IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 October 2024, accepted 26 November 2024, date of publication 29 November 2024,
date of current version 13 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3508793

RESEARCH ARTICLE

Optimizing Cloud Computing Performance With an Enhanced Dynamic Load Balancing Algorithm for Superior Task Allocation

RAIYMBEK ZHANUZAK[1], MOHAMMED ALAA ALA'ANZY[1], (Member, IEEE), MOHAMED OTHMAN[2,3], (Senior Member, IEEE), AND ABDULMOHSEN ALGARNI[4]

[1]Department of Computer Science, SDU University, 040900 Almaty, Kazakhstan
[2]Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia
[3]Laboratory of Computational Science and Mathematical Physics, Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia
[4]Department of Computer Science, King Khalid University, Abha 61421, Saudi Arabia

Corresponding author: Mohammed Alaa Ala'anzy (m.alanzy@ieee.org)

This work was supported by the Deanship of Scientific Research, King Khalid University, under Grant R.G.P.2/93/45.

ABSTRACT Cloud computing, particularly within the Infrastructure as a Service (IaaS) model, faces significant challenges in workload distribution due to limited resource availability and virtual machines (VMs). Efficient task allocation and load balancing are crucial to avoiding overloading or under-loading scenarios that can lead to execution delays or machine failures. This paper presents an Enhanced Dynamic Load Balancing (EDLB) algorithm designed to optimise task scheduling and resource allocation in cloud environments. Unlike benchmark algorithms that rely on static VM selection or post-hoc relocation of cloudlets, the EDLB algorithm dynamically identifies optimal cloudlet placement in real-time. Our approach proactively allocates cloudlets to VMs based on current system states and Service Level Agreement (SLA) deadlines, thereby preemptively addressing potential SLA violations. Additionally, if a VM cannot meet the deadline of the cloudlet, the algorithm redirects the cloudlet to a secondary data centre and reconfigures CPU resources among VMs to ensure optimal allocation. Evaluations using CloudSim simulations demonstrate that the EDLB algorithm achieves substantial average improvements over benchmark algorithm and the-state-of-the-art algorithm, including a 59.46% reduction in total makespan, a 12.70% reduction in average makespan, a 22.46% reduction in execution time, and a 3.10% increase in resource utilisation. Furthermore, the EDLB algorithm enhances load balancing by 46.46%. These results highlight the effectiveness of the EDLB algorithm in addressing critical load balancing issues and surpassing existing methods. This research contributes to the field by introducing a novel approach that significantly improves performance metrics and operational efficiency in cloud computing environments.

INDEX TERMS Cloud computing, task scheduling, load balancing, resource allocation, CloudSim simulation.

I. INTRODUCTION

Cloud Computing technology plays a vital role in modern business operations, offering a range of services such as software accessible through web browsers and platforms for developing cloud-based applications.

Cloud Service Providers (CSPs) play a crucial role in the infrastructure domain by overseeing the management of back-end operations, such as the maintenance of data centres and servers. The focus on the IaaS model within cloud computing has significantly increased in recent years. IaaS provides scalable and flexible virtualised computing resources, allowing users to build and manage IT infrastructure without the need for physical hardware. This importance of the model is underscored by the projected growth of the

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

global IaaS market, expected to rise from USD 146.2 billion in 2023 to USD 461.9 billion by 2029 [1]. With the rise of cloud computing, many service providers have built extensive data centres to supply essential resources like cloud server [2].

Applications based in the cloud are significantly reliant on virtualisation, an essential component in their functioning [3]. Inefficient management of the migration process and allocation of VM resources can significantly impact the performance of client services where the cloud performance is a major challenge in cloud computing [4].

In the cloud setting, users access services through requests that are executed on VMs [5]. The CSP should provide services that benefit businesses and enhance user satisfaction [6]. Therefore, our main objective is to propose load balancing algorithm focuses primarily on the IaaS model, addressing back-end issues of cloud computing technology, such as server workload. Figure 1 illustrate the assigning of the tasks among different physical machine to achieve efficient load balancing.

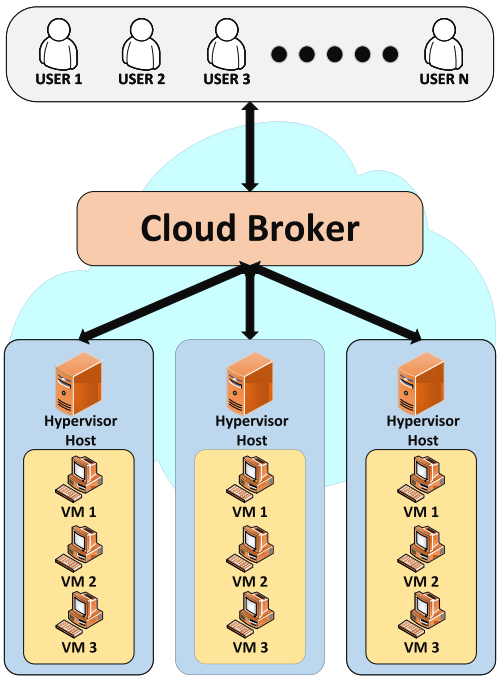

FIGURE 1. Load balancing model in IaaS.

In a typical cloud environment, there are two main components: the front-end, which users access through an Internet connection, and the back-end, where cloud service models are managed [7]. The back-end consists of multiple physical machines stored in data centres, commonly known as servers. User requests from applications are dynamically scheduled, and virtualisation is used to allocate the necessary resources to clients. Whereas the virtualisation helps with balancing the loads across the system, including scheduling and efficient resource allocation [8]. Both CSP and users in the cloud can benefit from virtualisation and dynamic task scheduling techniques. Hence, effective scheduling can significantly reduce execution time and improve resource utilisation in cloud-based applications.

Task scheduling is closely related to workload balancing, as user requests are routed through a cloud broker. This highlights the importance of developing efficient algorithms for task allocation to suitable VMs, considering critical parameters such as deadlines which ensure the provision of high-quality services. It is essential to execute and complete user requests within the specific requirements outlined in the SLA [9], [10], [11]. User requests are transmitted via the internet and stored in VMs. In every delivery model, CSP must maintain Quality of Service (QoS) by ensuring requests are fulfilled within set deadlines. The effectiveness of the scheduling policy is significantly impacts workload balance among VMs and servers. Thus, to achieve efficient scheduling and resource utilisation, a dynamic load balancer (DLB) can be developed and implemented. Cloud computing heavily relies on the VM monitor (VMM) or hypervisor, such as VMware, situated in the host layer. The VMM manages multiple VMs on a single hardware layer [12].

The performance of cloud-based applications can be compromised by inappropriate scheduling techniques or inefficient task mapping to the correct VMs/resources, given the crucial role of virtualisation in cloud technology [13], [14]. This imbalance has the potential to strain server workloads. Hence, there exists an opportunity in cloud computing technology to improve resource-to-task mapping by focusing on scheduling. It is important to consider key QoS parameters to achieve efficient resource utilisation without breaching the SLA, while taking into account constraints such as deadlines and priority [15].

Therefore, this research aims to improve resource allocation within the IaaS model by balancing resources for clients and handling user requests on servers. The contributions of this paper are:

- Introducing the EDLB algorithm for optimising task scheduling and load distribution in cloud environments, integrating preemptive scheduling to mitigate SLA violations.
- Offering a novel approach in dynamic load balancing by integrating preemptive scheduling and adaptive resource allocation based on SLA requirements.
- Evaluation of the proposed algorithm through CloudSim simulations to efficiently reduce total and average makespan and execution time while maintaining high resource utilisation and balancing percentages across varying VMs and cloudlets configurations.
- Evaluating key metrics to quantify the efficiency of the EDLB algorithm, scalability, and load balancing capabilities, showcasing its superiority over benchmark algorithms under diverse workloads.

The structure of the paper is as follows: Section II delves into related work, explaining concepts of load balancing

and task scheduling, recent research by other authors, and highlighting strengths, weaknesses, and future directions. Section III outlines the proposed EDLB algorithm, including the framework, flowchart, pseudocode, implementation details like simulation setup, and performance metrics. Section IV presents the discussion and results from the experiment, with a brief comparison to existing related work. Finally, Section V wraps up the paper by summarising its concept and content, and offering suggestions for future improvements in the algorithm.

II. RELATED WORK

In recent years, significant progress has been made in the field of task scheduling and load balancing within computing systems. This progress is driven by the increasing complexity of computational tasks and the continuous evolution of hardware architectures. Efficiently managing resources and distributing workloads across different computing elements has become crucial for optimising system performance and resource utilisation.

In this section, we explore existing research on task scheduling and load balancing. We discuss both general methodologies and specific approaches designed to tackle various challenges in this domain. Our discussion begins with an overview of foundational concepts and widely adopted techniques, providing a comprehensive understanding of the underlying principles and strategies.

Subsequently, we focus on examining the most relevant and beneficial work related to task scheduling and load balancing. By synthesising key findings and highlighting notable contributions, we aim to shed light on the current state-of-the-art approaches and identify promising directions for future research. Through this exploration, we offer a holistic perspective on ongoing efforts to enhance the efficiency, scalability, and robustness of task scheduling and load balancing mechanisms in computing systems.

The interdependence of the load balancing and task scheduling was highlighted in this section. While, we will delve into the foundational principles of these aspects of optimising cloud resources. Load balancing stands as a pivotal method for maximising the efficiency of VM resources in the cloud computing environment, ensuring an equitable distribution of workload and effective resource utilisation [16]. Dynamic workload distribution among nodes in the cloud environment is facilitated by load balancing, which not only enhances user satisfaction but also optimises resource allocation. Addressing load balancing issues is crucial for enhancing cloud application performance.

Task scheduling, a major objective of load balancing, becomes particularly pertinent as the number of cloud clients increases, potentially leading to improper task scheduling. Consequently, task scheduling issues require resolution through the implementation of algorithms [13], [17]. Task scheduling involves efficiently executing tasks to fully utilise system resources [18]. In the cloud environment, where users extensively utilise virtualised resources, manual task allocation is impractical.

