 ISSN:2302-8149
e-ISSN:2540-9719

Improving Cloud Task Scheduling in Cloud Sim Plus using Demand-Aware VM Selection

[1]Nagham Sultan*
[1]Department of Computer Science, University of Mosul, Mosul, 41002, Iraq
*e-mail: naghamajeel@uomosul.edu.iq

(***received*:** 15 March 2026, ***revised*:** 10 April 2026, ***accepted*:** 12 April 2026)

Abstract

Deep reinforcement learning and metaheuristics are becoming popular ways to study cloud task scheduling to make things like makespan and completion time better; however, these methods often require a lot of computing power and can make it harder to understand and repeat the results. This paper suggests a simple Demand-Aware VM Selection policy for CloudSim Plus (simulated tasks) that improves the basic scheduling process by better matching tasks with virtual machines based on task needs, while leaving the rest of the simulation the same. The method was tested against the default time-shared baseline with three workload sizes (200, 500, and 1000 cloudlets) and six random seeds (42-47). The experimental results demonstrate that the proposed method drastically shortened the overall time of task completion: it raised the average time by 35.65% for 200 cloudlets (A computational task/work unit submitted for execution is CloudSim abstraction) and 28.86% for 500 cloudlets. In terms of average completion time, the newly planned lowered the average by 26.68% at 200 cloudlets and 28.53% at 1000 cloudlets; on the other hand, the 500-cloudlet scenario showed very high variability and even a slightly negative average improvement (-1.42%), indicating that completion-time averaging was sensitive under that workload regime even though there was a strong makespan gain. The results show that demand-aware binding is a clear, repeatable, and easy-to-add improvement for scheduling in CloudSim Plus, which can serve as a better starting point or as part of more advanced optimization and learning systems.

Keywords: completion time; cloudsim plus; makespan; multi-datacenter; simulation; task scheduling; VM allocation; workload-aware scheduling

1 Introduction

Cloud task scheduling marks the point where a "clean" resource abstraction gradually transforms into a sophisticated optimization problem: various tasks, changing over time, must be efficiently assigned to virtual machines (VMs) under QoS constraints, and ineffective assignments typically show up as the long-tail completion times and continuous SLA pressure [1]. Therefore, the literature habitually describes scheduling as infeasible computationally under realistic conditions (NP-hard/NP-complete in practice) and uses makespan (Maximum completion time of all cloudlets) as a prime measure of the overall throughput as well as the user-perceived delay [2]. Nonetheless, scheduling has evolved way past the single-metric optimization stage a long time ago. Typically, makespan is not considered without linking it to the execution cost, resource utilization, migration overhead, reliability, and energy consumption; hence, multi-objective formulations have nowadays become standard practice rather than rare occasions [3]. This broader range of the objective space has spurred various solution families. Metaheuristics try to manage the complicated search spaces effectively, mostly through hybridization strategies that balance exploration and exploitation, while learning-based schedulers strive to facilitate decision-making in the face of changing workloads and conditions [4].

In the case of workflow-oriented applications, the problem is often redefined as task-resource matching because even if the CPU is a dominant bottleneck for a single task, other factors like memory, storage, and communication can be limiting, and giving a task the "wrong kind of capacity" is equivalent to wasting time and lowering the performance [5]. A key reason for constantly improving schedulers is that simulation environments can now provide quick, controlled, and comparable tests, allowing us to avoid large and costly real-world deployments. Many research papers

 ISSN:2302-8149
e-ISSN:2540-9719

assess scheduling policies using the CloudSim environment, making simulation data a common and easy-to-review standard for improving mappings [6], [7]. In such a milieu, CloudSim Plus emerges as a modern, actively maintained, and more elegant evolution of the CloudSim family. It puts more emphasis on clean software design and experiment reproducibility, but at the same time, it is made sufficiently friendly to established CloudSim conventions to allow comparisons with the previous results [8].

Firstly, recent studies show a clear trend: many smart schedulers clearly identify what tasks need and what VMs can actually provide, using methods like sensitivity-aware pairing, trust/QoS indicators, or multi-resource objective functions. However, typical simulation baselines consistently depend on the default time-shared model that is mainly concerned with an equal sharing after the tasks have already been allocated rather than the initial task-VM assignment being explicitly demand-aware [5]. Thus, a methodological gap remains; considerable improvements can be achieved by refining the selection of VM during the binding decision, specifically by first grounding this decision in task demand signals and then introducing heavier learning loops or more complex metaheuristic searches. This gap is, in a way, related to the notion of "optimal pairing" in the workflow scheduling research, which basically deals with how closely task requirements correspond to VM configurations as the main performance factor.

In this context, the study suggests a simple, rule-based demand-aware VM selection policy for CloudSim Plus that takes into account CPU, memory, and bandwidth needs when deciding which VM to bind a cloudlet to. The technique improves pre-submission binding choices, thereby boosting the default time-shared workflow. It does so without changing the execution semantics or tampering with the simulator's internal workings. We implement a well-defined evaluation framework for the purpose of a fair and repeatable test that presents makespan and completion-related performance metrics effectively after fixed infrastructure and paired workloads seed. Making the binding decision demand aware results in making the schedule shorter and reducing the average completion time for the entire workload, while the computational overhead remains O (|C| × |V|) and it is fully compatible with the standard CloudSim Plus workflow, as shown in the experimental analysis.

The rest of the paper is organized as follows: In Section 2 the authors briefly review a dozen different scheduling paradigms, such as trust-aware and metaheuristic scheduling, DRL-driven dual scheduling, multi-objective hybrids, workflow pairing, VM consolidation, and VM placement/migration. Section 3 presents the system model and formulates the problem of demand-aware binding in CloudSim Plus. In Section 4 the authors discuss the demand-aware VM selection policy. Section 5 describes the CloudSim Plus experimental environment and the evaluation criteria (makespan and completion indicators). Section 6 presents the experimental data and its interpretation. Section 7 concludes the paper and outlines the plans for further research.

2 Literature Review

Here are short summaries of important studies on cloud task scheduling and cloudlet-to-VM binding that have used approaches focused on quality of service, trust, and learning/optimization in their algorithms. We show the main approaches and respective limitations of these works before presenting our demand-aware selection, which can be considered a classy and straightforward substitution for very standard baselines.

