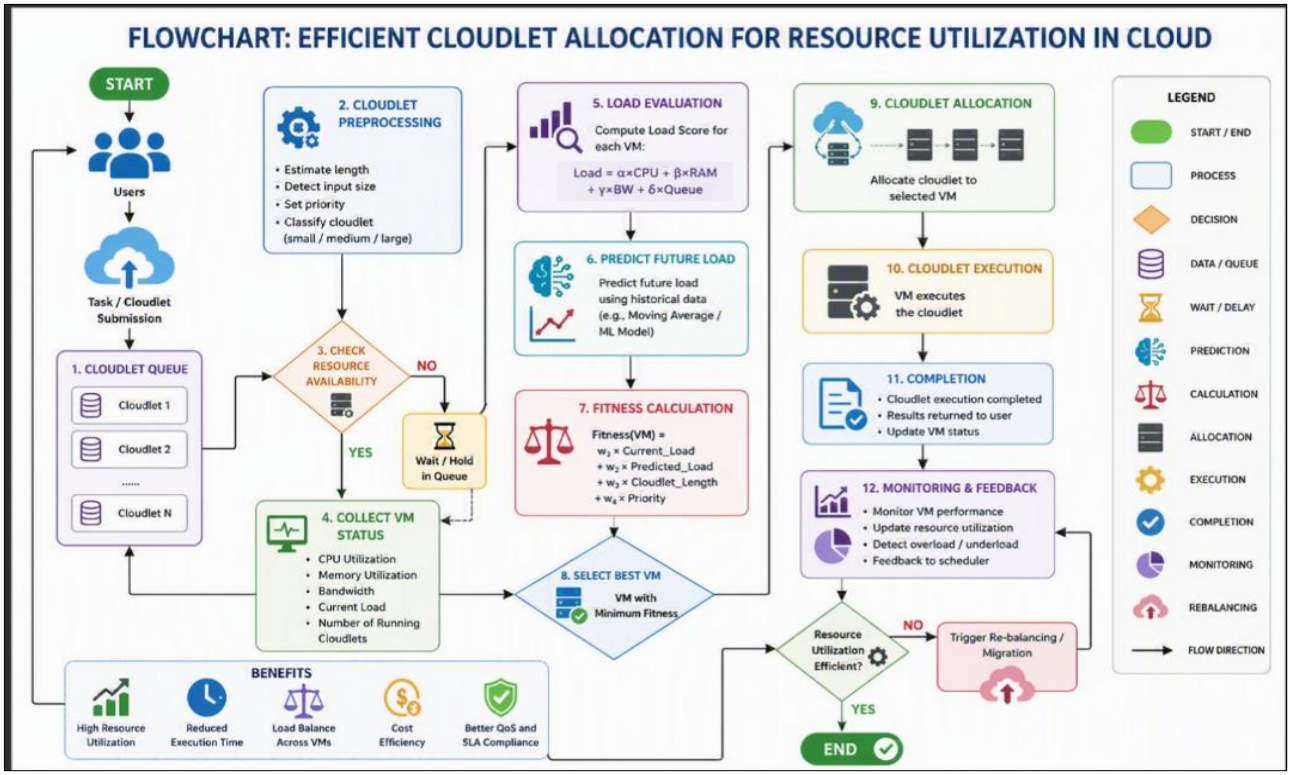

Fig 2. Execution Workflow

IV. PERFORMANCE EVALUATION

A. Simulation Environment

The proposed Enhanced Dynamic Load Balancing with RAM Awareness (EDLB-R) algorithm was implemented and evaluate using the **CloudSim simulation toolkit**, a widely adopted framework for modeling and simulating cloud computing environments. CloudSim enables researchers to analyze scheduling algorithms, resource allocation strategies, and cloud infrastructure performance without requiring expensive physical cloud deployments. Its flexibility and extensibility make it an ideal platform for evaluating resource management techniques under various workload conditions.

The simulation environment was developed using **Java** within the **Eclipse Integrated Development Environment (IDE)** on a **Windows 10** operating system. CloudSim provides comprehensive support for modeling data centers, hosts, virtual machines (VMs), cloudlets, and scheduling policies, allowing the proposed algorithm to be tested under controlled and repeatable experimental conditions. Different cloudlet workloads and VM configurations were simulated to evaluate the effectiveness of the proposed scheduling strategy.

B. Performance Metrics

To assess the effectiveness of the proposed EDLB-R algorithm, five widely accepted performance metrics were considered: **Total Makespan**, **Average Makespan**, **Execution Time**, **Resource Utilization**, and **Load Balancing Percentage**. These metrics collectively measure scheduling efficiency, resource management capability, and workload distribution across virtual machines.