Cloud computing services have become essential for major corporations like Google and Amazon, providing flexible data transfer and continuous streaming capabilities. However, the algorithms supporting these operations may encounter difficulties as the number of clients increases. In cloud computing technology, load balancing is essential to avoid task delays for users and reduce response times.

As technology advances and companies of all sizes adopt cloud services. The CSPs face challenges due to imbalanced load situations, which can hinder the delivery of high-quality services to users. Issues such as high makespan time can negatively impact performance and pose risks to SLAs. These violations can lead to starvation issues, where the system is overloaded and incoming tasks cannot be appropriately serviced, potentially resulting in rejection. It is essential to tackle these issues to minimise SLA violations by cloud providers for organisations. Several factors contribute to load unbalancing issues in IaaS clouds, including improper mapping of tasks to VMs, inadequate scheduling processes, varying task requirements for heterogeneous user tasks, and uneven distribution of tasks to VMs.

This paper aims to resolve the aforementioned issues in the IaaS cloud platform by proposing a dynamic task scheduling algorithm that takes into account important task requirements such as deadline and completion time, which are highly significant as QoS factors. Through proper scheduling and avoidance of VM violations, the algorithm ensures a balanced workload in the cloud system.

We now offer a synopsis of the current algorithms in load balancing and task scheduling. While numerous recent algorithms aim to improve task scheduling and load balancing, they still face limitations due to the use of fundamental algorithms in the process of assigning the cloudlets or selecting the VMs, which can lead to increased waiting times or makespan in task scheduling.

In [19], they propose a dynamic load balancing algorithm to minimise makespan time and efficiently utilise resources. This algorithm employs the bubble sort algorithm to sort tasks based on length and processing speed, allocating them to VMs in a First-Come-First-Serve order. While effective in optimising resources and reducing makespan, this approach does not consider priority or QoS parameters like deadline. Also, the bubble sort algorithm is considered a time consumer as its time complexity is $O(n^2)$.

Using Min-Min algorithm as in [20], where authors propose an enhanced load-balanced Min-Min (ELBMM) algorithm to optimise resources. This algorithm seeks tasks with the minimum execution time and assigns them to the VM with the minimum completion time, effectively enhancing the Min-Min algorithm and reducing utilisation costs and system throughput. A three-layer strategy for cloud network load balancing is introduced by [21]. This strategy incorporates both opportunistic load balancing (OLB) and load balance Min-Min (LBMM) techniques. While this

method enhances task scheduling in OLB via a hierarchical network structure, the complexity of multiple layers could lead to reduced processing speed. Another hybrid algorithm that used Min-Min approach for the resource-based load balanced Min-Min (RBLMM) [21]. The algorithm aims to reduce makespan and balance workload on VMs. RBLMM calculates makespan time after resource allocation, using this value to define a threshold. Results show a significant reduction in makespan time compared to the traditional Min-Min algorithm, although the approach lacks prioritisation for tasks or VMs.

In [22], the authors introduce the ED-LB algorithm to address load imbalance in IaaS clouds, primarily through selecting the best-fit VM for each task and dynamically allocating it to the least-loaded server. While this approach emphasises resource optimisation, it lacks preemptive scheduling and real-time monitoring for SLA compliance. By contrast, our EDLB algorithm dynamically reallocates resources based on SLA demands, integrates preemptive scheduling to mitigate potential SLA violations, and proactively redirects tasks across data centres when nearing deadline constraints. Additionally, our EDLB demonstrates enhanced scalability in real-world fluctuating environments, as opposed to the static synthetic settings used in ED-LB evaluations. This approach allows our algorithm to maintain deadline adherence and SLA compliance effectively, achieving notable improvements in makespan and resource utilisation.

In [23], the authors implement an enhanced dynamic load balancer based on the HTV load balancer, which allows users to input various parameters such as the number of hosts, VMs, job requests, and application types to prioritise job execution. The proposed algorithm achieves better performance and resource utilisation compared to the HTV load balancer. However, it lacks real-time dynamic allocation of resources across multiple data centres. Their approach dynamically generates a queue based on load and performance factors and uses dynamic round-robin scheduling. In contrast, our work adapts to fluctuating workloads and traffic in real-time, optimising SLA adherence and ensuring more efficient resource management across multiple cloud environments.

For improved QoSs, authors in [24] introduce a QoS-based algorithm allocating cloudlets with an enhanced balancing technique. While beneficial for balancing workload and decreasing completion time, it may lead to high makespan values for VMs and hosts, particularly in large-scale environments. A Grouped Tasks Scheduling (GTS) algorithm categorises tasks into groups based on QoS parameters, improving latency for urgent tasks [25]. However, it may not be suitable for tasks dependent on a specific order or other scheduling requirements.

An enhancement to the traditional Shortest Job First (SJF) scheduling algorithm is proposed by authors in [26], resolving issues of starvation by allocating longer tasks to high-response VM. However, their approach overlooks the availability and current load/status of the VM before task allocation, and tasks are scheduled solely based on task length, lacking priority.

In [27], the CMLB load balancing algorithm is presented, using the Dragonfly optimisation algorithm to determine optimal thresholds for reallocation of tasks to VMs. The algorithm exhibits better performance with fewer migrations compared to methods such as Honey-Bee and dynamic LB. In [28] proposed a novel load balancing algorithm for cloud computing using a hybrid approach combining Krill Herd, Whale Optimisation, and Deep Belief Neural Network. The method aims to optimise resource usage, reduce execution time, and improve overall system performance. It claims superiority over existing load balancing techniques based on comparative results. The authors of [28] have considered the hosts only to optimise load balancing, while they failed to consider the cloudlets in each VM.

The algorithm proposed in [29] combines deep learning with Particle Swarm Optimisation (PSO) and Genetic Algorithm (GA) to address dynamic workload balancing in cloud computing. Nevertheless, the algorithm focused solely on the hosts, overlooking the VMs execution time.

Authors in [30] propose a credit-based resource-aware load balancing scheduling algorithm (HO-CB-RALB-SA) for cloud computing. This algorithm leverages a hybrid of the Walrus Optimisation Algorithm (WOA) and Lyrebird Optimisation Algorithm (LOA) to optimise load distribution and resource utilisation across VMs. The framework balances the system load by evaluating the processing capacity of each VM and redistributing tasks to maintain equilibrium. The authors claim that their method outperforms existing models by efficiently managing resources and ensuring balanced workloads. The reliance on VM processing power and a credit-based system may not provide the flexibility needed to adapt to dynamic and unpredictable workloads, which are common in cloud computing. This could lead to inefficiencies, particularly when workload patterns shift rapidly, as the algorithm may struggle to maintain optimal load distribution under such conditions.

A dynamic task scheduling in the IaaS cloud platform presented by [31], considers important task requirements such as deadline and completion time as crucial QoS factors. It aims to improve cloud performance by balancing workload and maximising resource utilisation through appropriate task scheduling and load balancing techniques. The algorithm checks completion times against deadlines and reconfigure VM priorities based on CPU status to address SLA violations. It also involves workload relocation if necessary to ensure efficient task execution. Their algorithm aims to reduce makespan, execution time, and improve resource utilisation in cloud applications by addressing load balancing issues and considering task requirements. However, the reliance of the algorithm on static VM selection without considering real-time execution times may lead to sub-optimal resource allocation and inefficient task scheduling. By overlooking the dynamic nature of workload requirements and VM performance, the algorithm may fail to adapt effectively

TABLE 1. Comparison of related work on task scheduling and load balancing algorithms.

Ref.	Methodology	Strengths	Drawbacks
[19]	Dynamic load balancing using Bubble Sort and FCFS	Minimises makespan and improves resource utilisation	High time complexity ($O(n^2)$); lacks priority or QoS consideration
[20]	Enhanced Min-Min algorithm	Reduces utilisation cost and improves throughput	Ignores task prioritisation and complex multi-layer strategies may slow down processing
[22]	ED-LB: Best-fit VM selection with dynamic allocation	Emphasises resource optimisation and load balancing	Lacks preemptive scheduling and real-time SLA monitoring
[23]	HTV-based load balancer with dynamic round-robin scheduling	Better performance and resource utilisation than HTV	No real-time dynamic allocation across multiple data centres
[24]	QoS-based cloudlet allocation	Balances workload and decreases completion time	High makespan in large-scale environments
[25]	Grouped tasks scheduling based on QoS	Improves latency for urgent tasks	Unsuitable for dependent or order-specific tasks
[26]	Enhanced SJF scheduling	Resolves starvation by allocating longer tasks to high-response VMs	Ignores current load/status of VMs; lacks task priority consideration
[27]	CMLB using Dragonfly optimisation	Fewer migrations and better performance compared to Honey-Bee	Threshold determination depends heavily on the algorithm parameters
[28]	Hybrid Krill Herd, Whale Optimisation, and DBNN	Optimises resource usage and reduces execution time	Focuses only on hosts without considering cloudlets in VMs
[29]	Deep learning with PSO and GA for dynamic workload balancing	Addresses workload balancing dynamically	Focuses solely on hosts, ignoring VM execution time
[30]	HO-CB-RALB-SA: Hybrid WOA and LOA with credit-based system	Efficient resource management and balanced workloads	Inflexible with dynamic workloads; struggles under rapid workload shifts
[31]	Dynamic task scheduling with deadline and completion time	Reduces makespan and improves resource utilisation	Static VM selection leads to suboptimal resource allocation
Ours	Dynamic cloudlet allocation with SLA-based redirection	Ensures SLA compliance, reduces makespan, and improves resource allocation	Requires resource reconfiguration across data centres, which may add latency

to changing conditions in cloud environments. Table 1 summarises the methodology, strengths, and drawbacks of the previous studies.