With a view to making multi-cloud environments trustworthy, Mangalamapalli et al. (2023) developed a trust-oriented, fault-tolerant task scheduling method by integrating Harris Hawks Optimization and deep reinforcement learning; the method was reported to have significantly decreased the task failure rate and enhanced trust management; however, a rather high computational overhead was still associated with the method, and the complex deep learning models required by it made scaling large simulations impractical [9]. Pan et al. in 2025 set deep reinforcement learning as the optimizer for task scheduling and resource allocation in their two-level scheduling scheme; besides finding that makespan and resource utilization were increased, the framework tightly coupled learning agents with the system state, thus making it difficult to fairly compare the baseline methods [10].

In a paper examining the firefly optimization, Mangalamapalli et al. showed that trust-aware task scheduling can yield improvements in execution time and trust metrics; overemphasizing trust

 ISSN:2302-8149
e-ISSN:2540-9719

modeling, the algorithm failed to take into account VM-level demand characteristics granularity during task binding [6]. Pradhan and Bisoy (2022) presented a PSO-based load balancing algorithm for the cloud platform with the help of which they were able to lower the response time and improve load distribution significantly; nevertheless, their model was at the load-balancer level, and therefore cloudlet-to-VM binding methods were not extensively considered [11]. A multi-objective trust-aware scheduling approach based on the use of whale optimization was developed in 2023, which aims at makespan reduction as well as trust value increment; however, the method is still burdened with several objective tuning parameters, which, among other things, result in it being highly sensitive to workload configuration [12].

In 2023, Sudheer Mangalamapalli and his team suggested a new task scheduling method using Cat Swarm Optimization that reduced execution delays and set priorities but couldn't adjust to changes in runtime demand [13]. Meanwhile, Lilhore et al. introduced a quantum-inspired hybrid multi-objective framework (QHRMOF) in 2025, optimized for energy efficiency and load balancing. The experiments showed that energy consumption was significantly reduced; however, the framework primarily focused on infrastructure metrics rather than per-cloudlet demand awareness [14]. In 2025, Yadav and Sharma worked on a method for task allocation that considered quality of service and energy efficiency through hybrid scheduling, which improved throughput and compliance with service level agreements; however, their approach relied on hybrid heuristics and did not separately show the impact of virtual machine selection policies [15]. In 2024, Mangalamapalli et al. came up with a deep reinforcement learning-based scheduler for multi-cloud environments and emphasized the adaptability of the system under the changes of the workload; though, there was a trade-off between effective results and the lack of transparency caused by the learning-centric design, which in turn limited the direct comparison with the classical schedulers [16]. Moreover, Hua and Zheng in 2025 investigated the scheduling of workflow in IaaS clouds by means of optimal task-VM pairing and reported significantly lowered execution cost and better performance; however, the solution was based on the assumption that workflow features are constant, and hence, it did not generalize well to heterogeneous, mixed workloads [5].

Further to the above, the examined publications have contributed to the development of trust-aware, DRL-based, and metaheuristic cloud scheduling solutions, yet the main constraint with which they have still been struggling is understandable. Many methods either depend on complex learning algorithms, which are advanced computational techniques that can be difficult to understand and replicate, or they use traditional schedulers that overlook the specific resource needs of tasks when connecting cloudlets and virtual machines (VMs). Our study fills in a gap that is a lack of a nimble, demand-aware binding phase. We enhance conventional scheduling, making it more efficient by emphasizing CPU, memory, and bandwidth needs when selecting VMs in the binding stage, while ensuring it remains comprehensible and reproducible in CloudSim Plus. Table 1 summarizes previous works and identifies the gap this study aims to address.

Table 1 Related work summary table

Authors [Year]	Study focus	technique	Main metrics	Main results / improvement	Research Gaps
[9] Mangalam apalli et al., 2023	Fault tolerance, reliability.	Hybrid Harris Hawks Optimizati on + DRL	Task failure, trust management	Reduced task failure, improved trust management	High computational overhead and complex learning reduce transparency and reproducibility
[10] Pan et al., 2025	Joint task scheduling and resource allocation	Deep Reinforce ment Learning-based dual scheduling	Makespan, resource utilization	Improved makespan and resource utilization	binding-time demand awareness is not isolated or explicitly analyzed

 ISSN:2302-8149
e-ISSN:2540-9719

		framework			
[6] Mangalam apalli et al., 2023	Trust-aware task scheduling	Firefly Optimizati-on with trust modeling.	Execution time; trust metrics.	Improved execution time and trust metrics.	does not explicitly model CPU/RAM/BW demands during cloudlet-to-VM binding.
[11] Pradhan and Bisoy, 2022	Load balancing for cloud platforms	PSO	Response time, load distribution	Reduced response time, improved load distribution	Operates at load-balancer level, does not address cloudlet-to-VM binding or per-task demand-aware VM selection
[12] Mangalam apalli et al., 2023	Multi-objective trust-aware scheduling	Whale Optimizati-on	Makespan, trust value	gains in makespan and trust objectives	Sensitive to objective-weight tuning
[13] Mangalam apalli et al., 2023	Priority-based task scheduling	Cat Swarm Optimizati-on	Execution	Reduced execution delay	Static prioritization, does not adapt to dynamic demand fluctuations
[14] Lilhore et al., 2025	Energy-efficient and load-balancing scheduling	QHRMOF.	Energy consumption, load balancing indicators	reductions in energy consumption	Focuses on infrastructure-level metrics only
[15] Yadav and Sharma, 2025	QoS-aware, energy-efficient task allocation, throughput	Hybrid scheduling heuristics	Throughput, SLA compliance	Improved throughput and SLA compliance	Does not isolate the impact of VM selection
[16] Mangalam apalli et al., 2024	Adaptive scheduling under dynamic multi-cloud workloads	Deep Reinforce ment Learning-based scheduler	Adaptive behavior under dynamic workloads	adaptive behavior and improved outcomes under dynamic conditions	Learning-centric and less transparent, it does not provide a simple demand-aware binding baseline
[5] Hua and Zheng, 2025	Workflow scheduling in IaaS clouds	Optimal pairing between workflow tasks and VMs	Execution cost	Lower execution cost, improved performance	limited generality for heterogeneous mixed workloads

3 Problem Definition and System Model

This section provides a formal specification of the cloud environment and the task abstraction adopted in this work. It also clarifies the scheduling context within which demand-aware VM selection is analyzed.