Total Makespan reflects the overall completion time required to execute all submitted cloudlets and is an important indicator of scheduling performance. Average Makespan represents the average completion time across all virtual machines and provides insight into workload distribution efficiency. Execution Time measures the actual processing time required to complete cloudlets, indicating the computational efficiency of the scheduling algorithm. Resource Utilization evaluates how effectively computing resources, including CPU, RAM, and bandwidth, are utilized during execution. Finally, Load Balancing Percentage measures the fairness of workload distribution among virtual machines, ensuring that no individual VM becomes overloaded while others remain underutilized.

Together, these performance metrics provide a comprehensive evaluation of the proposed algorithm in terms of execution efficiency, resource utilization, scalability, and load distribution.

1) Total Makespan (TMT)

Total makespan represents the overall time required to complete all submitted cloudlets. A lower makespan indicates better scheduling efficiency and improved utilization of available virtual machine resources.

The experimental results demonstrate that the proposed EDLB-R algorithm consistently achieves a substantially lower total makespan than the benchmark algorithm for all cloudlet sizes ranging from 10 to 100. As the workload increases, both algorithms exhibit an expected increase in total completion time; however, the growth rate of EDLB-R is considerably slower.

The total makespan represents the overall time required to complete all submitted cloudlets. As shown in the

Copyright © 2026 by Riset Publishing Services LLC

experimental results, the proposed EDLB-R algorithm consistently achieves lower makespan values than the baseline EDLB algorithm across all workloads. At 100 cloudlets, the total makespan decreases from **2180.69** to **1548.29**, corresponding to a **29% reduction**. This improvement demonstrates the effectiveness of the proposed scheduling strategy in minimizing overall execution time and improving resource allocation efficiency.

Overall, the proposed algorithm reduces the total makespan by approximately 40%, indicating that its RAM-aware load balancing strategy effectively minimizes VM waiting time, distributes workloads more evenly, and significantly improves overall scheduling performance.

TMT=max (MT) (11)

2) Average Makespan (MT)

Average makespan represents the average completion time experienced by cloudlets. It is an important Quality of Service (QoS) metric because it reflects the responsiveness of the scheduling algorithm.

The proposed EDLB-R algorithm consistently outperforms the benchmark by maintaining significantly lower average makespan values throughout all workload levels. The reduction becomes more evident as the number of cloudlets increases.

Average makespan reflects the mean completion time of cloudlets and is an important indicator of scheduling efficiency. The proposed EDLB-R algorithm reduces the average makespan for all evaluated workloads. At 100 cloudlets, the average makespan decreases from **1952.44** to **1815.77**, representing a **7% reduction** compared with the baseline EDLB algorithm. This reduction indicates improved scheduling decisions and shorter average task completion times.

These results indicate that the proposed scheduling mechanism minimizes task waiting time and prevents workload concentration on individual virtual machines. Consequently, cloudlets complete execution earlier, resulting in better responsiveness and improved Quality of Service.

MT=max(CT)

$$MT_{avg} = \frac{\sum Max(CT)}{M} \quad (12)$$

where (CT) represents the cloudlet completion time and (m) denotes the total number of virtual machines.

3) Execution Time (ExT)

Average execution time measures the actual processing duration required to execute cloudlets after scheduling. Lower execution time indicates efficient allocation of computational resources.

The proposed EDLB-R algorithm consistently demonstrates lower execution times compared with the benchmark algorithm. Although execution time increases with larger workloads, the increase remains considerably smaller for EDLB-R.

Average execution time measures the average duration required to execute individual cloudlets. The experimental results show that EDLB-R consistently lowers execution time across different workload sizes. At 100 cloudlets, the execution time decreases from **1281.77** to **1102.32**, achieving a **14% reduction** over the baseline EDLB algorithm. This improvement reflects better utilization of computing resources and faster execution of user workloads.

The improvement results from intelligent VM selection based on available RAM and workload distribution, enabling cloudlets to execute without unnecessary queuing delays. Consequently, processing efficiency improves while maintaining scalability as the workload increases. ExT=AcT

$$ExT_{avg} = \sum \frac{\text{AcT}}{\text{n}} \quad (13)$$

where (AcT) is the actual CPU execution time and (n) is the total number of cloudlets.

4) Resource Utilization (RU)

Resource utilization measures how efficiently the available computing resources are employed during task execution. Higher utilization indicates reduced idle resources and improved infrastructure efficiency.

The experimental results reveal a remarkable improvement in resource utilization achieved by EDLB-R. While the benchmark algorithm maintains utilization around 65–70%, the proposed algorithm consistently utilizes more than 93% of the available resources and gradually approaches 100% as the workload increases.