Therefore, this paper aims to improve resource allocation within the IaaS model by balancing resources for clients and handling user requests on servers. Our algorithm dynamically allocates cloudlets to VMs based on completion time, while considering the cloudlet deadline, which is crucial for SLA requirements. Based on the available processor of a VM where if it is insufficient to complete a cloudlet before its deadline, the cloudlet is redirected to a second data centre. In the second data centre, the CPU resources are reconfigured among the VMs according to their needs, allowing for optimal cloudlet allocation. This approach helps achieve a balance between VMs and improves overall resource allocation efficiency. This collaboration of VMs is a key aspect that many existing algorithms tend to overlook. In short, [31] has randomly allocated cloudlets to VMs and then checked for any SLA violations, which are cloudlets that cannot be executed before their deadlines. If a violation occurs, the algorithm relocates the cloudlet to another VM. If this is still not sufficient, the CPU will be reconfigured. In contrast, our approach instantly identifies the best placement for cloudlets, ensuring compliance with the SLA.

III. THE PROPOSED EDLB ALGORITHM

This section provides an overview of the EDLB algorithm that has been introduced. It explains the underlying assumptions, presents the pseudocode, and concludes with a flowchart. The main goal of our algorithm is to improve the overall performance of cloud systems with task scheduling and load balancing. Using all available CPUs on the machines, the algorithm strategically schedules tasks to minimise total and average makespan, execution time, and optimise resource utilisation.

As depicted in the algorithm steps, both inputs and outputs are integral to every step. In this study, the primary input is a list of cloudlets with randomly assigned task length and deadline, which are crucial elements in the SLA metric. The SLA serves as a significant metric for CSPs, indicating the reduction of SLA violation factors, including deadline constraints and priorities. The main objective or output of the algorithm is to achieve a balanced workload among VMs in cloud systems, facilitating reallocation in cases of SLA violations.

The EDLB algorithm starts by allocating the minimum required processing capacity that represented by Million Instructions Per Second (MIPS) to operate a VM, referred to as NM. As the execution progresses, this allocation dynamically adapts to ensure timely completion of cloudlets while meeting SLA requirements. The primary goal is to efficiently distribute computational resources to maximise performance and meet deadlines. Figure 2 depicts the flowchart of the EDLB algorithm, which illustrates the steps involved in assigning cloudlets to VMs to optimise processing efficiency and prevent overloading of any single VM. Factors such as VM completion time, available processors, and cloudlet execution deadlines are taken into consideration. The available processing is measured by MIPS. Table 2 was provided to show key symbols used in equations.

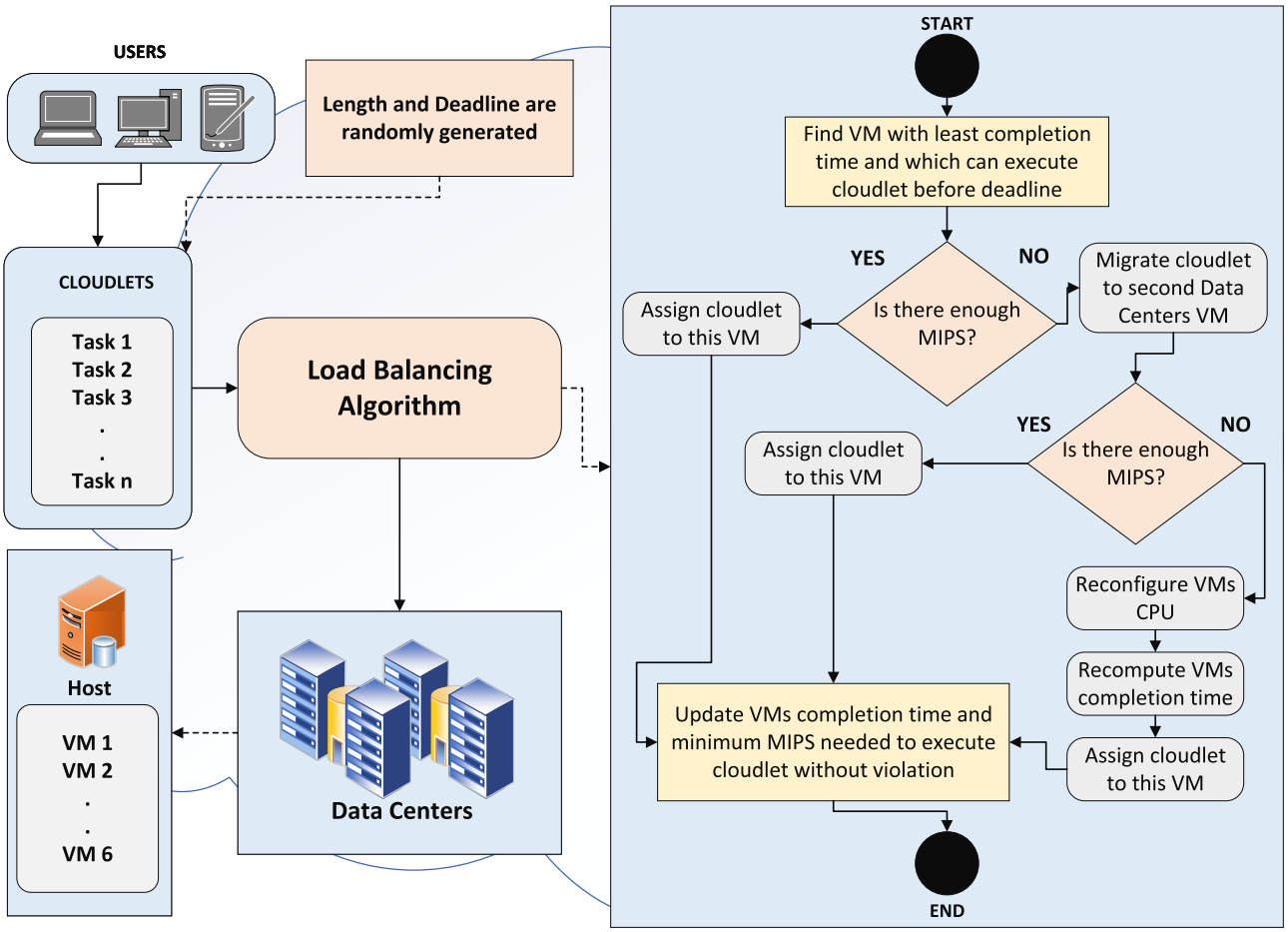

FIGURE 2. Cloudlet assignment flowchart for efficient resource utilisation.

TABLE 2. Key symbols used.

Symbol	Explanation
J	A collection of cloudlets
K	A collection of VMs of second data centre
I	A collection of all VMs
$CTVM_i$	Completion time of VM_i
CTC_j	Completion time of $cloudlet_j$
L_j	Length of cloudlet
D_j	Deadline of cloudlet
AM	Available MIPS of second data centre
N	MIPS required to execute cloudlet before deadline
NM	Minimum required MIPS to execute VM
CG_k	MIPS that VM_k can give without violating SLA agreements
$MIPS_i$	resources of VM_i
N_j	MIPS required to execute a cloudlet within its deadline
TG	MIPS required for VM to execute cloudlet except its own MIPS
P	Portions that each VM should give
S	TG without CG_k of VMs that can not give their P
α	Number of VMs in second data centre
β	Number of VMs that can not give their P
n	Number of cloudlets
m	Number of VMs

Initially, the EDLB algorithm carefully selects VMs with the shortest completion time ($CTVM$) as in Equation (1) while comparing the completion time of cloudlets (CTC) against their deadlines.

$$CTVM_i = \sum CTC_{ij}. \tag{1}$$

If a VM with sufficient resources is found, the cloudlet is promptly assigned to it. CTC can be calculated as shown in Equation (2),

$$CTC_j = \frac{L_j}{MIPS_i}, \tag{2}$$

where L is length of cloudlet and let $i \in I$ where $(i_1, i_2, \ldots i_m)$ and let $j \in J$ where $(j_1, j_2, \ldots, j_n)$.

In cases where no suitable VM is available, the algorithm orchestrates the relocation of cloudlets to a secondary data centre, streamlining operations within a single data centre for improved efficiency.

After relocation, the EDLB algorithm calculates several crucial values to facilitate resource allocation. These include the Available MIPS (AM), indicating the surplus MIPS across all VMs compared to the total NM. Additionally, CG represents the MIPS that VMs can release without violating constraints, while N quantifies the MIPS

required to execute a cloudlet within its deadline as in Equations (3), (4) and (5), where D is a deadline of the cloudlet and $k \in K$ where $(k_1, k_2, \ldots, k_{\frac{m}{2}})$.

$$AM = \sum MIPS_k - \sum NM_k. \tag{3}$$

$$CG_k = MIPS_k - NM_k. \tag{4}$$

$$N_j = L_j / D_j. \tag{5}$$

Next, EDLB algorithm evaluates whether the MIPS of the chosen VM are adequate for executing the cloudlet. If affirmative, allocation proceeds; otherwise, the MIPS reconfiguration process begins among VMs to meet the demand.

During reconfiguration, the EDLB algorithm ensures an equitable distribution of resources to ensure each VM contributes fairly. This involves determining TG and S. The TG represents the difference between the required MIPS and the current MIPS of VMs as represented in Equation (6),

$$TG = N - MIPS_{min}. \tag{6}$$

While S, initially identical, it facilitates balanced reallocation and after that, it can be updated, as in Equation (7).

$$S = S_{old} - CG_k. \tag{7}$$

If a VM cannot fully surrender its part, adjustments are made to the S value accordingly.

The number of VMs that unable to comply with the allocation is denoted by β. And the HE is a boolean indicating whether the VM has a sufficient MIPS to allocate its resources. In the reconfiguration phase, VMs that cannot meet their designated MIPS capacity contribute whatever resources they can, while the surplus resources are evenly distributed among the remaining VMs based on S as in (8),

$$P = \frac{S}{\alpha - 1 - \beta}, \tag{8}$$

where β is the number of VMs that cannot give their portions. Subsequently, after the reconfiguration is completed, the $CTVM$ is adjusted to represent the revised MIPS allocation among the VMs. The process culminates in updating $CTVM$ by incorporating the completion time of the new cloudlet and adjusting NM accordingly to ensure optimal resource allocation and adherence to SLAs.

The pseudocode for the proposed EDLB algorithm is demonstrated in Algorithm 1. It presents the formulas, parameters, and decisions incorporated in this load balancing algorithm.

