Received March 7, 2021, accepted March 8, 2021, date of publication March 10, 2021, date of current version March 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065308

A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications

DALIA ABDULKAREEM SHAFIQ[1], NOOR ZAMAN JHANJHI[1], AZWEEN ABDULLAH[1], (Senior Member, IEEE), AND MOHAMMED A. ALZAIN[2]

[1]School of Computer Science and Engineering (SCE), Taylor's University, Subang Jaya 47500, Malaysia
[2]Department of Information Technology, College of Computers and Information Technology, Taif University, Taif 21944, Saudi Arabia

Corresponding author: Noor Zaman Jhanjhi (noorzaman.jhanjhi@taylors.edu.my)

This work was supported by the Taif University Researchers Supporting Project, Taif University, Taif, Saudi Arabia, under Grant TURSP-2020/98.

ABSTRACT Despite the many past research conducted in the Cloud Computing field, some challenges still exist related to workload balancing in cloud-based applications and specifically in the Infrastructure as service (IaaS) cloud model. Efficient allocation of tasks is a crucial process in cloud computing due to the restricted number of resources/virtual machines. IaaS is one of the models of this technology that handles the backend where servers, data centers, and virtual machines are managed. Cloud Service Providers should ensure high service delivery performance in such models, avoiding situations such as hosts being overloaded or underloaded as this will result in higher execution time or machine failure, etc. Task Scheduling highly contributes to load balancing, and scheduling tasks much adheres to the requirements of the Service Level Agreement (SLA), a document offered by cloud developers to users. Important SLA parameters such as Deadline are addressed in the LB algorithm. The proposed algorithm is aimed to optimize resources and improve Load Balancing in view of the Quality of Service (QoS) task parameters, the priority of VMs, and resource allocation. The proposed LB algorithm addresses the stated issues and the current research gap based on the literature's findings. Results showed that the proposed LB algorithm results in an average of 78% resource utilization compared to the existing Dynamic LBA algorithm. It also achieves good performance in terms of less Execution time and Makespan.

INDEX TERMS Cloud computing, load balancing, makespan, optimization, QoS, SLA, task scheduling.

I. INTRODUCTION

As we shift more towards online storage and services, Cloud Computing technology becomes an essential part of the business. This technology provides services through various kinds such as in software via web browsers, in Platforms such as designing and developing cloud-based applications. In the Infrastructure, the backend is managed by Cloud Service Providers (CSPs) such as maintaining Data Centres, servers, etc. Although there exist many other service delivery models in this technology, however, in this research, the focus is on the Infrastructure as a Service (IaaS) model. It deals with the server-side of this technology for resource allocation [1].

Virtualization is the backbone and essential feature [2] of cloud-based applications. This technique can significantly affect the performance of the scalable and on-demand services provided to clients if the migration process and allocation of virtual machine resources are handled inefficiently. According to [3], cloud performance is proved to be in the top three Cloud Computing challenges. This research aims to enhance resource allocation in the IaaS model; this concept is fundamental [4] as it deals with the balancing of resources provided to clients and the workload/user requests on servers. The cloud users access services by sending requests; these are represented in Virtual Machines (VMs) [5] in the cloud environment. CSPs should deliver services that are beneficial to businesses and increase user satisfaction [6]. Thus, the proposed Load Balancing algorithm is developed mainly focusing on the IaaS model out of the three service models in the cloud where authors deal with the Cloud Computing technology's backend, such as server workload. There are two components in a typical cloud environment: the frontend is the user side, and it is accessible by connecting to the Internet [7]. The backend side handles the cloud service

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

models where the Data Center store multiple physical machines (known as servers). Incoming user requests are received from the application are dynamically scheduled, and through virtualization, the necessary resources are allocated to clients. The virtualization technique is also responsible for balancing the load in the entire system, scheduling [8], and efficient allocation of resources.

CSPs and cloud users can leverage the advantage of virtualization as well as dynamic task scheduling techniques. Thus, efficient scheduling can highly reduce execution time and increase the ratio of resource utilization in cloud-based applications.