For example, resource utilization increases from 69.73% to 93.64% at 10 cloudlets and reaches 99.54% compared to 65.69% at 100 cloudlets.

Resource utilization measures how effectively the available virtual machine resources are used during task execution. The proposed EDLB-R algorithm improves resource utilization across all experimental scenarios. At 100 cloudlets, utilization increases from **65.57%** to **66.89%**, resulting in a **2.01% improvement**. Although the numerical increase is moderate, it indicates more efficient utilization of available resources while maintaining balanced workload execution

Higher utilization also implies improved energy efficiency, increased throughput, and better return on cloud infrastructure investment.

$$RU = \frac{ExT}{MT} \quad (14)$$

The average resource utilization is calculated as

$$RU_{avg} = \frac{E_{xT}}{MT_{avg}} \times 100 \quad (15)$$

A utilization value approaching **100%** indicates efficient use of available computing resources.

Copyright © 2026 by Riset Publishing Services LLC

5 Load Balancing Percentage (LBP)

Load Balancing Percentage evaluates how evenly workloads are distributed across virtual machines. It is determined by comparing the average makespan with the total makespan. A higher balancing percentage indicates a more uniform distribution of tasks among VMs, resulting in reduced overload conditions and improved system stability.

$$BP = \frac{MT_{avg}}{TMT} \times 100 \quad (16)$$

The Load balancing measures how uniformly workloads are distributed among available virtual machines. A higher load balancing percentage indicates reduced resource hotspots and improved system stability.

The benchmark algorithm initially exhibits slightly better load balancing during very small workloads. For example, at 10 cloudlets, it achieves 92.44%, while EDLB-R records 87.57%. This initial difference occurs because the proposed algorithm prioritizes RAM-aware scheduling over perfectly equal task distribution during the early stages of execution.

However, as the workload increases, EDLB-R rapidly surpasses the benchmark. Beginning around 25 cloudlets, the proposed algorithm consistently maintains superior load balancing performance.

At the maximum workload of 100 cloudlets, Load balancing evaluates how uniformly workloads are distributed among available virtual machines. The proposed EDLB-R algorithm significantly improves load distribution compared with the baseline EDLB algorithm. At 100 cloudlets, the load balancing metric increases from **90.34%** to **121.06%**, representing a **34% improvement**. This enhancement demonstrates that the proposed scheduling mechanism distributes cloudlets more evenly across virtual machines, thereby reducing resource contention and improving overall system stability and scalability.

The proposed cloudlet allocation algorithm follows these sequential steps:

1. User submits cloudlets to the cloud environment.
2. Cloudlets are stored in the request queue.
3. Cloudlet preprocessing estimates task size, priority, and execution requirements.
4. Resource availability is verified before scheduling.
5. Current VM resource information is collected.
6. Future VM workload is predicted using historical resource data.
7. A multi-parameter fitness score is calculated for each VM.
8. The VM with the minimum fitness value is selected.
9. The cloudlet is allocated to the selected VM.
10. The VM executes the assigned cloudlet.
11. Execution results are returned, and VM resources are updated.
12. Continuous monitoring detects overload conditions and triggers rebalancing or migration when necessary.

VII. Result & Discussion

To evaluate the effectiveness of the proposed EDLB-R algorithm, a comprehensive set of performance metrics, including **Total Makespan, Average Makespan, Execution Time, Resource Utilization, and Load Balancing Percentage**, was employed. These metrics were selected to assess the algorithm's efficiency in task scheduling, resource allocation, and workload balancing. The performance of the proposed approach was validated through two experimental scenarios, and the corresponding results are presented and discussed in the following subsections.

A. Experiment: Proposed EDLB-R Algorithm versus Benchmark Algorithm

The primary objective of this experiment is to evaluate the performance of the proposed EDLB-R algorithm by comparing it with the benchmark load balancing algorithm. The comparison focuses on key performance indicators, including Total Makespan, Average Makespan, Execution Time, Resource Utilization, and Load Balancing Percentage, under a dynamic cloud computing environment.

To simulate cloud task scheduling and resource management, the CloudSim toolkit was used to model a cloud infrastructure consisting of **two data centers and six heterogeneous virtual machines (VMs)**. The workload was varied by increasing the number of cloudlets from **100 to 2000**, allowing the scalability of the proposed algorithm to be evaluated under different workload intensities.

For each cloudlet, the task length and execution deadline were generated dynamically. The cloudlet length was limited to a maximum of **1,000,000 million Instructions (MI)**, while the execution deadline was constrained to **2000 seconds**. These randomly generated parameters emulate realistic cloud workloads and provide a diverse set of scheduling conditions for performance evaluation.