The novel aspects of the EDLB algorithm within the problem definition primarily lie in the dynamic, adaptive resource allocation strategy. In contrast to the benchmark algorithm, which allocates cloudlets randomly and only checks for SLA violations post-allocation, the EDLB algorithm proactively allocates resources based on the completion time of VMs ($CTVM$) and cloudlets (CTC) in real-time, ensuring that deadlines are met and computational resources are efficiently distributed from the start.

Algorithm 1 Enhanced Dynamic Load Balancing Algorithm

```
Input: List of cloudlets with random length and deadline.
Output: Mapping of cloudlets to the appropriate VM.
Adjust NM_i to all VMs with value 1000
for j in J do
    find VM with least CTVM (VM_min)
    calculate N with Equation (5)
    if VM_min has enough MIPS to execute cloudlet before deadline
    (MIPS_min >= N) then
        allocate cloudlet_j to VM_min
    end
    else
        set min to α
        if VM_min has enough MIPS to execute cloudlet before
        deadline (MIPS_min >= N) then
            Allocate cloudlet_j to VM_min
            break
        end
        Calculate AM and TG with Equations (3), (6) and CG for
        each VM with Equation (4)
        Determine S
        else if AM is enough to execute cloudlet before deadline (AM
        >= N ) then
            Adjust N MIPS to VM_min
            Initialise β for k in K do
                if VM_k do not have its portion to give (CG_k < TG)
                then
                    Set HE_k to False Update S with Equation (7)
                    Update β to +1
                end
                else
                    Set HE_k to True
                end
            end
            Calculate P with Equation (8)
            for k in K do
                if VM_k have enough MIPS to give (HE_k is True)
                then
                    Give their portion to VM_min
                end
                else
                    Give CG_k to VM_min
                end
            end
            for k in K do
                while u <= j do
                    if cloudlet_u allocated to VM_k then
                        Calculate CTVM_k by using Equation (1)
                    end
                end
            end
            Allocate cloudlet_j to VM_min
        end
        else
            Allocate cloudlet_j to VM_min
        end
    end
    Update CTVM_min and NM_min
end
```

The EDLB algorithm introduces two key innovations:

1) Completion Time-Based Allocation: Instead of relying on random allocation, as seen in the benchmark, our algorithm selects VMs based on the shortest completion time ($CTVM$), as defined in Equation (1). This ensures that cloudlets are strategically assigned to VMs capable of completing them most efficiently, minimising overall makespan and avoiding workload imbalances.

2) Real-Time Adaptation: While the benchmark algorithm simply adds MIPS to underperforming VMs when SLA violations occur, our approach reallocates MIPS from VMs with surplus capacity to those needing additional resources. This real-time adjustment optimises resource usage and prevents idle VMs from waiting while others remain overloaded.
3) Furthermore, *CTC*, which measures the completion time of each cloudlet based on its length and the processing capacity of the VM (MIPS), as described in Equation (2), allows us to maintain a more granular level of control during allocation.
4) Available MIPS (AM): This represents the total surplus processing capacity across all VMs after meeting the minimum required MIPS (NM) to operate each VM. The calculation for AM is presented in Equation (3).
5) MIPS that Each VM Can Release (CG): After ensuring that a VM has sufficient capacity to handle its tasks, the remaining MIPS can be released for redistribution to other VMs that may need additional resources to meet deadlines. This is captured by Equation (4).

IV. PERFORMANCE EVALUATION

A. SIMULATION TOOL

Our algorithm is implemented throughout CloudSim simulation toolkit, which has gained significant popularity among researchers and developers in the current cloud-related research landscape. It effectively reduces the need for and costs associated with acquiring computing facilities for performance assessment and research modelling. This simulation tool serves as an external framework that can be easily downloaded and integrated into programming environments such as Eclipse, NetBeans IDE, and others. In order to conduct the simulation, the cloud computing environment CloudSim was integrated into the Eclipse IDE for Java developers running the Windows 10 operating system.

B. PERFORMANCE METRICS

A set of metrics was used to measure the algorithm efficiency such as total makespan, average makespan, execution time, resource utilisation, and balancing percentage.

These five performance metrics used in this approach were chosen to offer a comprehensive evaluation of the efficiency of the proposed algorithm. Total makespan and average makespan show the efficiency of the algorithm by measuring the total time and average time required to complete the cloudlets, respectively. Execution time provides insight into the speed with which individual tasks are processed, reflecting the performance of the proposed algorithm at a finer level. Resource utilisation assesses how well resources are managed and allocated, ensuring that available resources are used efficiently. Finally, the balancing percentage evaluates the distribution of workloads across VMs, ensuring that no single VM is overburdened and that resources are equally distributed. Together, these metrics provide a full view of the ability of the proposed algorithm to manage efficiency, resource allocation, and workload distribution.

1) Total Makespan (TMT): This metric represents the maximum makespan among all VMs. It provides insight into the overall efficiency of the algorithm in managing the completion times across different VMs. The calculation is based on the maximum completion time of cloudlets for each VM as in Equation (9) [32], [33]:

$$TMT = Max(MT). \tag{9}$$

2) Makespan (MT): This represents the time taken to schedule a cloudlet. It is a crucial metric for evaluating the efficiency of scheduling algorithms in terms of time [34], [35]. To enhance the execution of tasks and free up resources for other tasks, it is desirable to minimise makespan. The calculation involves cloudlet completion time (CT) and the number of VMs (m) as in Equations (10) and (11) [36]:

$$MT = Max(CT) \tag{10}$$

$$MT_{avg} = \frac{\sum Max(CT)}{m} \tag{11}$$

3) Execution Time (ExT): This metric measures the time taken to execute specified tasks on a VM [26]. A reduction in execution time contributes to improved algorithm performance. The calculation incorporates cloudlet Actual CPU Time (AcT) and the number of cloudlets (n) as in Equations (12) and (13) [36]:

$$ExT = AcT \tag{12}$$

$$ExT_{avg} = \frac{\sum AcT}{n} \tag{13}$$

4) Resource Utilisation (RU): This quantitative metric is interdependent on the aforementioned metrics and is assessed to enhance resource efficiency in the cloud environment [33]. ExT represents total execution time, and MT represents total makespan. The average resource utilisation indicates the efficiency of the EDLB algorithm in terms of CPU utilisation. The metric ranges from 0 to 1, with 1 being the best case (indicating 100% resource utilisation), and 0 being the worst case as in Equations (14) and (15) [37]:

$$RU = \frac{ExT}{MT} \tag{14}$$

$$RU_{avg} = \frac{ExT_{avg}}{MT_{avg}} \times 100 \tag{15}$$

5) Balancing Percentage (BP): This metric gauges the average makespan divided by the total makespan. It offers a measure of how well the algorithm distributes the workload among VMs, aiming for a more balanced

utilisation of resources as in Equation (16):

$$BP = \frac{MT_{avg}}{TMT} \times 100 \tag{16}$$

The five performance metrics—total makespan, average makespan, execution time, resource utilisation, and balancing percentage—were chosen to comprehensively evaluate the efficiency of the proposed algorithm. These metrics assess task completion times, resource management, and workload distribution across VMs, providing a well-rounded view of the effectiveness of the proposed algorithm in optimising performance and resource allocation.

C. RESULTS AND DISCUSSIONS

To assess the performance of the proposed algorithm and validate its efficiency, key metrics were selected, including makespan, execution time, and the balancing percentage. The results, obtained through two types of experiments, are discussed in the following subsections.

1) TYPE 1 EXPERIMENT (EDLB ALGORITHM VS. BENCHMARK)

The experiment aims to demonstrate a significant improvement in makespan, execution time, resource utilisation, and percentage of the balance, within a dynamic cloud environment.

To simulate the scenario of scheduling and load balancing in a cloud environment, a virtual representation of entities and computing resources was created to assess the effectiveness of the EDLB algorithm. The experiments were carried out using a simulation platform that included 2 data centres, 2-6 VMs, and 10-100 cloudlets. The length and deadline of each task were randomly generated, with the length staying below a maximum threshold of 1,000,000 Million Instructions (MI) and the deadline below a maximum threshold of 2000 seconds (sec), as listed in Table 3.

TABLE 3. Simulation configuration for experiment type 1.

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

The acceptable workload for each VM was determined based on factors such as processor speed, available memory space, and bandwidth. The parameters used in the simulation and the configuration for the proposed algorithm are illustrated in Table 4.

TABLE 4. Example of cloudlets.

Cloudlet ID	Length	Deadline	File Size	Output Size	CPU
0	243168	129	300	300	1
1	620082	85	300	300	1
2	504033	471	300	300	1
3	782555	1713	300	300	1
4	426539	1552	300	300	1
5	597913	1037	300	300	1
6	599203	1937	300	300	1
7	387146	1029	300	300	1
8	425832	1027	300	300	1
9	100034	571	300	300	1

The algorithm was tested with preemptive task scheduling, allowing tasks to be interrupted and relocated to another resource if SLA violations occur, as depicted in Table 5.

The scheduling process takes into account various QoS performance parameters of cloudlets.

1) The task length parameter in CloudSim determines the size of tasks in bytes, with smaller tasks leading to higher resource utilisation. Each cloudlet is assigned a length value to specify its type, whether heavy, light, or medium request. To ensure client requests are distinguished, the length of each cloudlet is randomly assigned for this experiment. This random assignment of length reflects the overall workload of the cloud environment and is crucial in determining the load for each VM. Additionally, this parameter impacts the time to complete requests in each VM, aiding in the identification of any SLA violations.
2) The task deadline is the maximum time allotted for task execution and is a critical consideration for CSPs within the SLA. In this experiment, each cloudlet has a unique deadline value, enabling SLA contracts to be customised to individual client needs and service expectations from cloud providers. It is advisable to use random deadline values instead of static ones. The deadline parameter holds significant importance as it embodies the SLA; exceeding the deadline with time to complete requests signifies an SLA violation.

In this experiment, the performance evaluation of the proposed EDLB algorithm was conducted across three distinct test cases:

- Scenario 1: 2 VMs and a range of 10 to 100 cloudlets.
- Scenario 2: 4 VMs and a range of 10 to 100 cloudlets.
- Scenario 3: 6 VMs and a range of 10 to 100 cloudlets.