 ISSN:2302-8149
e-ISSN:2540-9719

3.1 System and Task Model

We model the cloud environment using CloudSim Plus as a multi-datacenter infrastructure comprising datacenters, physical hosts, virtual machines (VMs), and computational tasks (cloudlets). The infrastructure is defined as:

$$D = \{DC_1, DC_2 \dots \dots, DC_K\}$$

where each datacenter DC_k contains a set of physical hosts:

$$DC_k = \{H_{k,1}, H_{k,2}, \dots, H_{k,M}\}$$

Each host *H* provides processing capacity, memory, bandwidth, and storage; hosts allocate resources to VMs using a time-shared VM scheduling policy. A set of VMs is deployed across the datacenters:

$$V = \{VM_l, VM_2 \dots VM_N\}$$

where each VM_j is configured by its number of processing elements and resource capacities (CPU/MIPS (Million Instructions Per Second), RAM, bandwidth, and storage). To isolate the effect of binding decisions, VMs are configured homogeneously across experiments, and cloudlets inside each VM are executed under a time-shared cloudlet scheduling policy. The workload is represented by a finite set of independent cloudlets:

$$C = \{C_1, C_2 \dots \dots . C_L\}$$

Each cloudlet C_i is characterized by its computational length and resource demand vector:

$$C_i = (Len_i, RAM_i, BW_i)$$

where Len_i is the cloudlet length in million instructions, and RAM_i, BW_i denote memory and bandwidth requirements, respectively. Cloudlets are independent, non-preemptive, and without precedence constraints. Cloudlets are created by a deterministic workload generator that is parameterized by a random seed; hence, for any specific seed, the very same workload instance is utilized across the strategies so that the differences noticed can be attributed to the scheduling/binding decisions rather than the variation of the workload.

3.2 Binding/Scheduling Model and Objectives

The execution process is decomposed into two phases: (i) a binding phase, where each cloudlet is associated with a target VM; and (ii) an execution phase, where the cloudlets run on their assigned VMs under fixed VM and cloudlet scheduling policies. In the baseline configuration, cloudlets are submitted to the broker without explicit VM binding; VM assignment is handled implicitly by the default broker behavior, which does not explicitly account for cloudlet-specific resource demands during binding. In contrast, the proposed approach introduces an explicit binding function:

$$f: C \rightarrow V$$

That assigns each cloudlet to exactly one VM prior to submission. The decision on which cloudlet goes with which VM depends on how well each cloudlet's needs (Len_i, RAM_i, BW_i) match the VM's available resources, and the rules for running them stay the No VM migration, replication, or dynamic reconfiguration is performed after the binding step. Given a fixed infrastructure *D*, a set of homogeneous VMs *V*, and a heterogeneous workload *C*, the problem addressed in this work is to determine a cloudlet-to-VM binding function f that improves system performance under shared-resource constraints. The primary objective is to minimize the makespan, as shown in equation 1.

 ISSN:2302-8149
e-ISSN:2540-9719

$$Makespan = \max_{C_i \in \mathcal{C}} \{FT(C_i)\} \quad (1)$$

where $FT(C_i)$ is the finishing time of cloudlet C_i. A secondary objective is to reduce the average completion time as shown in equation 2.

$$AvgCompletion = \frac{1}{|\mathcal{C}|} \sum_{i=1}^{|\mathcal{C}|} FT(C_i) \quad (2)$$

These objectives are optimized subject to the constraints that: (a) each cloudlet is assigned to exactly one VM; (b) the infrastructure configuration remains fixed; and (c) VM and cloudlet scheduling policies are identical across strategies, ensuring that any measured improvement is attributable to the binding strategy itself.

4 Proposed Method: Demand-Aware VM Selection

This section presents a proposed demand-aware virtual machine selection mechanism for cloud computing virtualization. This approach optimizes the connection phase by incorporating explicit resource demand awareness while preserving CloudSim Plus's execution semantics and scheduling policies. Intervention is intentionally minimized: the connection step is modified only before deployment.

4.1 Demand and Capacity Modeling

In a baseline CloudSim Plus workflow, task execution policies are separated from cloudlet-to-VM binding. While execution fairness is maintained after assignment, the binding phase itself is demand-agnostic. Consequently, heterogeneous cloudlets may be arbitrarily mapped to VMs, increasing contention and degrading completion time. To address this limitation, the proposed method introduces explicit demand–capacity modeling at binding time. Let the workload be:

$$C = \{C_1, C_2, \ldots, C_{|C|}\}$$

Each cloudlet (Ci) is represented by a demand vector:

$$D_{Ci} = \left(d_i^{cpu}, d_i^{ram}, d_i^{bw}\right)$$

where:

- d_i^{cpu} denotes the computational demand expressed in million instructions
- d_i^{ram} denotes the required memory
- d_i^{bw}denotes the required bandwidth

Each virtual machine ($VM_j \in V$) is represented by a capacity vector:

$$C_{VMj} = \left(c_j{}^{cpu}, c_j{}^{ram}, c_j{}^{bw}\right)$$

where:

- c_j^{cpu} represents the VM's processing capability
- c_j^{ram} represents the available memory
- c_j^{bw} represents the available bandwidth.

Although VMs are configured homogeneously in the experimental setup, their runtime availability may differ due to previously assigned cloudlets.

 ISSN:2302-8149
e-ISSN:2540-9719

4.2 Demand-Aware Binding Rule and Algorithm

For each cloudlet (Ci), the proposed method evaluates all candidate virtual machines based on their compatibility with the cloudlet demand profile. The compatibility score is calculated using standard demand-to-capacity ratios as shown in equation 3.

$$Score(C_i, VM_j) = w_{cpu} \cdot \frac{d_i^{cpu}}{c_j^{cpu}} + w_{ram} \cdot \frac{d_i^{ram}}{c_j^{ram}} + w_{bw} \cdot \frac{d_i^{bw}}{c_j^{bw}} \quad (3)$$

where w_{cpu}, w_{ram} *and* w_{bw}: Fixed weights reflect the relative importance of each dimension of resources.

A low score indicates a better match between the demand for small cloud computing cloudlets and the capacity of virtual machines. A small cloudlet is associated with a virtual machine that lowers this score while meeting basic feasibility constraints. For the binding algorithm, the binding procedure is executed prior to cloudlet submission and follows a deterministic process. Below, Algorithm 1 shows Demand-Aware Cloudlet-to-VM Binding:

1. For each cloudlet Ci ∈ C:

   -The DCi demand vector is extracted.