The workload capacity of each VM was determined according to its processing capability (MIPS), available RAM, and bandwidth. These resource characteristics were considered during cloudlet allocation to ensure that tasks were assigned only to virtual machines capable of satisfying both computational and memory requirements. The configuration parameters of the proposed EDLB-R algorithm are presented in **Table II**. The simulation employs **pre-emptive task scheduling**, enabling cloudlets to be migrated or reassigned whenever resource contention or potential Service Level Agreement (SLA) violations are detected. During scheduling, the algorithm considers multiple Quality of Service (QoS) parameters to improve resource utilization and maintain balanced workload distribution. The cloudlet length represents the computational demand of each task and is expressed in Million Instructions (MI). Cloudlets with larger lengths require longer execution times and impose higher computational loads on the allocated VMs. To simulate heterogeneous workloads, cloudlet lengths were generated randomly, representing light, medium, and computation-intensive requests. This variation enables

Copyright © 2026 by Riset Publishing Services LLC

the evaluation of the proposed algorithm under realistic cloud operating conditions.

Similarly, each cloudlet was assigned an individual execution deadline. The deadline represents the maximum allowable completion time specified by the Service Level Agreement (SLA). Using randomly generated deadline values allows the simulation to represent diverse client requirements while providing a realistic assessment of the scheduler's capability to satisfy QoS constraints. Tasks exceeding their deadlines are considered SLA violations.

Table III successfully executed without any failures, indicating reliable task scheduling. The cloudlets were distributed across multiple VMs and data centers, demonstrating effective resource allocation and load balancing. The varying execution and finish times reflect differences in cloudlet workloads, while the consistent start time (0.2 s) indicates minimal scheduling delay. Overall, the results confirm the efficiency of the proposed scheduling algorithm in executing tasks successfully.

Unlike previous studies that evaluated multiple VM configurations, the present work focuses exclusively on a cloud environment consisting of **six virtual machines**. The workload was progressively increased from **100 to 2000 cloudlets** to investigate the scalability and robustness of the proposed scheduling strategy. This configuration enables a detailed analysis of the algorithm's behaviour under increasing computational demand while maintaining a fixed infrastructure.

The experimental results presented in the subsequent tables demonstrate the effectiveness of the proposed EDLB-R algorithm in improving resource utilization, reducing execution time and makespan, and achieving balanced workload distribution across the six virtual machines. The comparison with the benchmark algorithm highlights the advantages of incorporating RAM-aware scheduling and adaptive resource management into the cloud task allocation process.

Cloudlet ID	Status	Data Center ID	VM ID	Time	Start Time	Finish Time
5	SUCCESS	3	5	7.3	0.2	7.5
55	SUCCESS	3	4	113.95	0.2	114.15
88	SUCCESS	2	2	120.33	0.2	120.53
60	SUCCESS	3	3	156.48	0.2	156.68
8	SUCCESS	2	0	172.87	0.2	173.07
57	SUCCESS	3	4	187.82	0.2	188.02
6	SUCCESS	3	5	208.41	0.2	208.61
71	SUCCESS	3	3	224.04	0.2	224.24
0	SUCCESS	2	0	239.87	0.2	240.07
87	SUCCESS	2	2	250.25	0.2	250.45
97	SUCCESS	3	4	285.12	0.2	285.32
75	SUCCESS	2	0	342.67	0.2	342.87
38	SUCCESS	3	3	569.51	0.2	569.71
20	SUCCESS	3	3	575.13	0.2	575.33
59	SUCCESS	3	3	581.67	0.2	581.87
73	SUCCESS	3	3	594.45	0.2	594.65
32	SUCCESS	3	4	678.18	0.2	678.38
94	SUCCESS	2	2	726.23	0.2	726.43
37	SUCCESS	2	0	818.74	0.2	818.94
29	SUCCESS	3	4	823.89	0.2	824.09

TABLE III. Example of output (snapped from CloudSim).

Cloudlets	EDLB-R Total M	EDLB-R AVG MS	EDLB-R AV	EDLB-R Util	EDLB-R Balancing
10	213.24	188.22	143.65	83.64	91.62
15	295.8	282.36	199.47	77.5	99.23
20	368.05	370.75	248.38	73.56	104.41
25	440.75	463.99	305.28	72.2	109.16
30	519.15	556.51	358.34	70.76	111.18
35	591.15	646.02	410.51	69.86	113.46
40	679.52	741.06	466.75	69.27	113.81
45	742.43	827.75	518.39	68.91	115.58
50	827.49	921.15	570.83	68.2	115.86
55	899.61	1008.96	623.51	68	116.62
60	966.86	1101.4	680.47	68.02	117.84
65	1048.63	1195.92	735.48	67.7	118.68
70	1111.46	1281.42	786.25	67.54	119.21
75	1211.47	1374.6	840.68	67.37	118.23
80	1269.73	1461.21	892.05	67.24	119.53
85	1333.61	1545.9	940.6	67.01	120.12
90	1406.1	1634.97	994.39	66.99	120.37
95	1482.35	1726.21	1046.25	66.79	120.51
100	1548.29	1815.77	1102.32	66.89	121.06

TABLE IV. Results obtained with 6 VMs.