Task Scheduling is a process that highly relates to workload balancing. As illustrated in figure 1 above, as users send requests, the task is submitted through a cloud broker; this is where researchers should focus on providing an efficient algorithm. The proposed algorithm should efficiently submit jobs to appropriate VMs following essential parameters such as deadline [10] to maintain a high quality of services and ensuring the requests sent by users are executed and completed within these specific requirements provided in the Service Level Agreement (SLA) document. The user sends requests via the Internet. These requests are stored in Virtual Machines (VMs), and CSP in every delivery model must maintain the QoS by ensuring the users' requests can be executed and completed within a specific deadline. This process depends highly on the scheduling policy's efficiency (Data Broker) which should be programmed to result in a high technique for balancing workload among the machines and servers. Efficient scheduling and utilization of resources can be achieved by designing and developing a dynamic

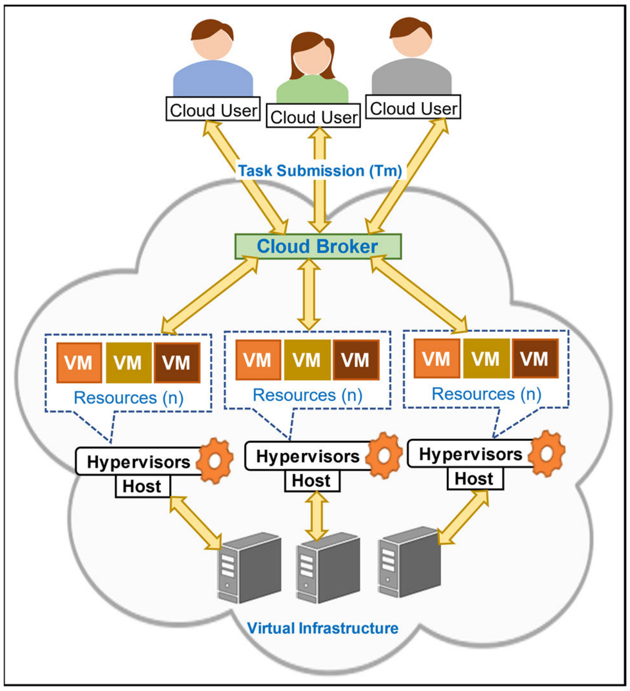


FIGURE 1. Task scheduling in IaaS cloud computing [9].

load balancer (LB). Cloud Computing highly depends on Virtual Machine Monitor (VMM) or hypervisor, as shown in the figure above. Hypervisor is a process that helps execute multiple Virtual Machines on a single layer which is the hardware [11]. VMware is an excellent example of types of hypervisors that reside in the host. Since virtualization plays an essential role in cloud technology, issues such as inappropriate scheduling techniques or efficient mapping of tasks [1] to correct Virtual Machines/resources can quickly degrade cloud-based applications' performance. This, in turn, can lead to an imbalanced workload on servers.

Therefore, there is still room in cloud computing technology to improve mapping resources to tasks with the objective of scheduling. Important QoS parameters should be considered to achieve efficient resource utilization without affecting the SLA and consider constraints such as Deadline, priority, etc. [12]. Resource allocation is one of the challenges in cloud technology, and it contributes to the process of load balancing. This challenge also exists in wireless communication systems [13] where priority among users should be applied and resources must be distributed equally and fairly.

A. RESEARCH CONTRIBUTION

This subsection highlights the contribution made by the authors in this paper.

The research mainly aims to optimize the cloud resources by enhancing the Load Balancing process through efficient Task Scheduling procedures. Our contribution to the study can be summarized as follows:

- A survey of existing Load Balancing and Task Scheduling algorithms.
- A proposed Load Balancing algorithm addresses the VM violation issue in the cloud and provides high-quality service in terms of workload scheduling and balancing. Although researchers have addressed this issue in the past, most do not consider important QoS parameters such as Deadline and Completion Time.
- Additionally, the proposed algorithm includes the migration of load to balance VMs, which is still not fully addressed yet.
- Algorithm results in reducing two main Load Balancing parameters: Makespan and Execution time in the cloud applications and improvement on Resource utilization.

Further, this paper can benefit the forthcoming researchers studying the Cloud Computing field to improve cloud-based applications' performance in terms of Load Balancing and resource allocation.

The rest of the paper is organized as follows. Section II provides the problem statement that is addressed by the proposed algorithm. Section III covers the related work whereby the Load Balancing and Tasks Scheduling concept is explained along with recent research presented by other authors highlighting their strengths, weaknesses, and future work. Section IV provides details regarding the proposed LB algorithm covering the proposed framework, the flowchart, and the pseudocode. Section V includes the details of implementing the algorithm, such as simulation setup and performance metrics. Section VI provides the discussion and results obtained from the experiment. In section VII, our research is briefly compared to existing related work. Finally, in section VIII concluded to review the concept and content of the paper and suggestion for future enhancement in the algorithm is provided.

II. PROBLEM STATEMENT

This section highlights the problem statement which is extracted from the review made in this research. Following the solutions to these problems, the new algorithm is proposed.