This approach, varying the number of VMs and the range of cloudlets, was designed to observe the effectiveness of the proposed algorithm in different simulation environments. By incrementally adjusting these parameters, we aimed to thoroughly assess the scheduling process and workload distribution among VMs, gaining comprehensive insights into the performance of the algorithm across diverse scenarios.

Tables 6, 7, and 8 illustrate the outcomes for the three scenarios with cloudlets ranging from 10 to 100. These results show a consistent increase in balancing percentage, along

TABLE 5. Example of output (snapped from CloudSim).

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 6. Results obtained with 2 VMs.

Cloudlets	Total Ms	Avg Ms	Avg ExTime	Util	Balancing
10	211.92	195.42	138.14	69.73	92.44
15	315.59	296.96	205.16	68.38	94.22
20	409.7	391.13	266.13	67.48	95.55
25	513.14	492.44	331.77	66.94	96.02
30	610.45	588.58	393.96	66.53	96.45
35	713.27	690.19	461.44	66.57	96.81
40	809.68	783.55	518.86	65.93	96.8
45	911.48	885.79	588.44	66.19	97.21
50	1011.93	984.07	651.54	65.99	97.27
55	1109.97	1080.91	714.21	65.85	97.4
60	1215.17	1184.15	782.81	65.91	97.46
65	1306.79	1275.81	840.25	65.67	97.64
70	1410.52	1376.74	908.85	65.84	97.61
75	1512.97	1477.99	975.35	65.84	97.7
80	1611.49	1574.87	1038.13	65.77	97.74
85	1713.41	1675.66	1105.2	65.81	97.81
90	1809.99	1768.37	1163.0	65.63	97.71
95	1912.86	1872.17	1233.67	65.76	97.88
100	2011.94	1968.18	1295.19	65.69	97.83

Total Ms = Total Makespan; Avg Ms= Average Makespan;
Avg ExTime = Average Execution Time; Util = Utilisation

TABLE 7. Results obtained with 4 VMs.

Cloudlets	Total Ms	Avg Ms	Avg ExTime	Util	Balancing
10	261.36	202.3	155.62	76.23	78.63
15	361.41	299.17	216.73	71.85	83.87
20	472.0	397.85	278.91	69.68	85.99
25	574.33	496.98	343.39	68.73	88.12
30	677.65	596.33	406.61	67.86	89.32
35	780.92	692.79	468.81	67.41	89.93
40	887.63	789.61	530.59	66.94	90.52
45	981.5	888.19	594.37	66.69	91.44
50	1088.38	992.17	663.84	66.71	92.13
55	1209.13	1091.55	726.38	66.33	91.76
60	1303.69	1186.17	785.22	66.04	92.26
65	1399.03	1284.39	849.13	65.96	92.84
70	1507.98	1385.96	916.85	66.01	93.04
75	1610.8	1477.75	973.49	65.75	92.96
80	1718.39	1578.82	1038.87	65.67	93.33
85	1811.14	1679.11	1104.82	65.68	93.73
90	1893.7	1769.21	1161.98	65.56	94.16
95	2027.48	1870.89	1224.12	65.32	93.55
100	2141.18	1976.74	1297.39	65.54	93.76

Total Ms = Total Makespan; Avg Ms= Average Makespan;
Avg ExTime = Average Execution Time; Util = Utilisation

with favourable trends in total makespan, average makespan, and execution time. The high utilisation percentages further demonstrate the efficiency and reliability of the algorithm in handling various VM configurations and cloudlet thresholds.

Next, we choose 6 VMs to demonstrate the improvement distribution of our algorithm against the benchmark algorithm. Figures 3, 4, and 5 collectively demonstrate a marked enhancement of our algorithm, indicating a significant performance improvement for the matter of tasks submission and processing that represented by makespan and the execution time metrics. Notably, our algorithm exhibits superior efficiency in reducing the total makespan and average makespan, as well as in execution time.

Furthermore, our algorithm effectively maintains system scalability by sustaining high resource utilisation as depicted in Figure 6. The resource utilisation begins at 84% and remains consistently high, even as the number of cloudlets increases. This demonstrates the capability of the proposed algorithm to handle larger workloads while efficiently utilising available resources. Figure 7 shows a comparison of the balancing percentage, where EDLB algorithm starts at approximately 68% and increases to 90%. In contrast, the benchmark algorithm only grows up to 70%. This demonstrates the superior balancing capability of our algorithm over time.

Additionally, to further enhance the results, a more extensive dataset consisting of 1,000 cloudlets was employed. The

TABLE 8. Results obtained with 6 VMs.

Cloudlets	Total Ms	Avg Ms	Avg ExTime	Util	Balancing
10	300.34	202.39	167.04	81.99	68.37
15	416.62	303.61	231.94	75.97	74.05
20	518.38	398.66	288.81	72.11	77.92
25	620.78	498.91	354.98	70.78	81.46
30	731.2	598.4	416.68	69.37	82.97
35	832.61	694.64	477.34	68.48	84.67
40	957.07	796.84	542.73	67.91	84.93
45	1045.68	890.05	602.78	67.55	86.25
50	1165.48	990.48	663.76	66.86	86.46
55	1267.06	1084.9	725.01	66.66	87.03
60	1361.77	1184.3	791.24	66.68	87.94
65	1476.95	1285.94	855.21	66.37	88.57
70	1565.44	1377.87	914.24	66.21	88.96
75	1706.3	1478.06	977.53	66.04	88.23
80	1788.35	1571.19	1037.27	65.92	89.2
85	1878.32	1662.26	1093.72	65.69	89.64
90	1980.42	1758.03	1156.27	65.67	89.83
95	2087.82	1856.14	1216.57	65.47	89.93
100	2180.69	1952.44	1281.77	65.57	90.34

Total Ms = Total Makespan; Avg Ms= Average Makespan;
Avg ExTime = Average Execution Time; Util = Utilisation

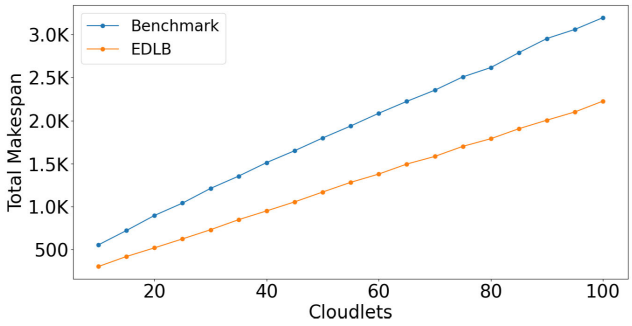

FIGURE 3. Total makespan comparison for 6 VMs.

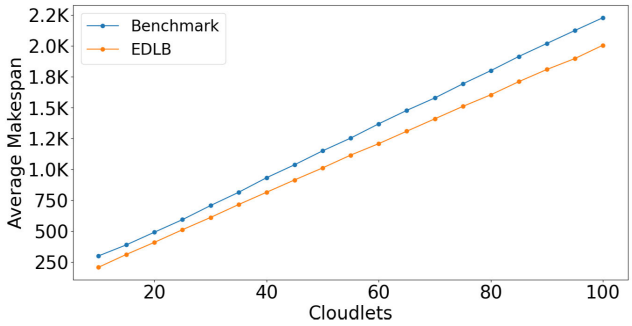

FIGURE 4. Average makespan comparison for 6 VMs.

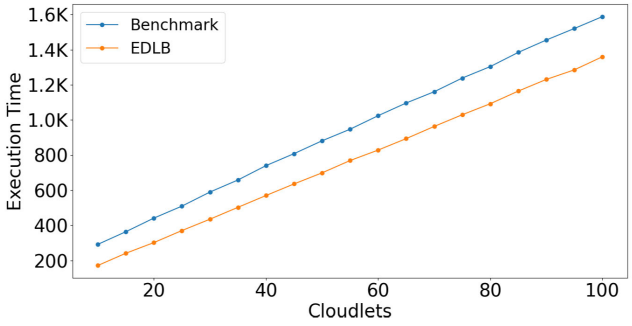

FIGURE 5. Execution time comparison for 6 VMs.

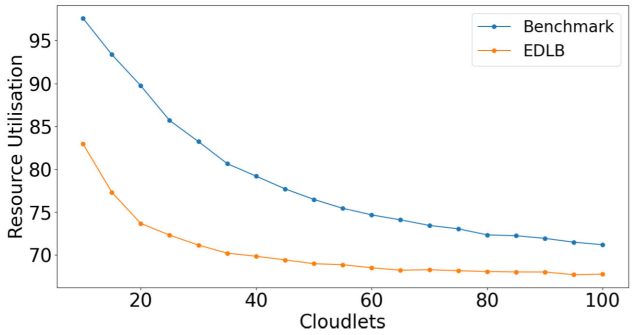

FIGURE 6. Resource utilisation comparison for 6 VMs.

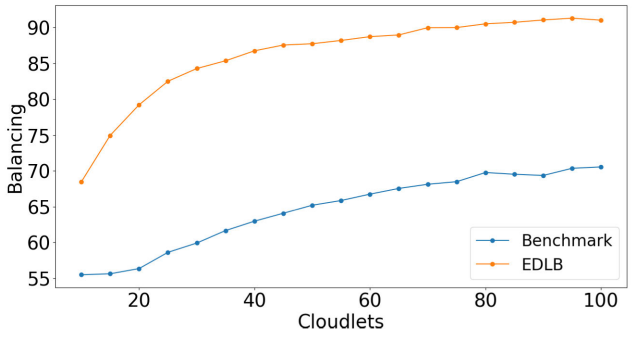

FIGURE 7. Balancing percentage comparison for 6 VMs.