   - For each virtual machine: VMj∈V:
     - Calculate the score (Ci, VMj)
     - Select the virtual machine with the lowest score;
     - Bind Ci to the selected virtual machine

2. Submit all bound cloudlets to the broker. This algorithm works deterministically, and it does not introduce any stochastic or adaptive behavior.

4.3 Computational Properties and Integration

Let |C| be the number of cloudlets and |V| the number of virtual machines. The binding algorithm essentially compares each cloudlet with all the VMs, which leads to a time complexity of:

$$O(|C| \times |V|)$$

Since VM numbers in normal CloudSim Plus experiments are very low, this overhead is almost negligible compared to the total simulation runtime. The binding is done immediately before the cloudlets are submitted. The binding function picks the VM for each cloudlet even before the broker's submit function gets called. No changes are made to:

- VM scheduling policies,
- cloudlet execution policies,
- broker behavior after submission,
- event handling or simulation timing.

The technique in question thus functions like a thin pre-processing layer in the normal CloudSim Plus workflow. By merely refining the binding phase through demand, capacity matching, a Demand-Aware VM Selection method is capable of improving cloud scheduling performance. The method without altering the execution semantics and even without the application of learning-based or metaheuristic mechanisms stays opaque, reproducible, and very efficient computationally. Furthermore, it offers measurable improvements in makespan and average completion time.

 ISSN:2302-8149
e-ISSN:2540-9719

5 Experimental Setup

From the experiment methodology section, we demonstrate the assessment of the relationship between the cloudlet-to-VM binding and demand awareness for setting up a controlled, repeatable, and fair experiment, which can be relied on to compare the performance of the baseline model and the new model accurately. Correlation was the only variable factor which experiment took into account.

5.1 Simulation Environment and System Configuration

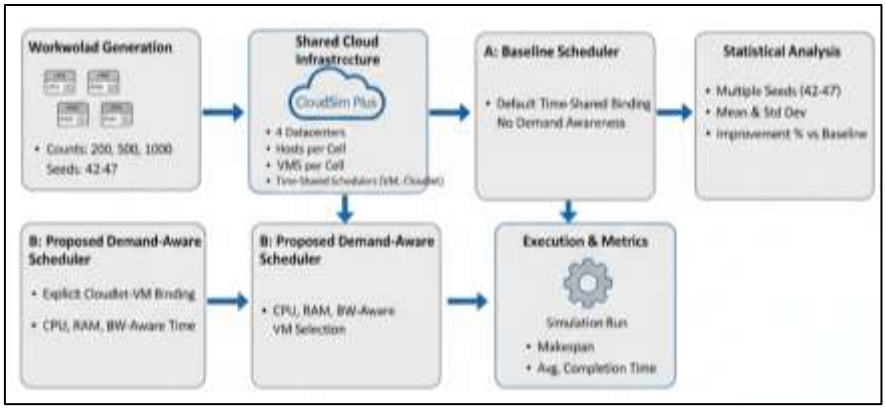
We conducted all the tests in the CloudSim Plus program, a discrete-event simulation (simulation based on successive, instantaneous changes in the system) software framework that is the most popular one among researchers in cloud scheduling. Moreover, the CloudSim Plus core has not been modified at all; the logic of scheduling was only at the application level. In addition, the baseline and the new proposals were executed under the same simulator configurations.

According to the simulation, there are four data centers (cells), each of which has a limited number of homogeneous physical hosts. A host is a physical machine that consists of processing elements (PEs), memory, bandwidth, and storage. In addition, it is mentioned that the host applies a time-shared VM scheduling policy whereby the resources are delivered to the virtual machines. A fixed number of virtual machines are scattered across the datacenters; all VMs are identical in terms of the number of processing elements, MIPS capacity, memory, bandwidth, and storage. Furthermore, the cloudlet execution in each VM is controlled by the time-shared cloudlet scheduling policy. Additionally, the setup of the infrastructure remains the same in all tests and for all scheduling methods; therefore, any differences in performance are solely due to how tasks are assigned to VMs.

We employ CloudSim Plus in creating a computational model of a multi-datacenter IaaS system with CELLS = 4 datacenters, each of which consists of HOSTS_PER_CELL = 4 identical hosts. The specification of each host includes HOST_PES = 8 processing elements, HOST_MIPS_PER_PE = 2000, HOST_RAM_MB = 32768, HOST_BW_Mbps = 20000, and HOST_STORAGE_MB = 1, 000, 000; and for VMs it has a time-shared scheduler for VM. We create the VMS_PER_CELL = 4 identical VMs per datacenter (VM_PES = 2, VM_MIPS_PER_PE = 2000, VM_RAM_MB = 4096, VM_BW_Mbps = 2000, VM_SIZE_MB = 10, 000) with time-shared cloudlet scheduling. Figure 1 displays a very simple version of the experimental procedure aimed at comparative evaluation. The scheduling strategies to be compared are:

- **The baseline strategy**
  Without any special VM binding, cloudlets are submitted; the default CloudSim Plus broker behavior handles the case, which means that cloudlet-specific demand features are not taken into consideration during binding.

- **The proposed strategy**
  Each cloudlet is explicitly linked to a VM through the demand-aware VM selection function before it is submitted. The execution policies remain the same as the baseline. No learning-based, metaheuristic, adaptive, or predictive mechanisms are introduced.



 ISSN:2302-8149
e-ISSN:2540-9719

Figure 1 Experimental workflow for comparative evaluation

Figure 2 illustrates the centralized configuration module through which parameters for datacenter, VM, and workload can be set for the entire experimental run.