Dealing with incoming user requests/tasks and keeping a balanced workload in cloud systems can be challenging due to inappropriate allocation to VMs. One cause of this is the limited task factors considered; for example, if the arrival time is not considered, all tasks would arrive simultaneously, which does not work in a dynamic environment such as cloud systems since the requests are not prioritized. With the increasing number of requests, more problems could occur if the requests are not assigned to their designated VM or when the CPU is not fully utilized or insufficient to handle the requests, leading to performance issues due to an unbalanced load in the cloud. To overcome these issues, it is necessary to consider QoS factors and provide an efficient algorithm to improve the cloud's performance in IaaS. This can be done by optimizing the usage of the system's resources, which reduces the Makespan and Execution time of user tasks.

The authors have limited the scope in this research to emphasize enhancing the cloud's performance in terms of Task Scheduling and Load Balancing. Based on the literature discussed in section III, the authors concluded the following points to address the research issues that have been resolved in the proposed work:

- Most researchers do not consider the priority, which is a critical factor in Task Scheduling. This will lead to issues such as an increase in Makespan time, which is the time taken to schedule a task/request, or an increase in the number of task rejections and latency [14]–[18].
- Although Task Scheduling is one of the main goals of providing an efficient Load Balancing and improving performance, most researchers focus on one or two aspects. For example, to enhance Load Balancing and considers few Task Scheduling parameters. Thus, only a few metrics are taken into consideration to improve the overall performance. This is an issue as improper Task Scheduling leads to an imbalanced load in the hosts [16], [17], [19], [20]. For example, if tasks arrive simultaneously following the FCFS algorithm's procedure, this could highly increase Makespan as the task will wait longer to finish executing. Each client may also send a different request; this should be indicated by providing random values for Task Length to make up a dynamic workload.
- Several new approaches have been made to improve Load Balancing; however, the workload migration challenge is still not fully addressed. Tasks are still allocated to VM regardless of its SLA violation state, which indicates it doesn't follow the specified Deadline and requirements stated in the agreement document [16], [21]. Each client receives a different SLA contract based on their needs from CSPs; hence, assigning random values for the Deadline parameter is crucial in scheduling since it can illustrate the algorithm's violation problem.

III. RELATED WORK

This section includes the literature review of this paper. The concept of Load Balancing will be explained, highlighting its model, metrics, and existing standard algorithms. Leading to the recent literature on Load Balancing, where researchers' proposed algorithms are explained and analyzed—followed by existing algorithms proposed by researchers in the field of Load Balancing.

An organization chart for section III is illustrated in figure 2 below. First, Task Scheduling and Load Balancing are discussed in the subsections, highlighting their importance in the cloud environment. Then, recent literature regarding their techniques is provided to highlight the limitations that are resolved in this proposed work.

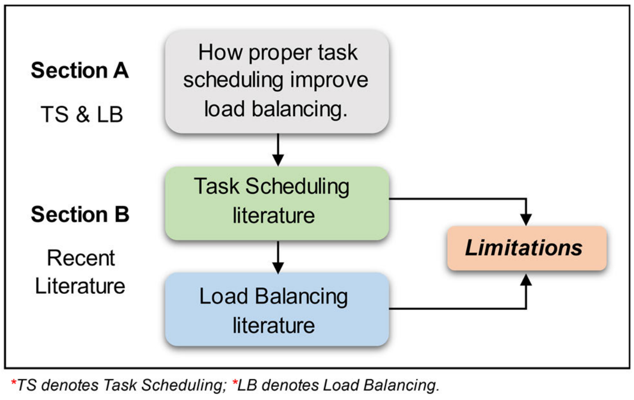

Section A
TS & LB
How proper task scheduling improve load balancing.
Section B
Recent Literature
Task Scheduling literature
Load Balancing literature
Limitations
*TS denotes Task Scheduling; *LB denotes Load Balancing.

FIGURE 2. The flow of related work.

A. TASK SCHEDULING & LOAD BALANCING

This subsection explains Task Scheduling and Load Balancing's concept to highlight how they relate to each other to optimize cloud resources.

Load balancing is a method for optimizing the resources of virtual machines in the Cloud Computing environment. Load balancing in the cloud environment is one of the critical techniques used to ensure an equal distribution of workload and efficient resource utilization. One crucial aspect required in the cloud environment to distribute dynamic workload among nodes is Load Balancing. The efficient balance of workload leads to higher user satisfaction and better resource allocation. In cloud systems, applying Load Balancing reduces delays in sending and receiving data [22]. Thus, it's essential to solve Load Balancing issues and enhance cloud application performance, which is discussed in more detail in this section.

A significant goal of Load Balancing is Task Scheduling. An increase in the number of clients using the cloud could lead to improper scheduling of jobs in the system [23]. Thus, issues around Task Scheduling should be resolved through a utilized algorithm, which is discussed in more detail in this section. Task scheduling is the process of executing tasks efficiently to utilize the resources of the system fully. In the cloud environment, users might use colossal amounts of virtualized resources, making it impossible to allocate each task manually [23].