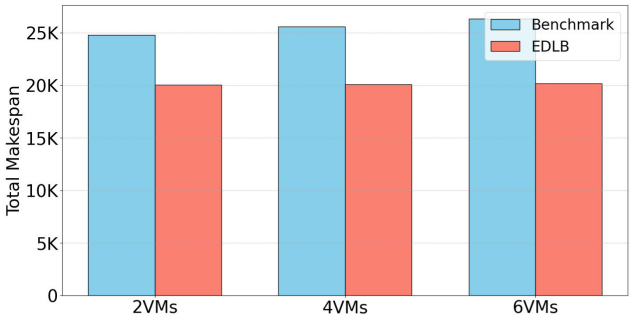

FIGURE 8. Total makespan comparison with 1000 cloudlets.

algorithm continued to demonstrate superior performance, as illustrated in Figures 8, 9, 10, 11, 12, respectively.

The aforementioned figures are highlighting the effectiveness of our algorithm. Figures 8, 9, and 10, prove that our algorithm achieves lower total makespan, average makespan, and execution time compared to the benchmark, indicating more efficient task scheduling and execution.

Figure 11 shows that both our algorithm and the benchmark maintain a resource utilisation rate of around 65% which remains consistently high. Figure 12 demonstrates the superior balancing percentage of our algorithm at 98 to 99% compared to the benchmark at 81 to 85% showcasing better cloudlet distribution among VMs. These results confirm that

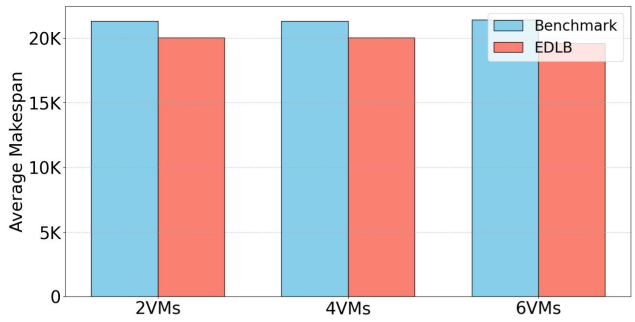

FIGURE 9. Average makespan comparison with 1000 cloudlets.

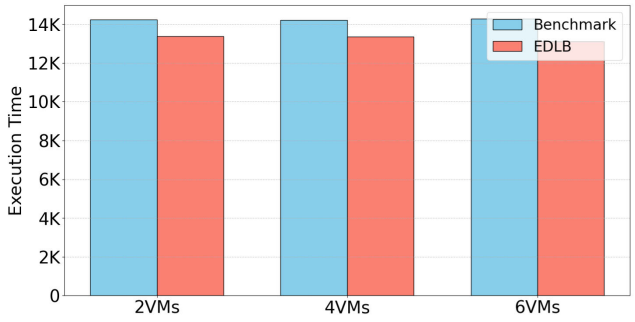

FIGURE 10. Execution time comparison with 1000 cloudlets.

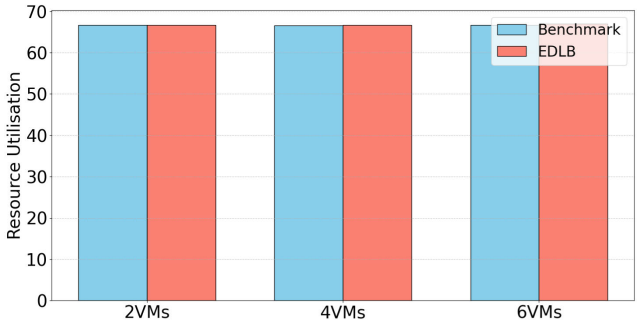

FIGURE 11. Resource utilisation comparison with 1000 cloudlets.

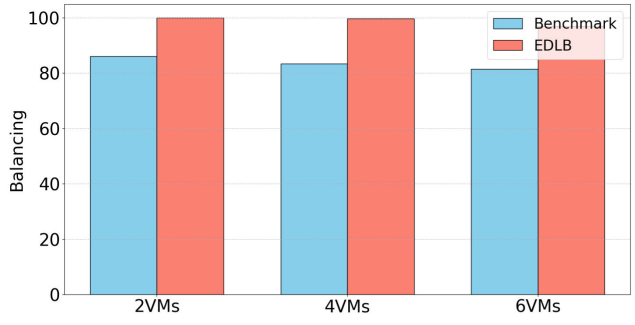

FIGURE 12. Balancing percentage comparison with 1000 cloudlets.

our algorithm effectively handles a large number of cloudlets while optimising performance and resource use.

The EDLB algorithm consistently exhibits a stable trend across these figures. This indicates that the performance of the proposed algorithm remains reliable even as the number of cloudlets and VMs varies. While the benchmark algorithm maintain the SLA violations and adjusting resource configurations, it tends to fall short in load balancing, potentially leading to sub-optimal resource utilisation. On the other hand, EDLB algorithm prioritises load balancing and makespan reduction while still considering SLA and QoS parameters.

The benchmark algorithm utilises a random allocation strategy for cloudlets. After allocation, it detects SLA violations, triggering migrations and subsequently adjusting CPU configurations if necessary. However, this random allocation approach often leads to imbalanced workloads across VMs, negatively affecting overall system efficiency. In contrast, the EDLB algorithm adopts a more sophisticated approach. By considering the completion times of individual cloudlets, it strategically allocates them to VMs. When SLA violations occur, the algorithm relocates cloudlets and reconfigures CPU resources if insufficient capacity is detected. This approach aims to achieve load balancing and reduce makespan by intelligently distributing tasks based on their characteristics. Notably, the performance results from both algorithms highlight distinct characteristics. The benchmark algorithm exhibits variability in performance as the number of cloudlets increases, potentially affecting SLA adherence and resource utilisation. Meanwhile, the EDLB algorithm demonstrates consistent load balancing and makespan reduction, showcasing robust performance under varying conditions.

2) TYPE 2 EXPERIMENT (EDLB ALGORITHM VS. STATE-OF-THE-ART ALGORITHM)

In the second experiment, we assess the performance of our EDLB algorithm against a state-of-the-art algorithm, the credit-based resource-aware load balancing scheduling algorithm (HO-CB-RALB-SA) [30]. The HO-CB-RALB-SA algorithm employs a hybrid approach, combining the Walrus Optimisation Algorithm (WOA) and the Lyrebird Optimisation Algorithm (LOA) to enhance scheduling efficiency and fairness in cloud computing environments.

The simulation configuration for this experiment is presented in Table 9.

TABLE 9. Simulation configuration for experiment type 2.

Type	Parameters	Value
Cloudlet	Number of cloudlets	400-2000
	Length of tasks (bytes)	1000-10000
	Number of Processor Elements	1
VM	Number of VMs	100
	Number of Processor Elements	1
	MIPS per VM	5000
	VMM	Xen
	Cloudlet Scheduler	Time Shared
Data centre	Number of Data centres	15
	Number of Hosts in Data centre	2

In this experiment, we aimed to demonstrate that our proposed EDLB algorithm outperforms the-state-of-the-art

algorithm (i.e., HO-CB-RALB-SA algorithm). The HO-CB-RALB-SA algorithm is designed to enhance scheduling efficiency and fairness by addressing common challenges in traditional scheduling algorithms, such as load imbalance and suboptimal resource utilisation. It uses a hybrid approach that incorporates credit-based job prioritisation and resource-aware load balancing. By evaluating tasks based on parameters such as duration, cost, deadline, and priority, and by assigning additional credits to effectively prioritise jobs, the algorithm ensures balanced task distribution across VMs. This approach not only improves task scheduling but also optimises resource usage, leading to better overall performance and efficiency.

The same metrics of the first experiment has been selected which are the five key metrics—total makespan, average makespan, execution time, resource utilisation, and balancing percentage—to evaluate the performance of the algorithms. These metrics are crucial for assessing the efficiency and effectiveness of cloud computing algorithms, as they offer valuable insights into overall performance and the ability to manage and process tasks, which is essential for determining practical utility.

Total makespan: measures the efficiency with which an algorithm handles the complete set of tasks from start to finish. This metric directly reflects the capability of the algorithm to minimise processing time, a critical factor in cloud environments where time efficiency translates into cost savings and improved user satisfaction. As shown in Figure 13, the EDLB algorithm consistently achieves a lower total makespan compared to HO-CB-RALB-SA across all cloudlet sizes. For instance, with 1600 cloudlets, EDLB records a total makespan of 93.76 sec., while HO-CB-RALB-SA logs 461.87 sec., highlighting a significant improvement in task completion time. The overall improvement of the total makespan achieved 62.90% compared with HO-CB-RALB-SA algorithm.

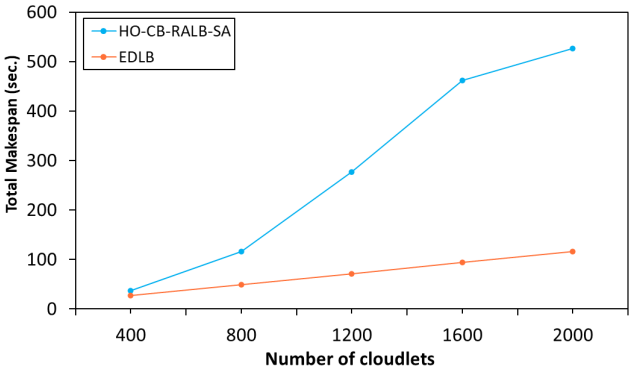

FIGURE 13. Total makespan.

Average makespan: provides insight into time efficiency per task, which is especially important in scenarios with varying task complexities. By focusing on this metric, we can evaluate how well the algorithm handles individual tasks, ensuring high performance under diverse workloads. Figure 14 illustrates that the EDLB algorithm consistently outperforms HO-CB-RALB-SA with an average makespan of 65.85 sec. for 1200 cloudlets compared to HO-CB-RALB-SA 90.32 sec. This result underscores the EDLB algorithm effectiveness in optimising task execution times. The overall improvement in the average makespan compared to the HO-CB-RALB-SA algorithm is 10%.