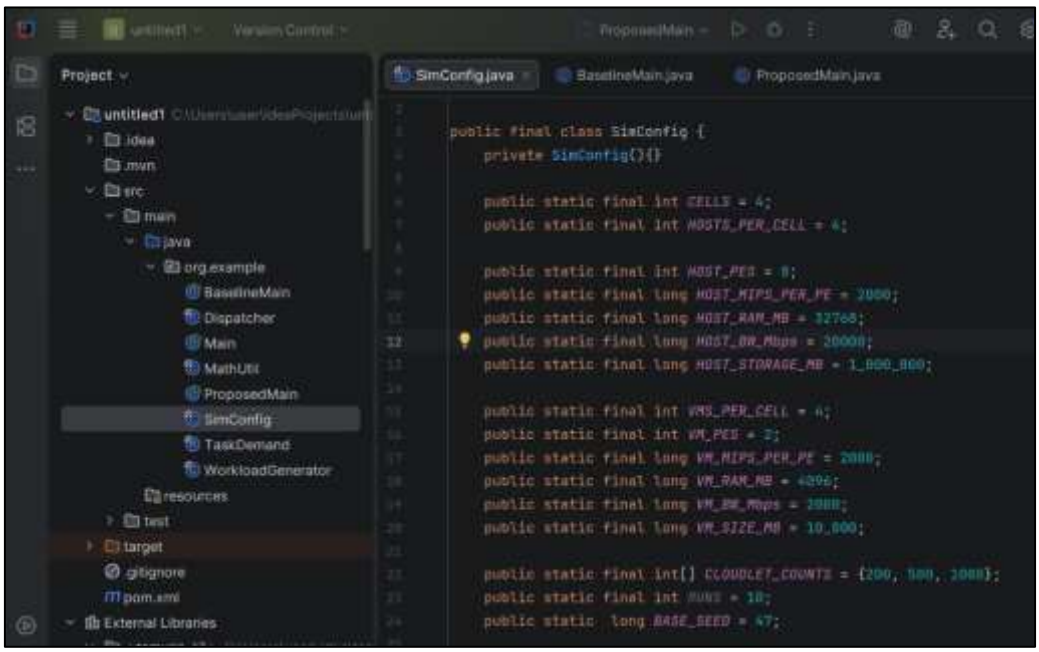


Figure 2 Simulation configuration parameters of the proposed cloudsim

5.2 Workload Configuration and Execution Protocol

The workload contains independent cloudlets generated by a deterministic workload generator. Computational length, memory, and bandwidth requirements are the three characteristics of a cloudlet. For scalability evaluation, three workload sizes of 200, 500, and 1000 cloudlets are considered. To eliminate the stochastic variability, each workload size is run with six different random seeds (42-47), thus resulting in (N=6) runs for each scenario. Steps are repeated for each workload size and random seed combination:

- The CloudSim Plus environment is set up.
- Using the pre-defined parameters, the datacenters, hosts, and VMs are created.
- The workload is obtained from the selected seed.
- The baseline strategy is executed, and the performance parameters are noted.
- The proposed strategy is executed in the same environment.
- The outcomes are stored for further analysis.

There are no changes of parameters between the different runs, and even during the run, no VM migration, elasticity, or dynamic reconfiguration is allowed. In Figure 3 it is clear that the experiments are conducted according to a methodical pipeline from the very first step to the last one, from definition of the workload to doing statistical analysis.

 ISSN:2302-8149
e-ISSN:2540-9719

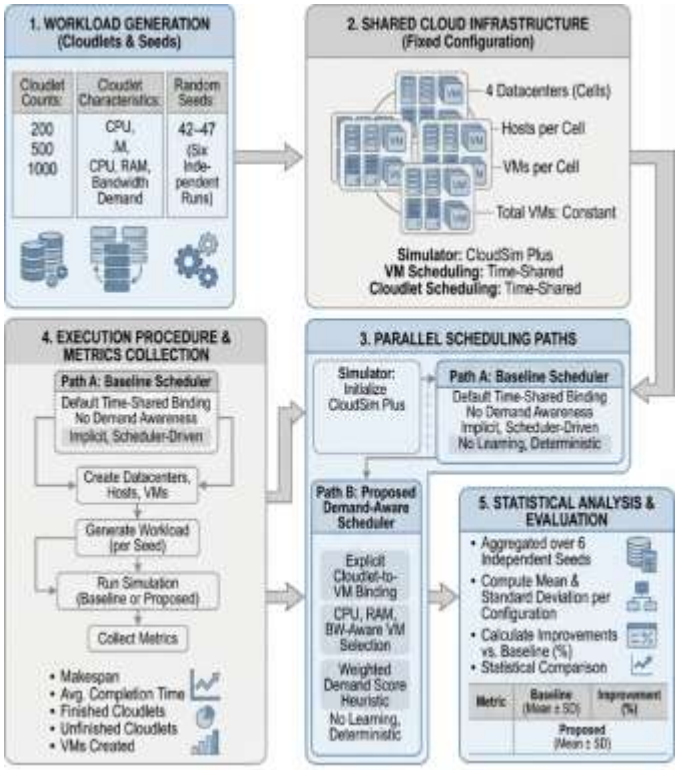


Figure 3 End-to-end experimental workflow in cloudsim plus

5.3 Performance Metrics and Statistical Analysis

The assessment focuses on standard cloud computing scheduling metrics; these metrics include: Makespan, defined as the finishing time of the last completed cloudlet, and Average Completion Time (AvgCompletion), defined as the mean finishing time of all cloudlets, and the Number of finished cloudlets, Number of unfinished cloudlets, and Number of created virtual machines. Relative performance is quantified through per-seed improvement ratios computed as shown in equation 4.

$$Improvement\% = \frac{Baseline - Proposed}{Baseline} * 100 \qquad (4)$$

The time duration (makespan) and the average completion time are aggregated across seeds for each workload size using mean and standard deviation. The percentage improvements are calculated per seed, and then the statistical summaries are obtained, rather than just being derived from aggregated means. This methodology is less vulnerable to stochastic fluctuations and also avoids misleading interpretations due to outlier runs.

5.4 Experimental Validity and Reproducibility

Internal validity is ensured by maintaining identity by setting up the same infrastructure configurations, generating the workload deterministically, using the same scheduling policies, and having paired seed execution on different strategies. The only thing that differentiates the experiments is the cloudlet-to-VM binding decision. This kind of design allows for a very simple performance analysis of demand-aware binding and, moreover, can be easily reproduced in a standard CloudSim Plus environment.

6 Results and Discussion

This section of the paper presents the experiments conducted to show the advantages of the proposed demand-aware binding method. Firstly, we cover the major patterns that emerged in all the scenarios and then focus on makespan and average completion time analysis. The performance

 ISSN:2302-8149
e-ISSN:2540-9719

variations discussion points to the origin of the fluctuations in the demand and capacity-matching behavior at cloudlet-to-VM binding.

6.1 Overview of Experimental Outcomes

Table 2 presents the total results for six different simulation runs with random seeds 42-47 for three different workload sizes (200, 500, and 1000 cloudlets). Both scheduling strategies were able to process the entire workload during all the tests (Finished Cloudlets = Cloudlets; Not Finished = 0), and the number of virtual machines instantiated stayed the same (VMsCreated = 16). Therefore, any performance change must come from the task, VM binding mechanism. The default time-shared binding policy is the base setup, and the proposed method is a demand-aware dispatcher.