Cloud computing services have become a vital part of big companies such as Google, Amazon, etc. Such services promise a flexible transfer or streaming of data all the time. However, the algorithms behind these necessary operations might be slow and raise some challenges or issues as the number of clients increase. Load balancing is a critical aspect of Cloud Computing technology; without it, users' tasks could be delayed and consume more time in terms of responses [24]. Since the technology is rapidly growing over time and big and small companies have adopted it, CSPs still face challenges due to unbalanced load situations and sometimes fail to deliver high-quality services to users. This could happen due to parameters such as high Makespan time that could degrade performance. Such issues could lead to threats on Service Level Agreements (SLA), an agreement document between service providers and the consumers [25] that can be easily violated if the performance of CC applications degrade. Such violations lead to starvation issues where the system is highly overloaded and incoming tasks cannot be served appropriately, and it can be rejected.

Thus, these issues should be addressed to reduce violations in SLA delivered by cloud providers to organizations. According to [26], there are a few factors that could lead to load unbalancing issues in IaaS clouds as listed below:

- No proper and accurate, or efficient mapping of tasks to appropriate resources/VMs.
- An inappropriate scheduling process can be a problem.
- Different task requirements for heterogeneous (various) user tasks.
- Unequal distribution of tasks to resources/VMs.

This paper aims to solve the above issues in the IaaS cloud platform by providing a dynamic Task Scheduling algorithm whereby important task requirements such as Deadline and Completion time are considered. These parameters are highly important as QoS factors. With proper scheduling and no VM violation, the algorithm results in a balanced workload in the cloud.

B. RECENT LITERATURE

This subsection provides a review of previous existing algorithms in the field of Load Balancing and Task Scheduling. Many recent algorithms aimed to improve Task Scheduling and Load Balancing. Yet, few limitations still exist due to the underlying basic algorithms used, such as Round Robin or First Come First Serve. These algorithms can increase the waiting time or Makespan in scheduling tasks.

Authors in [14] proposed a dynamic Load Balancingalgorithm to minimize the Makespan time and utilize resources efficiently. It sorts tasks using length and processing speed by using the bubble sort algorithm. Then, tasks are allocated to Virtual Machines in a First-Come-First-Serve order. After allocation is complete, balancing the load is done considering and calculating the load of Virtual Machines. This approach can easily optimize the resources and reduce Makespan; however, it does not consider priority or any QoS parameters such as Deadline.

Authors in [18] have proposed an algorithm where the load balancing concept is applied in a three-layer cloud computing network. The technique combines both Opportunistic Load Balancing (OLB) and Load Balance Min-Min (LBMM). With the ZEUS network framework's help, the algorithm improves OLB task scheduling by introducing a hierarchical network to process user tasks. In the first layer, the task is received and assigned to one service manager from the second layer. Lastly, the third layer is where the requests are divided into subtasks, speeding up the process. Assigning tasks to the service node depends on several attributes, such as the remaining CPU space, to check whether the node is available to handle such request. The approach helps to keep every node busy and working to serve the users' requests. However, it may be slow to process the request in a hierarchical form as it has to pass every layer of the framework.

The enhanced load balanced Min-Min (ELBMM) algorithm is proposed by authors in [15] authors to utilize resources. It looks for a request with the min execution time to allocate it to the VM with the min completion time; this way, it enhances the Min-Min algorithm. The advantage of this technique is to decrease the utilization cost and the system throughput.

A Resource-based Load balanced Min-Min (RBLMM) is another algorithm proposed in [18]. The algorithm is also developed to consider the reduction of Makespan and to balance the workload on Virtual Machines. Makespan time is calculated after resource allocation. The algorithm makes use of this value to define a threshold. The results obtained from this algorithm proves that RBLMM greatly reduced the Makespan time compare to the traditional Min-Min algorithm by 3 secs. While the above approaches significantly optimize resources, they mostly rely on allocating tasks in the same order manner, indicating no priority for tasks or Virtual Machines. Besides that, they do not focus on the QoS parameters that are vital for Task Scheduling, such as Deadline and priority.

In [27], the authors proposed efficient Scheduling and Load Balancing algorithms to minimize execution time to benefit the cloud users and service providers. The proposed algorithm is designed to select the VM with the lowest cost and considers the network latency in Data Centers. It chooses the best data center with minimum cost and workload.

The proposed algorithm known as State-Based Load Balancing (SBLB) can assign tasks to idle hosts dynamically; however, the broker algorithm does not consider SLA to handle dynamic user requests and resource allocation. The approach also results in high execution time [28].