Copyright © 2026 by Riset Publishing Services LLC

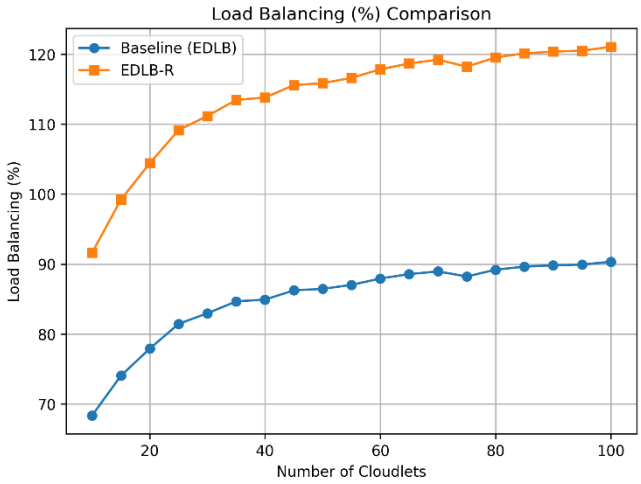

FIGURE 3. Balancing percentage comparison for 6 VMs.

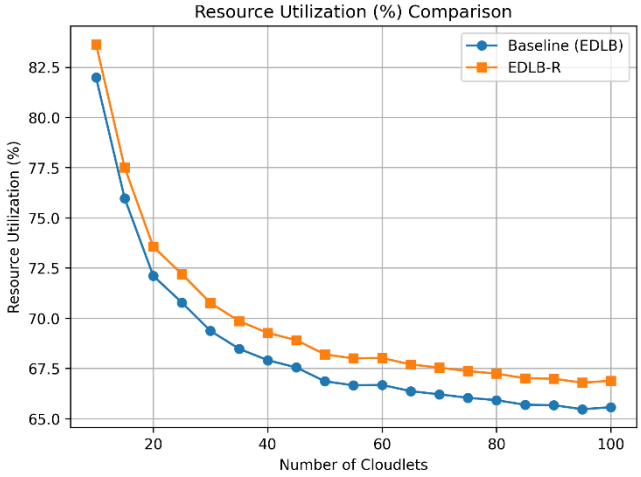

FIGURE 4. Resource utilisation comparison for 6 VMs.