Table 2 CloudSim plus results: seeds from 42 to 47, statistics (n = 6 per senario)

	Algorithm	Cloudlets	Seed	VM Total	Makespan	Avg Completion	VMs Created	Makespan Improvement %	AvCompletion Improvement %
1	Baseline	200	42	16	123.97	64.10	16		
2	Proposed	200	42	16	86.93	52.88	16	29.88	17.50
3	Baseline	500	42	16	192.08	124.64	16		
4	Proposed	500	42	16	191.97	124.64	16	0.06	0.00
5	Baseline	1000	42	16	568.17	344.29	16		
6	Proposed	1000	42	16	389.40	252.55	16	31.46	26.65
7	Baseline	200	43	16	150.7	75.85	16		
8	Proposed	200	43	16	82.01	44.67	16	45.58	41.11
9	Baseline	500	43	16	303.57	173.63	16		
10	Proposed	500	43	16	191.32	115.96	16	36.98	33.21
11	Baseline	1000	43	16	570.55	348.54	16		
12	Proposed	1000	43	16	396.23	248.74	16	30.55	28.63
13	Baseline	200	44	16	135.75	65.63	16		
14	Proposed	200	44	16	78.55	48.34	16	42.14	26.34
15	Baseline	500	44	16	304.38	167.37	16		
16	Proposed	500	44	16	195.81	380.35	16	35.67	-127.25
17	Baseline	1000	44	16	579.15	337.9	16		
18	Proposed	1000	44	16	116.38	240.21	16	79.91	28.91
19	Baseline	200	45	16	123.19	63.34	16		
20	Proposed	200	45	16	82.74	49.39	16	32.84	22.02
21	Baseline	500	45	16	321.87	169.31	16		
22	Proposed	500	45	16	200.27	117.02	16	37.78	30.88
23	Baseline	1000	45	16	597.59	356.15	16		
24	Proposed	1000	45	16	387.98	242.95	16	35.08	31.78
25	Baseline	200	46	16	124.05	66.88	16		
26	Proposed	200	46	16	83.65	50.05	16	32.57	25.16

 ISSN:2302-8149
e-ISSN:2540-9719

27	Baseline	500	46	16	314.85	168.69	16		
28	Proposed	500	46	16	210.64	124.14	16	33.10	26.41
29	Baseline	1000	46	16	629.28	339.36	16		
30	Proposed	1000	46	16	376.25	245.27	16	40.21	27.73
31	Baseline	200	47	16	121.43	66.42	16		
32	Proposed	200	47	16	83.91	47.87	16	30.90	27.93
33	Baseline	500	47	16	282.3	166.78	16		
34	Proposed	500	47	16	198.83	119.75	16	29.57	28.20
35	Baseline	1000	47	16	596.48	337.03	16		
36	Proposed	1000	47	16	389.81	244.47	16	34.65	27.46

The presented statistics refer to six seeds (N=6) based on the workload size and scheduling strategy. Standard deviation (StdDev) is calculated from the independent runs and shows the degree of variation between the runs. The percentage improvement for each seed is computed as (Baseline-Proposed) / Baseline × 100, and the resulting numbers are combined. Table 3 presents the average values and standard deviations of each parameter for each workload size along with the combined improvement figures.

Table 3 Table of summary

Cloudlets	Metric	Baseline Mean	Baseline StdDev	Proposed Mean	Proposed StdDev	Improvement Mean %	Improvement StdDev %	N
200	Makespan	129.85	11.43	82.97	2.74	35.65	6.54	6
200	AvgCompletion	67.04	4.52	48.87	2.71	26.68	7.97	6
500	Makespan	286.51	48.16	198.14	7.09	28.86	14.42	6
500	AvgCompletion	161.74	18.33	163.64	106.22	-1.42	62.82	6
1000	Makespan	590.20	22.87	342.68	111.05	41.98	18.89	6
1000	AvgCompletion	343.88	7.42	245.70	4.37	28.53	1.79	6

Variability across seeds (42-47) is inevitable since each seed produces a different combination of cloudlet requests, resulting in varied contention scenarios under time-shared execution. The makespan is especially sensitive to a few large cloudlets that could heavily the tail, on the other hand, average completion time is a broader measure of the distribution of completion times. Demand-aware binding as proposed reduces the demand-capacity mismatch leading to better stability most of the time; still, the remaining variance is mainly due to seed-dependent workload composition instead of changes in infrastructure or simulator settings.

6.2 Makespan Reduction

Table 2 provides detailed figures on how the new demand-aware binding method systematically reduces makespan regardless of the workload size. For *200 cloudlets*, the proposed method manages to deliver a makespan of 82.97 ± 2.74, compared to 129.85 ± 11.43 (baseline), which is equivalent to a 35.65% ± 6.54% improvement. The two methods differ not only in the degree but also in the stability of the reduction, which is demonstrated by the lower standard deviation suggested in the proposed method.

For *500 cloudlets*, the baseline makespan is 286.51 ± 48.16 while that of the proposed method is 198.14 ± 7.09, which means a 28.86% ± 14.42% improvement. The improvement is always on the positive side for every seed, although it changes in size; one seed gave a result that was almost zero, which raised the spread of the improvements.

 ISSN:2302-8149
e-ISSN:2540-9719

The senario of *1000 cloudlets* see the makespan being reduced from 590.20 ± 22.87 to 342.68 ± 111.05, thus the average improvement comes out as 41.98% ± 18.89%. In absolute terms, decrease is considerable, hence the benefits at the level of very high workload intensity are quite substantial as suggested by the results. However, variance also increases at this point; a seed-by-seed analysis shows that one single seed was able to deliver a very high gain, thus lifting both the average improvement and its standard deviation. Therefore, the method always yields a better makespan; however, the extent of the benefit it may be more seed-sensitive at very high loads.

The aforementioned trend is consistent with the functioning of the approach. When binding decisions are informed by demand data, the scheduler is thus less likely to commit mismatches, which are a major cause of resource contention under time-shared operation, such that the completion of the last cloudlets is getting faster. As workload size increases, this effect becomes more pronounced since binding errors are being accumulated and therefore have a stronger impact. However, the situation where a workload is merely random and happens to align with the baseline policy, the relative gain would be smaller. Figure 4 reports the absolute makespan values, while Figure 5 summarizes the corresponding percentage improvements.