To enhance the quality of service, authors in [21] designed a new QoS-based algorithm which allocates cloudlets with improved balancing technique to decrease further the Completion Time of tasks/cloudlets, the Makespan Time of Virtual Machines and host. It is great to make sure the system stays active, and the workload is balanced; however, it still results in high Makespan value for VMs and hosts. It is also not scalable in a large-scale environment as the experiment is built for 3 VMs only.

Another enhancement to another traditional approach known as the SJF scheduling algorithm is presented by researchers in [16]. The traditional Shortest Job First (SJF) has some limitations. It completes tasks with a small length first, resulting in starvation issues as the longer tasks are put in a waiting status and thus increases waiting time. This is resolved by allocating longer tasks to high response VM, and therefore it can reduce overall Makespan time effectively compared to traditional algorithms such as SJF and FCFS. However, in both previous approaches, the algorithm does not check the availability or the current load/status of the Virtual machine before allocating tasks to it. Besides that, tasks are being scheduled using the length parameter, which indicates no priority for the task is applied.

Another approach that works based on length of task and user priority is suggested in [19]. Researchers provide a credit system that works by selecting a middle task, which means it selects neither the highest nor lowest task length, but it focuses on the mid-value. The value is found by taking the difference value of the length taken based on the average of all task requests; later, it assigns credit to the task. The approach still relies on the task's size and overlooks important quality parameters, for example, Deadline.

Researchers in [17] presented a multi-objective algorithm for the improvement of throughput in cloud models. It assigns a high priority to Virtual Machines with a high value of Million Instruction Per Second (MIPS). It sorts tasks in descending order by allocating the first task from the list to the first Virtual Machine and so on. The algorithm follows the QoS requirements; however, it considers very few parameters, such as execution time.

A Grouped Tasks Scheduling (GTS) algorithm was introduced in [20]. It applies QoS parameters such as user type, expected priority, length, and latency of tasks. Tasks with similar parameters will be categorized into five groups (urgent user & task, urgent user, urgent task, long task, and finally normal task). Giving high priority to tasks in the first group, GTS improves latency whenever an urgent number of tasks increases. However, it may not be suitable for tasks that depend on a particular order or other scheduling tasks.

Researchers in [29] proposed a priority algorithm based on task length to resolve the starvation issue for small tasks. Giving the highest priority to the smaller tasks optimizes resources. The result shows that the waiting time decrease when the number of VMs is increased. The approach may be inefficient when large tasks, and since the length is used, no priority is enforced.

Authors in [30] proposed a distributed LB algorithm with an adaptive threshold. The authors introduce a starvation threshold to enforce a transfer policy for the migration process. A VM with high delay results in a high starvation value. This helps in balancing the load between VMs. Results show that the STLB algorithm can significantly reduce the number of migrations compared to the nature-inspired baseline algorithm Honey-bee behavior.

In [31], the authors presented a CMLB load balancing algorithm for reallocation of tasks to VMs in case of imbalance situations. The approach uses the Dragonfly optimization algorithm to define the optimal threshold value. Based on this value, the load of VM is compared and determined. Results show that the algorithm has better performance as it migrates only three cloudlets than other methods such as Honey-Bee and dynamic LB.

IV. PROPOSED WORK

This section explains the proposed and improvised Load Balancing in Cloud Computing Environment. This algorithm's primary goal is to provide services of high quality to clients in Cloud Computing applications. The method consists of both processes: Task Scheduling process to assign deadline and completion time to cloudlets (tasks) and secondly, Load Balancing process to perform migration of workload in case of VM violation to maintain a balanced load in the cloud environment.

A. PROPOSED FRAMEWORK

In this subsection, we describe this research's objective in an illustrative diagram to explain the problem in Load Balancing and the role of the proposed LB algorithm, as seen in figure 3 below.

This proposed model's main goal is to provide efficient resource allocation in a cloud environment whereby it avoids unbalanced workload in Cloud Computing applications. This model resolves issues related to workload migration and task rejection in the cloud. The proposed framework consists of two layers:

- **Top Layer**: deals with requests from multiple different clients (application's users) of both mobile and desktop. Clients can access the Internet using different devices to send requests to the cloud. In this layer, the model uses the Cloudlet Scheduler Time Shared algorithm to submit tasks in a random order (Arrival Time) and schedule them to Virtual Machines by considering two main parameters: Deadline and Completion Time. In Cloud Computing, Data Center (DC) can be described as big storage for cloud servers and data. DC receives requests and sends them to the active load balancer. In this layer of the model, the proposed algorithm is implemented