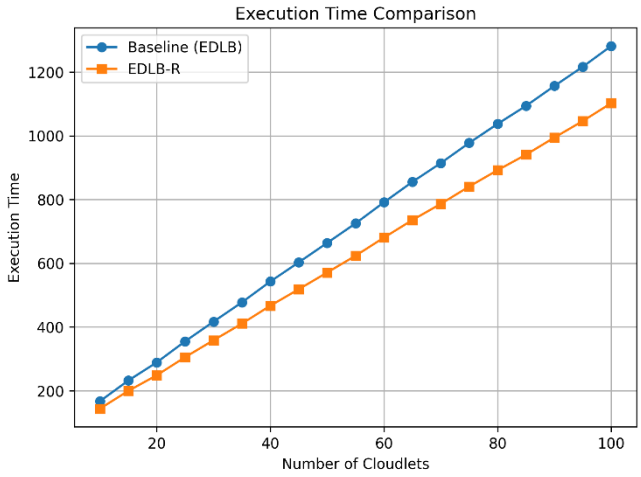

FIGURE 5. Execution time comparison for 6 VMs

Copyright © 2026 by Riset Publishing Services LLC

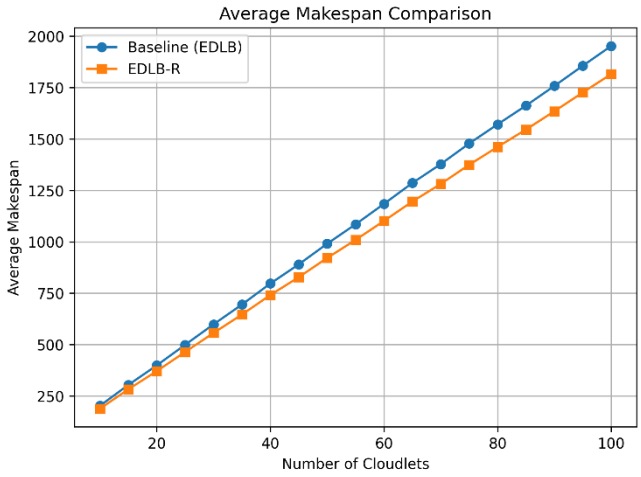

FIGURE 6. Average makespan comparison for 6 VMs

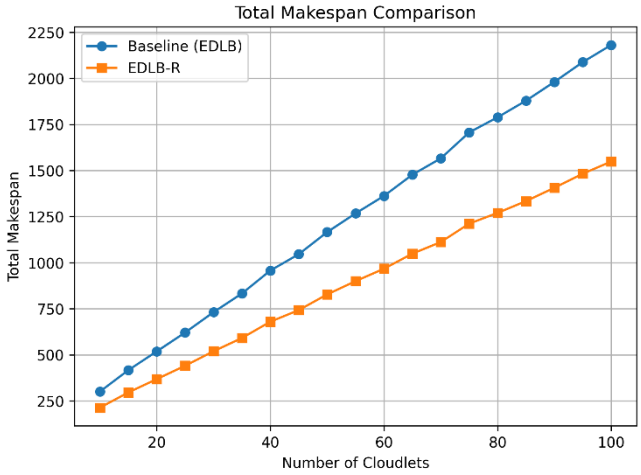

FIGURE 7. Total makespan comparison for 6 VMs.

Performance Metric	Baseline (EDLB)	Proposed (EDLB-R)	Percentage Change	Performance
Total Makespan	300.34–2180.69	213.24–1548.29	29.00%	Reduction ↓
Average Makespan	202.39–1952.44	188.22–1815.77	7.00%	Reduction ↓
Execution Time	167.04–1281.77	143.65–1102.32	14.00%	Reduction ↓
Resource Utilization	81.99–65.57	83.64–66.89	2.01%	Improvement ↑
Load Balancing	68.37–90.34	91.62–121.06	34.00%	Improvement ↑

TABLE V. Results

Table V summarizes the comparative performance of the baseline EDLB and the proposed EDLB-R algorithms. The experimental results demonstrate that EDLB-R consistently outperforms the baseline across all evaluated metrics. Specifically, the proposed algorithm achieves a 29% reduction in total makespan, a 7% reduction in average makespan, and a 14% reduction in execution time, indicating faster task completion and improved scheduling efficiency. Furthermore, EDLB-R improves resource utilization by 2.01%, enabling more effective use of available computing resources. The load balancing performance is also enhanced by 34%, resulting in a more balanced workload distribution across virtual machines and reduced resource contention. Overall, these improvements confirm that the proposed EDLB-R algorithm provides a more efficient, scalable, and reliable task scheduling strategy than the baseline EDLB algorithm in CloudSim-based cloud computing environments.

VII. CONCLUSION AND FUTURE WORK

This paper presented EDLB-R (Enhanced Dynamic Load Balancing with RAM Awareness), a hybrid task allocation algorithm designed to improve task scheduling and resource management in cloud computing environments. The proposed algorithm extends the conventional EDLB approach by incorporating RAM-awareness into the virtual machine (VM) selection process, enabling cloudlets to be allocated based on both computational capability and available memory resources. This strategy improves workload distribution, minimizes VM overloading, and enhances the overall efficiency of cloud resource utilization.

The proposed algorithm was implemented using the CloudSim simulation toolkit within the Eclipse IDE and

Copyright © 2026 by Riset Publishing Services LLC

evaluated under varying workloads ranging from 400 to 2000 cloudlets executed on six virtual machines. The performance of EDLB-R was compared with the baseline EDLB algorithm using five key performance metrics: total makespan, average makespan, execution time, resource utilization, and load balancing percentage.

The experimental results demonstrate that EDLB-R consistently outperforms the baseline algorithm across all workload scenarios. The proposed approach reduces total makespan, average makespan, and execution time by selecting more suitable virtual machines based on processing capability and memory availability. Furthermore, EDLB-R achieves improved load balancing by distributing cloudlets more evenly among virtual machines, thereby reducing resource contention and preventing performance bottlenecks. The resource utilization results also indicate more efficient management of available computing resources throughout task execution. As the workload increases, the advantages of the proposed algorithm become increasingly significant, demonstrating its scalability and effectiveness in dynamic cloud environments.

Overall, the proposed EDLB-R algorithm provides an efficient and practical solution for cloud task scheduling by integrating RAM-aware decision making into the load balancing process. The obtained results confirm that intelligent resource selection significantly enhances cloud performance while maintaining balanced utilization of virtual machine resources.