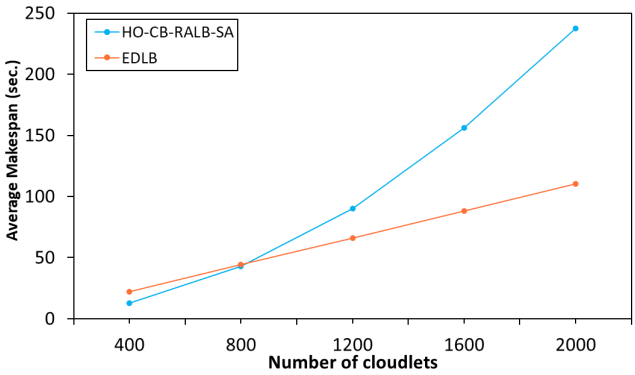

FIGURE 14. Average makespan.

Execution time is critical for assessing the efficiency of algorithm in utilising processing resources. Reducing execution time not only decreases operational costs but also increases throughput, which is paramount in large-scale cloud environments. As shown in Figure 15, EDLB demonstrates more efficient use of processing resources. For instance, with 2,000 cloudlets, EDLB records an execution time of 81.29 sec., while HO-CB-RALB-SA requires 172.32 sec., further demonstrating the superiority of EDLB in terms of execution time reduction. The overall average improvement in execution time was 15.6%.

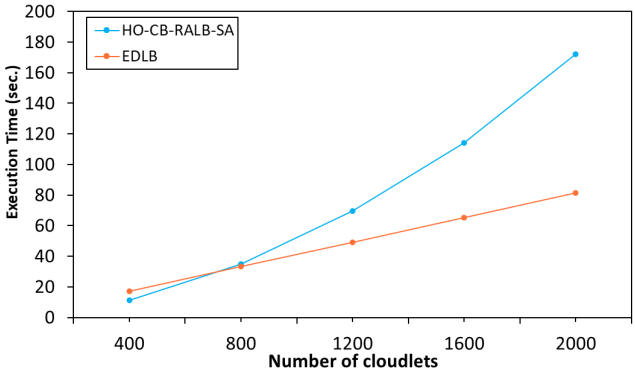

FIGURE 15. Execution time.

Resource utilisation reflects how effectively the algorithm leverages available resources, with a higher percentage indicating more efficient usage. As shown in Figure 16, EDLB consistently maintains high resource utilisation. For instance, in scenarios with 800 cloudlets EDLB achieves a resource

TABLE 10. Comparison of HO-CB-RALB-SA (HO-CB.) and EDLB algorithms across various metrics.

Cloudlets	Total makespan		Average makespan		Execution time		Resource utilisation		Balancing percentage	
	HO-CB.	EDLB	HO-CB.	EDLB	HO-CB.	EDLB	HO-CB.	EDLB	HO-CB.	EDLB
400	37.39	27.7704	12.6625	21.9991988	11.36	17.25287628	89.71	78.41	33.87	79.23
800	115.57	49.73136	43.0221	44.041538	34.88	33.22872084	80.61	75.44	37.27	88.57
1200	277.094	71.67702	90.3242	65.8495791	69.54	49.00320871	76.99	74.41	32.57	91.87
1600	461.87	93.76393	156.0465	87.979621	114.13	65.11754391	73.14	74.01	33.79	93.83
2000	526.02	115.96867	237.7045	110.148627	172.32	81.28508741	72.49	73.79	45.19	94.98

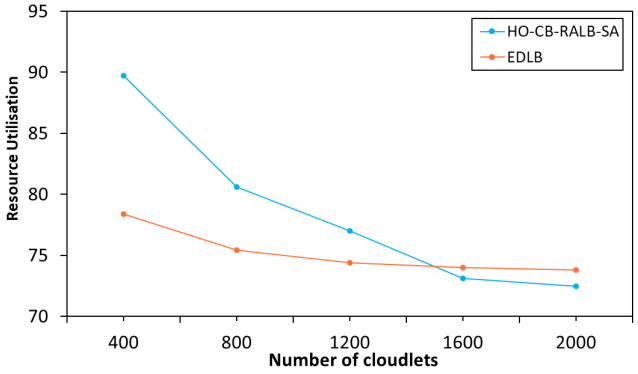

FIGURE 16. Resource utilisation.

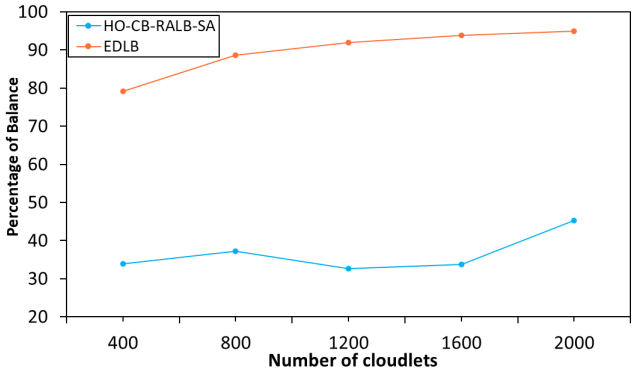

FIGURE 17. Balancing percentage.

utilisation rate of 75.44% compared to HO-CB-RALB-SA 80.61%. This trend persists across various cloudlet sizes confirming the robustness of EDLB in efficiently utilising resources.

Even as the number of cloudlets increases, EDLB maintains steady resource utilisation, while the HO-CB-RALB-SA algorithm shows a decline. The overall improvement of our algorithm, across cloudlet sizes ranging from 400 to 2,000, was 3.8%.

Balancing percentage: measures the effectiveness of workload distribution across available resources, with a higher score indicating a more even distribution. Figure 17 reveals that the EDLB algorithm demonstrates superior balancing capabilities, particularly in scenarios with higher cloudlet counts. For instance, with 2,000 cloudlets, EDLB achieves a balancing efficiency of 95%, while HO-CB-RALB-SA manages only 45.2%. This superior balancing performance is consistent across all tested scenarios. The overall improvement of our algorithm, across cloudlet sizes ranging from 400 to 2,000, was 59.2%.

Overall, the EDLB algorithm consistently outperforms HO-CB-RALB-SA across all evaluated metrics, making it a more reliable and efficient choice for managing cloud computing tasks. The results clearly demonstrate the practical benefits of EDLB, affirming its superiority over HO-CB-RALB-SA. Table 10 illustrates the superiority of our algorithm against the state-of-the-art algorithm.

Furthermore, EDLB algorithm employs a sophisticated task allocation strategy by considering the completion times of individual cloudlets before assigning them to VMs. This ensures an optimal distribution of workloads, preventing imbalances that can negatively impact system efficiency. In the event of SLA violations, EDLB dynamically relocates cloudlets and adjusts CPU configurations when resource insufficiencies are detected. By intelligently managing task distribution based on cloudlet characteristics, EDLB consistently achieves load balancing, reduces makespan, and maintains high resource utilisation, even as the number of cloudlets increases. This methodology distinguishes EDLB from other algorithms that rely on random allocation, making it more robust and efficient under varying conditions.

V. CONCLUSION AND FUTURE WORK

This research highlights the critical role of cloud computing, particularly within the IaaS model, and emphasises the importance of employing advanced load balancing algorithms to optimise resource allocation. The EDLB algorithm, evaluated using the CloudSim toolkit, focuses on preemptive task scheduling and cloudlet parameters such as arrival time, task length, and deadlines. By doing so, it consistently improves performance metrics and addresses SLA concerns. The results confirm the reliability of the algorithm across various scenarios, demonstrating its robust performance under different quantities of cloudlets and VMs. Compared to the benchmark algorithm, which employs random cloudlet allocation, and the HO-CB-RALB-SA algorithm, EDLB shows superior stability and load balancing capabilities. The benchmark algorithm often leads to imbalanced workloads and inefficiencies, while HO-CB-RALB-SA exhibits higher execution times. In contrast, EDLB effectively balances workloads, optimises makespan, execution time, and resource utilisation, enhancing the overall efficiency of cloud-based services.

Future research should incorporate reliability and fairness as key metrics to further strengthen the robustness of the algorithm. Additionally, expanding the evaluation to include a broader range of QoS parameters—such as response time, throughput, and fault tolerance—would provide deeper insights into its effectiveness. Furthermore, it is essential to consider other resources, such as RAM, in the allocation process, as this could impact the overall performance and efficiency of cloud operations. Exploring the scalability of EDLB in larger, more diverse cloud environments and refining the inter-VM resource transfer mechanism, potentially with machine learning for dynamic resource prediction, could further optimise allocation. This could also extend the applicability of the algorithm to emerging areas like green computing.

ACKNOWLEDGMENT

Raiymbek Zhanuzak, Mohammed Alaa Ala'anzy, and Mohamed Othman would like to extend their gratitude to the Science Department, SDU University, and Dr. Abdulmohsen Algarni for their invaluable guidance and support throughout this research.

REFERENCES

[1] Statista. (Jul. 2024). *Infrastructure as a Service: Market Data & Analysis*. Accessed: Sep. 8, 2024. [Online]. Available: https://www.statista.com/study/84972/infrastructure-as-a-service-report/

[2] F. Xia, Y. Chen, and J. Huang, "Privacy-preserving task offloading in mobile edge computing: A deep reinforcement learning approach," *Softw., Pract. Exper.*, vol. 54, no. 9, pp. 1774–1792, Sep. 2024.

[3] T. Kumar, P. Sharma, J. Tanwar, H. Alsghier, S. Bhushan, H. Alhumyani, V. Sharma, and A. I. Alutaibi, "Cloud-based video streaming services: Trends, challenges, and opportunities," *CAAI Trans. Intell. Technol.*, vol. 9, no. 2, pp. 265–285, Apr. 2024.

[4] S. M. Rozehkhani, F. Mahan, and W. Pedrycz, "Efficient cloud data center: An adaptive framework for dynamic virtual machine consolidation," *J. Netw. Comput. Appl.*, vol. 226, Jun. 2024, Art. no. 103885.

[5] J. Sung, S.-J. Han, and J.-W. Kim, "Cloning-based virtual machine pre-provisioning for resource-constrained edge cloud server," *Cluster Comput.*, vol. 27, no. 2, pp. 1831–1847, Apr. 2024.

[6] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 2, pp. 149–158, Feb. 2020.