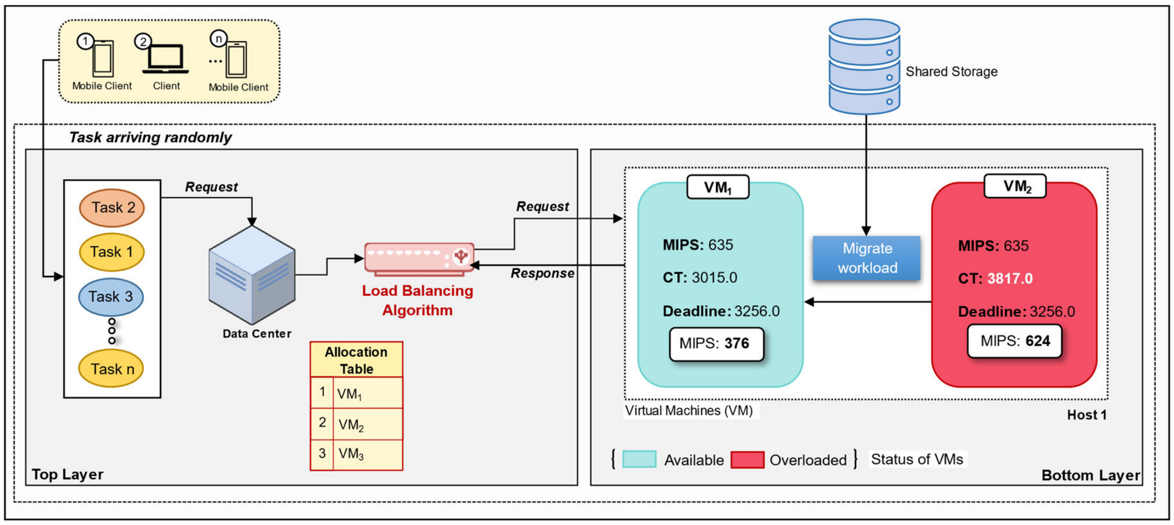


FIGURE 3. Proposed framework.

as a Load Balancer, which acts as the primary balancer in the cloud environment to perform migration in the case of violation, which has not been addressed in the previous literature up to the author's knowledge.

- **Bottom Layer**: deals with allocation of user requests to Virtual Machines (VMs). As the figure illustrates, we have a primary batch of VMs; VM2's status is set to high priority since it violates the SLA requirement, which means its Completion Time is higher than the Deadline. Thus, the proposed LBA should apply a migration technique to transfer the workload to another available Virtual Machine by reconfiguring the MIPS of both VMs before and after allocating the resources to them. The allocation table is then updated whenever a Virtual Machine becomes violated or not, along with the number of requests it's been allocated. There is a case where there is no SLA violation. Suppose the Time to Complete (TTC) is less than SLA (Deadline) given for tasks to run on VMs. Then, no SLA violation occurs.

Overall, the proposed framework supports dynamic scheduling and load balancing to fully utilize the CPU and fully the cloud resources.

B. PROPOSED LOAD BALANCING ALGORITHM

In this subsection, the proposed LB algorithm is explained to highlight the assumptions made in the implementation, the algorithm's pseudocode, and finally, the flowchart.

The proposed algorithm aims to improve the cloud's performance by considering both aspects of Task Scheduling and Load Balancing. It utilizes all available CPUs in machines and schedules tasks appropriately to reduce Makespan, Execution Time, and maximize resource utilization. Below are the assumptions made in the proposed algorithm:

- One-to-many cloudlets (also known as task or user request) per Virtual Machine (VM).
- Cloudlets arrive in a random order (Arrival Time)
- Each Cloudlet has a length, a time to complete known as Deadline (included in Service Level Agreement document), a completion time, and finally, the arrival time.
- The proposed algorithm checks the completion time for each workload (a total of cloudlets) against the Deadline.
- If there is any violation, whereby the completion time exceeds the Deadline, then the proposed algorithm will reconfigure the VM's priority based on its CPU. If it is in a successful state, the cloudlets get scheduled else; it will migrate the VM's workload.
- Expected Completion Time is calculated by taking the cloudlet length (also known as Million instruction per second (MIPS)) and dividing it by Virtual Machine MIPS (also known as CPU).
- Initially, all VMs share an equal portion of the available CPU; then, it is reconfigured based on the violation status. The CPU is set to its full utilization in the proposed algorithm.

Table 2 below shows the terms used in the proposed algorithm and their meanings.

The Pseudocode of the proposed LB Algorithm is provided below. The purpose of providing the pseudocode is to illustrate the formulas, the parameters, and the decisions made in this Load Balancing algorithm.