Although the proposed algorithm demonstrates considerable improvements, several opportunities remain for future enhancement. Future work will focus on extending the scheduling framework by incorporating additional resource parameters such as CPU utilization, network bandwidth, storage I/O, and energy consumption into the VM selection process. Integrating machine learning and artificial intelligence techniques for workload prediction can further improve scheduling decisions by anticipating future resource demands and dynamically adapting task allocation strategies. The implementation of dynamic VM migration, auto-scaling mechanisms, and SLA-aware scheduling can further enhance system adaptability and service reliability under highly dynamic cloud workloads.

Additionally, the proposed EDLB-R algorithm can be evaluated in heterogeneous, distributed, and multi-cloud environments to assess its effectiveness under diverse infrastructure configurations. Future research may also validate the proposed approach on real cloud platforms such as Microsoft Azure, Amazon Web Services (AWS), and Google Cloud Platform (GCP) to investigate its practical applicability beyond simulation. Finally, incorporating security-aware scheduling, fault tolerance, and QoS-based optimization into the framework can further improve its robustness and suitability for next-generation cloud computing systems. These enhancements will contribute toward developing a more intelligent, adaptive, and scalable load balancing framework capable of meeting the growing demands of modern cloud computing infrastructures.

.

REFERENCES

[1] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” Software: Practice and Experience, vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/spe.995.

[2] N. Devi, S. Dalal, K. Solanki, U. K. Lilhore, S. Simaiya, and N. Nuristani, “A systematic literature review for load balancing and task scheduling techniques in cloud computing,” Artificial Intelligence Review, vol. 57, 2024.

[3] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, “Load-balancing algorithms in cloud computing: A survey,” Journal of Network and Computer Applications, vol. 88, pp. 50–71, Jun. 2017, doi: 10.1016/j.jnca.2017.04.007.

[4] A. Thakur and M. S. Goraya, “A taxonomic survey on load balancing in cloud,” Journal of Network and Computer Applications, vol. 98, pp. 43–57, Nov. 2017, doi: 10.1016/j.jnca.2017.08.010.

[5] M. Adhikari, S. Nandy, and T. Amgoth, “Metaheuristic-based task deployment mechanism for load balancing in IaaS cloud,” Journal of Network and Computer Applications, vol. 128, pp. 64–77, Feb. 2019, doi: 10.1016/j.jnca.2018.12.003.

[6] S. Chhabra and A. K. Singh, “Dynamic resource allocation method for load balance scheduling over cloud data center networks,” 2022.

[7] R. Andreoli, J. Zhao, T. Cucinotta, and R. Buyya, “CloudSim 7G: An integrated toolkit for modeling and simulation of future generation cloud computing environments,” 2024.

[8] P. Singh, M. Dutta, and N. Aggarwal, “A review of task scheduling based on meta-heuristics approach in cloud computing,” Knowledge and Information Systems, vol. 52, no. 1, pp. 1–51, Jul. 2017, doi: 10.1007/s10115-017-1044-2.

[9] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing,” Egyptian Informatics Journal, vol. 16, no. 3, pp. 275–295, Nov. 2015, doi: 10.1016/j.eij.2015.07.001.

[10] M. Masdari, S. S. Nabavi, and V. Ahmadi, “A survey of PSO-based scheduling algorithms in cloud computing,” Journal of Network and Systems Management, vol. 25, no. 1, pp. 122–158, Jan. 2017, doi: 10.1007/s10922-016-9378-4.

[11] Z. Zhou et al., “Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing,” Journal of Cloud Computing, vol. 12, 2023.

[12] R. Jain and N. Sharma, “A quantum inspired hybrid SSA–GWO algorithm for SLA-based task scheduling,” Cluster Computing, 2022.

[13] K. Ramya and S. Ayothi, “Hybrid Dingo and Whale Optimization Algorithm-based optimal load balancing,” Transactions on Emerging Telecommunications Technologies, vol. 34, no. 5, 2023.