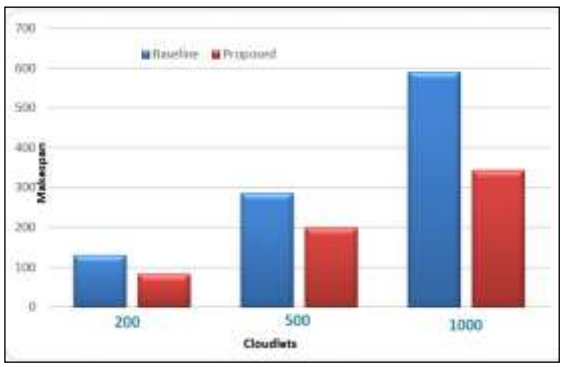


Figure 4 Makespan baseline vs proposed

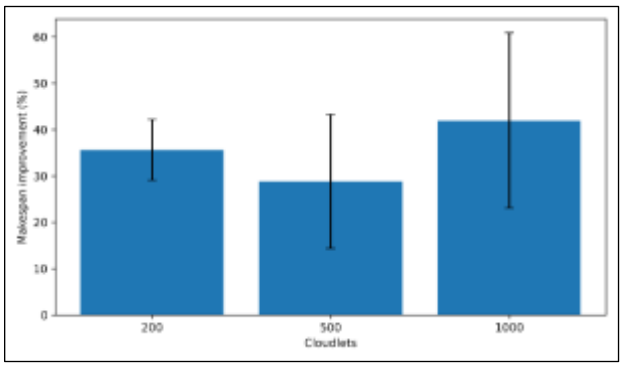


Figure 5 Makespan improvement %

6.3 Average Completion Time

Table 2 reveals that the AvgCompletion results mostly mirror the makespan trend for 200 and 1000 cloudlets. The enhancements of 26.68% and 28.53%, respectively, confirm that demand-aware binding not only cuts the tail completion time but also reduces the overall completion latency. The 500-cloudlet case stands out as an exception, where the average improvement turns out to be slightly negative (-1.42%) and the proposed StdDev is at an extraordinarily high level (106.22), exposing a strong variability component driven by at least one outlier seed.

At *200 cloudlets*, the baseline average completion time stands at 67.04 ± 4.52, with the proposed method bringing it down to 48.87 ± 2.71, which is a 26.68% ± 7.97% improvement. This indicates an enhancement not only in the makespan but also in the central tendency of task completion. In the case of *1000 cloudlets*, the base level is measured at 343.88 ± 7.42 and the state-of-the-art at 245.70 ± 4.37, which translates to an increase of 28.53% ± 1.79%. The magnitude of the gain here is great and statistically solid although the makespan variance is higher. On the other hand, at *500 cloudlets*, the baseline average completion time registers at 161.74 ± 18.33, but the proposed method shows 163.64 ± 106.22, which amounts to, 1.42% ± 62.82%.

This is a senario with only a negative mean improvement. The huge variation shows an unstable situation caused by a single deteriorated run, while the majority of seeds still receive positive gains that are in line with the makespan improvements. The 500-cloudlet case of this behavior is quite a revealing insight. Even though the proposed binding rule at this workload scenario keeps on increasing the makespan, thus the tail latency is lowered, the average completion time is rendered unstable due to one unexpected run. While makespan is influenced by the last finishing tasks only, average completion is more reflective of the overall distribution of task finishing times. Therefore, by

 ISSN:2302-8149
e-ISSN:2540-9719

disclosing both metrics, the performance is roughly characterized in a much more comprehensive and transparent manner as opposed to only depending on makespan. Figure 6 gives the average completion time results, while Figure 7 illustrates the improvement ratios.

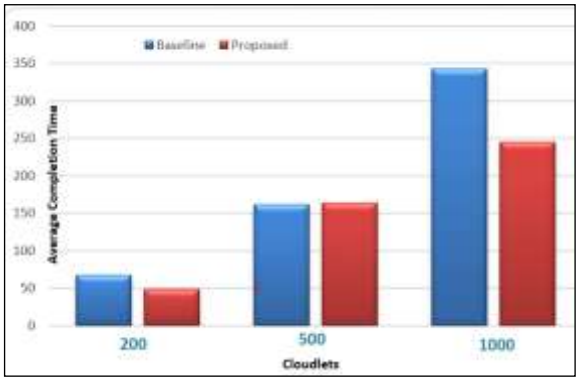


Figure 6 AvgCompletion baseline vs proposed

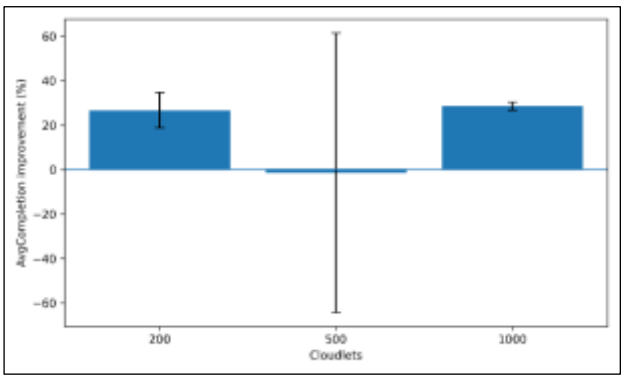


Figure 7 AvgCompletion improvement %

Figure 8 illustrates the relative improvement in performance of the demand-aware VM selection method proposed over the baseline method, represented at three different workload sizes (200, 500, and 1000 cloudlets). For each workload, the improvement rate was first calculated for each seed by using the formula (Baseline - Proposed)/Baseline × 100, after which the results were combined for seeds 42, 47 (N = 6). Therefore, the graph shows how the profit of making a demand awareness factor at binding time depends on the degree of workload and also shows the different patterns for the makespan reduction against the average completion-time behavior at the same fixed infrastructure and execution policies.