As can be seen from the algorithm steps above, there are input and output to every algorithm. In this research, the input is mainly two random values for task length and Deadline, which is an essential factor in the SLA document. SLA is an important document considered by CSPs that denotes the number of reductions of SLA violation factors [33] in terms of deadline constraint, priority, etc. The algorithm's main goal/output is to achieve a workload balance among VMs in cloud systems, migrate and reallocate resources in case of SLA violation. Step 5 is to assign an equal por-

TABLE 1. An overview of reviewed algorithms.

Ref.	Strengths	Weaknesses	Authors & Year	Research Gap
Dynamic Load Balancing (DLBA) [14]	Efficient utilization of resources; Reduce Makespan;	Uses FCFS for the allocation of requests; No prioritization applied.	(Kumar & Sharma, **2017**)	More optimization to the scheduling technique to consider the QoS.
CA, LA, LAOC & SBLB [27]	Combines both Load Balancing and broker techniques; Reduces response & processing time.	Low performance; high execution time; does not consider SLA.	(Naha & Othman, **2016**)	Further improvement to dynamically allocate resources.
ELBMM [15]	Efficient utilization of resources; lower Makespan.	Does not apply any priority policy for tasks or VMs.	(Patel et al., **2015**)	Improvement needed to consider QoS.
RBLMM [18]			(Shanthan & Arockiam, **2018**)	
OLB & LBMM [32]	Dynamic algorithm; Minimizes the execution time for tasks; Keep every node busy & improves load balancing.	May be slow due to its hierarchical nature to process the requests.	(Wang et al., **2010**)	Improvement can be made to avoid delays in processing user requests.
QoS-Based Cloudlet Allocation Strategy [21]	Efficient utilization of system resources; Reduced Makespan; Considers Quality of Service; Balanced workload; Lower completion time of Cloudlet.	May not be scalable, uses only 3 VMs; Still suffer from high Makespan time; Does not resolve balancing situations such as overloading of VMs	(Banerjee et al., **2015**)	Further enhancement to reduce the Makespan & avoid system congestion.
MSJF [16]	Maximize resource utilization; Sends the longest task to the fastest VM; Manage load balancing between VMs.	Task is allocated based on length; no priority enforced; does not check VM condition before scheduling.	(Alworafi & Dhari et al., **2016**)	Need to consider other parameters for tasks such as Deadline and must check whether VM is free or not before the allocation process.
Credit-Based Scheduling [19]	Reduced Makespan; Resolves issues for tasks with the same priority.	Does not consider task deadline.	(Thomas & G, **2015**)	More parameters can be considered to enhance the scheduling process.
Multi-Objective Task Scheduling [17]	Reduce wastage of resources; Improve throughput of datacenter; Reduce violations in SLA.	No task priority is enforced; Considers few QoS parameters.	(Lakra & Yadav, **2015**)	Further optimization required to reduce Makespan when a larger no. of VMs used.
GTS [20]	Considers equal distribution of load; Categorizes and prioritize tasks.	Only works for independent tasks; Does not consider Deadline.	(Ali & Saroit et al., **2016**)	Further improvement to the QoS.
Optimal task scheduling [29]	Reduces starvation for small tasks; Considers priority of tasks.	High waiting time for large tasks.	(Monika & Jindal, **2016**)	Further enhancement to reduce waiting time of tasks.
STLB [30]	Minimize response time; maximize utilization rate of servers; reduce overall migration cost; maintain a stable system.	May not be suitable for dependent tasks in scheduling; high communication costs.	(Semmoud et al., **2019**)	Further enhancement to provide better scheduling techniques.
CMLB [31]	Reduce the number of task migration;	Can handle tasks only within the specified threshold limit.	(Polepally & Chatrapati, **2017**)	More optimization for better threshold technique.

tion of MIPs to each VM. MIPS is required to allocate the host CPU. Then, each task has a completion time, which results from dividing the total VM length by its MIPS as stated in step 9. To check for SLA violation, each VM has a violation cost that is calculated by deducting the Deadline from completion time, as shown in step 10. If there are 2 VMs, the algorithm checks for the higher violation cost then gives it a high priority. CPU will be reconfigured for that VM, and the workload from the VM is migrated if the MIPS available on the host is insufficient to run the VM.

Following figure 4 shows the flow diagram of the proposed SLA-LB Algorithm. The steps illustrate the process of Load Balancing and Task scheduling, whereby it includes processes, the decisions to be made in the algorithm, and the output of the findings.