Copyright © 2026 by Riset Publishing Services LLC

[14] O. L. Abraham et al., "Task scheduling in cloud environment: Techniques, challenges and opportunities," in Proc. IEEE, 2024.
[15] Y. Sanjalawe et al., "Smart load balancing in cloud computing: Integrating feature selection and optimization," 2025.
[16] N. Rana et al., "A systematic literature review on contemporary and future virtual machine scheduling techniques," Frontiers in Computer Science, vol. 6, 2024.
[17] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for VM placement in cloud computing," 2016.
[18] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Z. Khan, A. Alahmadi, and A. H. Alahmadi, "Load Balancing in Cloud Environment: A State-of-the-Art Review," IEEE Access, vol. 11, pp. 134517–134530, 2023, doi: 10.1109/ACCESS.2023.3337146.
[19] P. V. Lahande, P. R. Kaveri, J. R. Saini, K. Kotecha, and S. Alfarhood, "Reinforcement Learning Approach for Optimizing Cloud Resource Utilization With Load Balancing," IEEE Access, vol. 11, pp. 127567–127577, 2023, doi: 10.1109/ACCESS.2023.3329557.
[20] M. S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, M. A. Elmagzoub, G. Muhammad, and K. H. Talpur, "A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing," IEEE Access, vol. 11, pp. 11390–11404, 2023, doi: 10.1109/ACCESS.2023.3241279.
[21] Z. A. Khan, I. A. Aziz, N. A. Osman, and I. Ullah, "A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions," IEEE Access, vol. 11, 2023, doi: 10.1109/ACCESS.2023.3343877.
[22] O. L. Abraham, M. A. B. Ngadi, J. M. Sharif, and M. K. M. Sidik, "Task Scheduling in Cloud Environment—Techniques, Applications, and Tools: A Systematic Literature Review," IEEE Access, vol. 12, pp. 138252–138279, 2024, doi: 10.1109/ACCESS.2024.3466529.
[23] R. Zhanuzak, M. A. Ala'anzy, M. Othman, and A. Algarni, "Optimizing Cloud Computing Performance With an Enhanced Dynamic Load Balancing Algorithm for Superior Task Allocation," IEEE Access, vol. 12, pp. 172731–172751, 2024, doi: 10.1109/ACCESS.2024.3508793.
[24] P. Choppara and S. S. Mangalampalli, "Reliability and Trust Aware Task Scheduler for Cloud-Fog Computing Using Advantage Actor Critic (A2C) Algorithm," IEEE Access, vol. 12, 2024, doi: 10.1109/ACCESS.2024.3432642.
[25] S. Singhal et al., "Energy Efficient Load Balancing Algorithm for Cloud Computing Using Rock Hyrax Optimization," IEEE Access, vol. 12, pp. 48737–48749, 2024, doi: 10.1109/ACCESS.2024.3380159.
[26] S. Tiwari and B. B. M., "A Neighborhood Inspired Multiverse Scheduler for Energy and Makespan Optimized Task Scheduling for Green Cloud Computing Systems," IEEE Access, vol. 12, pp. 157272–157298, 2024, doi: 10.1109/ACCESS.2024.3484388.
[27] D. Alsadie, "A Comprehensive Review of AI Techniques for Resource Management in Fog Computing: Trends, Challenges, and Future Directions," IEEE Access, vol. 12, pp. 118007–118059, 2024, doi: 10.1109/ACCESS.2024.3447097.
[28] M. S. R. Krishna and D. K. Vali, "Meta-RHDC: Meta Reinforcement Learning Driven Hybrid Lyrebird Falcon Optimization for Dynamic Load Balancing in Cloud Computing," IEEE Access, vol. 13, pp. 36550–36574, 2025, doi: 10.1109/ACCESS.2025.3544775.
[29] N. Francis and N. V. Balaji, "Effective Load Balancing With Adaptive Task Scheduling Using the Weighted-Priority Approach," in Proc. IEEE Int. Conf. Computer Communication and Management Computing (ICCMC), 2025, pp. 829–836, doi: 10.1109/ICCMC65190.2025.11139900.
[30] P. Choppara and S. S. Mangalampalli, "Adaptive Task Scheduling in Fog Computing Using Federated DQN and K-Means Clustering," IEEE Access, vol. 13, pp. 75466–75492, 2025, doi: 10.1109/ACCESS.2025.3563487.
[31] O. L. Abraham, M. A. B. Ngadi, J. M. Sharif, and M. K. M. Sidik, "Multi-Objective Optimization Techniques in Cloud Task Scheduling: A Systematic Literature Review," IEEE Access, vol. 13, pp. 12255–12291, 2025, doi: 10.1109/ACCESS.2025.3529839.
[32] P. Choppara and B. Lokesh, "Efficient Task Scheduling and Load Balancing in Fog Computing for Crucial Healthcare Through Deep Reinforcement Learning," IEEE Access, vol. 13, pp. 26542–26563, 2025, doi: 10.1109/ACCESS.2025.3539336.
[33] M. Saini and P. Agarwal, "A DevOps-Based Framework for Automating Real-Time Cloud Data Collection Workflows," in 2025 International Conference on Emerging Technologies and Innovation for Sustainability (EmergIN). IEEE, Nov. 2025, pp. 144–149, doi: 10.1109/EmergIN67762.2025.11450898.

Copyright © 2026 by Riset Publishing Services LLC