[7] A. Rajagopalan, D. Swaminathan, M. Bajaj, I. Damaj, R. S. Rathore, A. R. Singh, V. Blazek, and L. Prokop, "Empowering power distribution: Unleashing the synergy of IoT and cloud computing for sustainable and efficient energy systems," *Results Eng.*, vol. 21, Mar. 2024, Art. no. 101949.

[8] M. Zakarya, A. A. Khan, M. R. C. Qazani, H. Ali, M. Al-Bahri, A. U. R. Khan, A. Ali, and R. Khan, "Sustainable computing across datacenters: A review of enabling models and techniques," *Comput. Sci. Rev.*, vol. 52, May 2024, Art. no. 100620.

[9] F. Qazi, D. Kwak, F. G. Khan, F. Ali, and S. U. Khan, "Service level agreement in cloud computing: Taxonomy, prospects, and challenges," *Internet Things*, vol. 25, Apr. 2024, Art. no. 101126.

[10] M. A. Ala'anzy and M. Othman, "Mapping and consolidation of VMs using locust-inspired algorithms for green cloud computing," *Neural Process. Lett.*, vol. 54, no. 1, pp. 405–421, Feb. 2022.

[11] M. A. Ala'anzy, M. Othman, S. Hasan, S. M. Ghaleb, and R. Latip, "Optimising cloud servers utilisation based on locust-inspired algorithm," in *Proc. 7th Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2020, pp. 23–27.

[12] S. Sha, C. Li, X. Wang, Z. Wang, and Y. Luo, "Hardware-software collaborative tiered-memory management framework for virtualization," *ACM Trans. Comput. Syst.*, vol. 42, nos. 1–2, pp. 1–32, May 2024.

[13] M. Ala'anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: A meta-study," *IEEE Access*, vol. 7, pp. 141868–141887, 2019.

[14] Z. Ahmad, A. I. Jehangiri, M. A. Ala'anzy, M. Othman, and A. I. Umar, "Fault-tolerant and data-intensive resource scheduling and management for scientific applications in cloud computing," *Sensors*, vol. 21, no. 21, p. 7238, Oct. 2021.

[15] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *J. Netw. Comput. Appl.*, vol. 143, pp. 1–33, Oct. 2019.

[16] J. Zhou, U. K. Lilhore, P. M. T. Hai, S. Simaiya, D. N. A. Jawawi, D. Alsekait, S. Ahuja, C. Biamba, and M. Hamdi, "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," *J. Cloud Comput.*, vol. 12, no. 1, p. 85, Jun. 2023.

[17] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.

[18] T. Bu, Z. Huang, K. Zhang, Y. Wang, H. Song, J. Zhou, Z. Ren, and S. Liu, "Task scheduling in the Internet of Things: Challenges, solutions, and future trends," *Cluster Comput.*, vol. 27, no. 1, pp. 1017–1046, Feb. 2024.

[19] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm to minimize the makespan time and utilize the resources effectively in cloud environment," *Int. J. Comput. Appl.*, vol. 42, no. 1, pp. 108–117, Jan. 2020.

[20] G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Procedia Comput. Sci.*, vol. 57, pp. 545–553, Jan. 2015, doi: 10.1016/j.procs.2015.07.385. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915019146

[21] B. H. Shanthan and L. Arockiam, "Resource based load balanced min min algorithm (RBLMM) for static meta task scheduling in cloud," in *Proc. Int. Conf. Adv. Comput. Sci. Technol.*, 2018, pp. 1–8.

[22] M. Adhikari and T. Amgoth, "An enhanced dynamic load balancing mechanism for task deployment in IaaS cloud," in *Proc. Int. Conf. Comput., Power Commun. Technol. (GUCON)*, Sep. 2018, pp. 451–456.

[23] S. Acharya and D. A. D'Mello, "Enhanced dynamic load balancing algorithm for resource provisioning in cloud," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, vol. 2, Aug. 2016, pp. 1–5.

[24] S. Banerjee, M. Adhikari, S. Kar, and U. Biswas, "Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud," *Arabian J. Sci. Eng.*, vol. 40, no. 5, pp. 1409–1425, May 2015.

[25] H. Gamal El Din Hassan Ali, I. A. Saroit, and A. M. Kotb, "Grouped tasks scheduling algorithm based on QoS in cloud computing network," *Egyptian Informat. J.*, vol. 18, no. 1, pp. 11–19, Mar. 2017.

[26] M. A. Alworafi, A. Dhari, A. A. Al-Hashmi, A. B. Darem, and Suresha, "An improved SJF scheduling algorithm in cloud computing environment," in *Proc. Int. Conf. Electr., Electron., Commun., Comput. Optim. Techn. (ICEECCOT)*, Dec. 2016, pp. 208–212.

[27] V. Polepally and K. Shahu Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing," *Cluster Comput.*, vol. 22, no. S1, pp. 1099–1111, Jan. 2019.

[28] P. Neelakantan and N. S. Yadav, "An optimized load balancing strategy for an enhancement of cloud computing environment," *Wireless Pers. Commun.*, vol. 131, no. 3, pp. 1745–1765, Aug. 2023.

[29] S. Simaiya, U. K. Lilhore, Y. K. Sharma, K. B. V. B. Rao, V. V. R. M. Rao, A. Baliyan, A. Bijalwan, and R. Alroobaea, "A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques," *Sci. Rep.*, vol. 14, no. 1, p. 1337, Jan. 2024.

[30] A. Narwal, "Resource utilization based on hybrid WOA-LOA optimization with credit based resource aware load balancing and scheduling algorithm for cloud computing," *J. Grid Comput.*, vol. 22, no. 3, pp. 1–24, Sep. 2024.

[31] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021.

[32] M. Alanzy, R. Latip, and A. Muhammed, "Range wise busy checking 2-way imbalanced algorithm for cloudlet allocation in cloud environment," *J. Phys., Conf. Ser.*, vol. 1018, May 2018, Art. no. 012018.

[33] M. A. Ala'anzy, M. Othman, Z. M. Hanapi, and M. A. Alrshah, "Locust inspired algorithm for cloudlet scheduling in cloud computing environments," *Sensors*, vol. 21, no. 21, p. 7308, Nov. 2021.

[34] J. Konjaang, F. H. Ayob, and A. Muhammed, "An optimized max-min scheduling algorithm in cloud computing," *J. Theor. Appl. Inf. Technol.*, vol. 95, no. 9, pp. 1–10, 2017.

[35] A. Dhari and K. I. Arif, "An efficient load balancing scheme for cloud computing," *Indian J. Sci. Technol.*, vol. 10, no. 11, pp. 1–8, Mar. 2017.

[36] Y. Vijay and B. V. Ghita, "Evaluating cloud computing scheduling algorithms under different environment and scenarios," in *Proc. 8th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2017, pp. 1–5.
[37] 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, Jul. 2022.

RAIYMBEK ZHANUZAK is currently pursuing the degree with the Faculty of Natural Sciences and Engineering, SDU University.

With a strong interest in algorithm design and optimization, he has worked on projects to improve existing algorithms and solve complex computational problems. He has a good understanding of algorithmic paradigms, such as dynamic programming, greedy algorithms, and graph theory. He collaborates well with peers and mentors and actively engages in research opportunities and extracurricular activities related to computer science. He participates in coding competitions, workshops, and conferences to enhance his skills and knowledge. His research interests include data science, distributed systems, and cloud computing.

MOHAMMED ALAA ALA'ANZY (Member, IEEE) received the Ph.D. degree in computer science from Universiti Putra Malaysia (UPM), in 2023.

He is currently an Assistant Professor with Suleyman Demirel University (SDU). He specializes in advanced fields, such as algorithms, cloud computing, green computing, load balancing, task scheduling, fog computing, and the Internet of Things (IoT). He is widely recognized for his significant contributions to the academic community through high-impact journals and conference publications. Additionally, he serves as a respected reviewer for prestigious journals, such as IEEE, Elsevier, and Springer. He is also a member of the Dissertation Council, SDU University, where he contributes to the evaluation of doctoral theses. In addition, he plays an essential role on the admissions committee responsible for evaluating candidates for the Kazakhstan government grant program. He is also a member of the International Program Committee for the 2024 11th International Conference on Soft Computing and Machine Intelligence (ISCMI), Melbourne, Australia, and Technical Program Committee for 2024 Seventh International Symposium on Telecommunication Technologies in Langkawi, Malaysia.

MOHAMED OTHMAN (Senior Member, IEEE) received the Ph.D. degree (Hons.) from the National University of Malaysia. He is currently a Professor in computer science with the Department of Communication Technology and Network, Universiti Putra Malaysia (UPM). Prior to that, he was the Deputy Director of the Information Development and Communication Centre, where he was in charge of UMPNet Network Campus, uSport Wireless Communication Project, and the UPM DataCentre. He is also an Associate Researcher and a Coordinator in high speed machine with the Laboratory of Computational Science and Mathematical Physics, Institute of Mathematical Research (INSPEM), UPM. In 2017, he received an Honorable Professor from South Kazakhstan Pedagogical University, Shymkent, Kazakhstan, where he was a Visiting Professor, and L. N. Gumilyov Eurasian National University, Astana, Kazakhstan. He has published more than 300 international journals and 330 proceeding articles. He has also filed six Malaysian, one Japanese, one South Korean, and three U.S. patents. His research interests include computer networks, parallel and distributed computing, high speed interconnection networks, network design and management (network security, wireless, and traffic monitoring), consensus in IoT, and mathematical model in scientific computing. He is a Life Member of Malaysian National Computer Confederation and Malaysian Mathematical Society. He was a recipient of the Best Ph.D. Thesis, in 2000, by Sime Darby Malaysia and Malaysian Mathematical Science Society.

ABDULMOHSEN ALGARNI received the Ph.D. degree from Queensland University of Technology, Australia, in 2012. He was a Research Associate with the School of Electrical Engineering and Computer Science, Queensland University of Technology, in 2012. He is currently an Associate Professor with the College of Computer Science, King Khalid University. His research interests include artificial intelligence, data mining, text mining, machine learning, information retrieval, and information filtering.