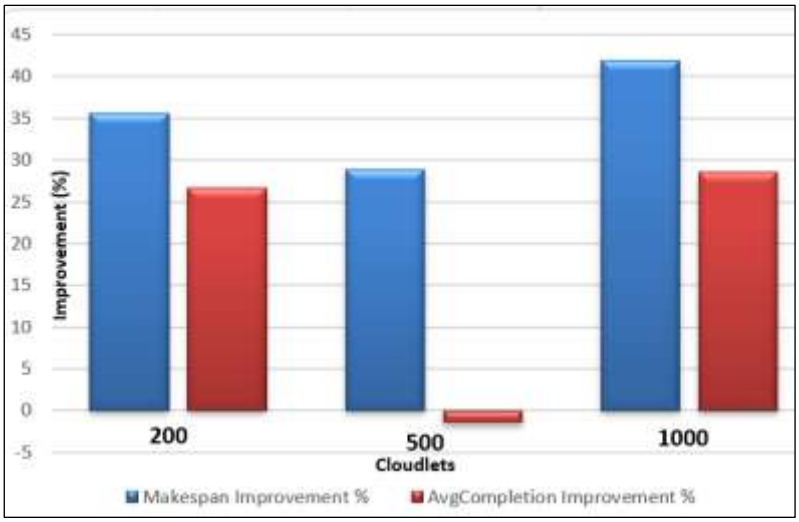


Figure 8 Relative improvement in performance of demand-aware VM selection across different workloads

7 Conclusion

In this paper, we considered a lightweight, demand-aware cloudlet-to-VM binding policy that can potentially deliver measurable improvements in cloud task scheduling results in CloudSim Plus without incurring the training overhead and reproducibility challenges typical of learning-based or heavy metaheuristic schedulers. We started by using the default time-shared baseline as a comparator and then realized a demand-aware VM selection scheme which takes per-cloudlet demand into account explicitly when selecting a VM at binding time, while the rest of the simulation configuration was kept intact. We ran the experiment for six different random seeds (42-47) and three workload sizes (200, 500, and 1000 cloudlets), and the proposed method consistently produced makespan reductions that averaged 35.65%, 28.86%, and 41.98%, respectively. The average completion time

 ISSN:2302-8149
e-ISSN:2540-9719

was also better for 200 and 1000 cloudlets, respectively, with mean drops of 26.68% and 28.53%. This indicates that the acquisitions extend beyond just one tail metric. The 500-cloudlet AvgCompletion data exhibited considerable fluctuation and a slightly negative average improvement. Hence, demand-aware binding might be effective for global completion time but it might not be as effective for specific workload compositions and contention patterns where completion-time averaging is used as the criterion. Demand awareness at binding time can be a simple, reproducible, and easily integrable upgrade to a baseline scheduler in CloudSim Plus. Moreover, it can serve as a solid baseline and as a component of more complex frameworks (e.g., trust-aware, multi-objective, or learning-driven schedulers) where cloudlet-to-VM pairing is a feature that can be leveraged.

References

[1] N. A. Sultan, W. Hadeed, and D. Abdullah, "*Smart Task Scheduling for Cloud-based Big Data Systems*," Informatica, Vol. 49, No. 28, 2025.

[2] A. Gunduz, S. Senan, and Z. Gurkas-Aydin, "*A Hybrid DECSO Algorithm for Efficient Multi Objective Task Scheduling in Cloud Computing Environments*," Cluster Computing, Vol. 28, No. 13, pp. 848, 2025.

[3] W. Shen, W. Lin, W. Wu, H. Wu, and K. Li, "*Reinforcement Learning-based Task Scheduling for Heterogeneous Computing in End-Edge-Cloud Environment*," *Cluster Computing*, Vol. 28, 2025, DOI: 10.1007/s10586-024-04828-2.

[4] N. A. Sultan, W. Hadeed, and D. Abdullah, "*Smart Task Scheduling for Cloud-based Big Data Systems*," Informatica, Vol. 49, No. 28, 2025.

[5] X. Hua and L. Zheng, "*Workflow Scheduling in IaaS Clouds with the Optimal Pairing Between Tasks and Virtual Machines*," J. King Saud Univ. Comput. Inf. SCI., Vol. 37, No. 8, pp. 237, 2025.

[6] S. Mangalampalli, G. R. Karri, and A. A. Elngar, "*An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing using Firefly Optimization*," Sensors, Vol. 23, No. 3, pp. 1384, 2023.

[7] S. Chandrasiri and D. Meedeniya, "*Energy-Efficient Dynamic Workflow Scheduling in Cloud Environments using Deep Learning*," Sensors, Vol. 25, No. 5, pp. 1428, 2025.

[8] M. C. Silva Filho, C. C. Monteiro, P. R. M. Inácio, and M. M. Freire, "*A Distributed Virtual-Machine Placement and Migration Approach based on Modern Portfolio Theory: MCS Filho et al.*," J. Netw. Syst. Manag., Vol. 32, No. 1, pp. 2, 2024.

[9] S. Mangalampalli et al., "*Fault Tolerant Trust-based Task Scheduler using Harris Hawks Optimization and Deep Reinforcement Learning in Multi Cloud Environment*," SCI. Rep., Vol. 13, No. 1, pp. 19179, 2023.

[10] J. Pan, Y. Wei, L. Meng, and X. Meng, "*A Dual Scheduling Framework for Task and Resource Allocation in Clouds using Deep Reinforcement Learning*," J. King Saud Univ. Comput. Inf. SCI., Vol. 37, No. 5, pp. 81, 2025.

[11] A. Pradhan and S. K. Bisoy, "*A Novel Load Balancing Technique for Cloud Computing Platform based on PSO*," J. King Saud Univ. Comput. Inf. SCI., Vol. 34, No. 7, pp. 3988-3995, 2022.

[12] S. Mangalampalli, G. R. Karri, and U. Kose, "*Multi Objective Trust Aware Task Scheduling Algorithm in Cloud Computing using Whale Optimization*," J. King Saud Univ. Comput. Inf. SCI., Vol. 35, No. 2, pp. 791-809, 2023.

[13] S. Mangalampalli et al., "*Prioritized Task-Scheduling Algorithm in Cloud Computing using Cat Swarm Optimization*," Sensors, Vol. 23, No. 13, pp. 6155, 2023.

[14] U. K. Lilhore et al., "*QHRMOF: A Quantum-Inspired Hybrid Multi-Objective Framework for Energy-Efficient Task Scheduling and Load Balancing in Cloud Computing*," J. Cloud Comput., Vol. 14, No. 1, pp. 54, 2025.

[15] A. Yadav and A. Sharma, "*Enhancing Cloud Orchestration using Hybrid Scheduling for QoS-Aware and Energy-Efficient Task Allocation*," Cluster Computing, Vol. 28, No. 13, pp. 859, 2025.

[16] S. Mangalampalli et al., "*Efficient Deep Reinforcement Learning based Task Scheduler in Multi Cloud Environment*," SCI. Rep., Vol. 14, No. 1, pp. 21850, 2024.