The algorithm's flow starts by assigning random values for the length of the task. These values are below the threshold

Algorithm 1 Proposed Load Balancing

```
1.  Input: Random Integers $L_i$ and $D_i$, where $L \geq D$
2.  Output: Mapping of resources (Cloudlets) to appropriate Virtual Machines (VM)
3.  Begin
4.  for each VM
5.      Assign an equal portion of MIPS
6.  end for
7.  for $i = 1$ to $M$
8.      for $j = 1$ to $N$
9.          $C_{ij} = L_i / \text{MIPS}_j$
10.         $V_{ij} = |C_{ij} - D_i|$
11.     end for
12. end for
13. do until all tasks are allocated to the appropriate VM
14.     If $VM_s[[space]] \geq [[space]]VM_{s+1}$, where $S \leq 6$ then
15.         Reconfigure $VM_i$ CPU
16.         If MIPS available on the host to run $VM_i$ is insufficient, then
17.             Migrate Workload
18.             If $VM_i$ is migrated, then
19.                 Assign 0
20.             Else
21.                 Recompute $C_{ij}$ for each $VM_i$
22.         Else
23.             Allocate resources to $VM_i$
24.     Else
25.             No violation occurs
26.     End if
27. End do
28. Update the ready queue and expected $C_{ij}$ of corresponding $VM_i$
29. Compute Avg. ExT for all tasks, MT for each $VM_i$ & RU
30. End
```

TABLE 2. Denote terms and meaning.

Term	Meaning
C_{ij}	The completion time of requests in each VM.
MIPS_j	Millions of instructions per second of VM.
L_i	Task Length.
D_i	Task Deadline.
V_{ij}	VM violation cost.
ExT	Execution Time.
MT	Makespan.
RU	Resource Utilization.

of 1000000 and below 2000 for Deadline. These values act as an input to the algorithm, and they make up the workload in cloud systems (requests from clients). Each Virtual Machine

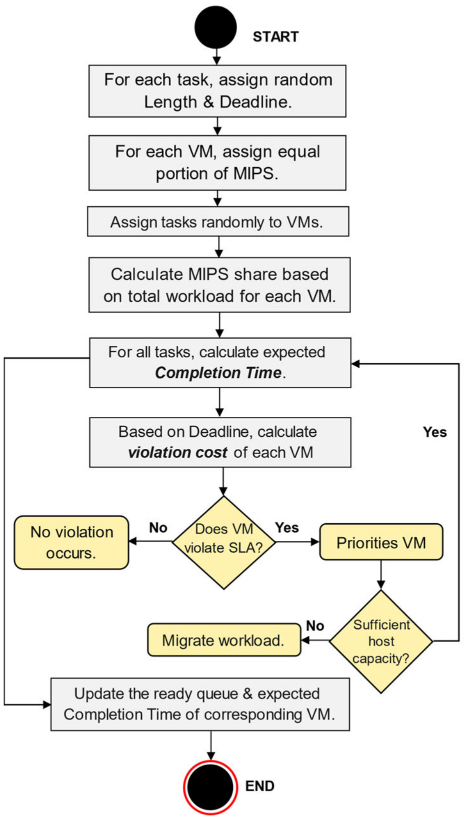

START
For each task, assign random Length & Deadline.
For each VM, assign equal portion of MIPS.
Assign tasks randomly to VMs.
Calculate MIPS share based on total workload for each VM.
For all tasks, calculate expected Completion Time.
Based on Deadline, calculate violation cost of each VM
Yes
No violation occurs.
No
Does VM violate SLA?
Yes
Priorities VM
Migrate workload.
No
Sufficient host capacity?
Update the ready queue & expected Completion Time of corresponding VM.
END

FIGURE 4. Flowchart of the proposed algorithm. Flowchart of the proposed algorithm.

is then assigned an equal share of CPU based on the total workload. By calculating the VM cost, we can determine whether the VM has violated SLA requirements by monitoring if the completion time is higher than the Deadline. If yes, then the workload is migrated to another VM, and so on. This way, the algorithm fully utilizes all CPU, and the system workload is balanced.

V. IMPLEMENTATION

In this section, the experimental results and implementation details of the proposed LBA algorithm are described.

A. SIMULATION SETUP

CloudSim simulation tool is the most popular tool used by researchers and developers nowadays for cloud-related issues in the research field. It can significantly eliminate the need and expenses of computing facilities [77] for performance evaluation and modeling the research solution. This simulation tool is an external framework that can be downloaded and imported to programming software such as Eclipse, NetBeans IDE, Maven etc. To simulate the Cloud Computing environment, the CloudSim toolkit is integrated into NetBeans IDE 8.2, and the Operating System used is Windows 10.

The entities and computing resources were virtually modeled to reflect a scenario of scheduling and load balancing in a cloud environment to evaluate the proposed algorithm's performance. The experiments were implemented with 2 Datacenters, 2-6 VMs, and 2-40 cloudlets (tasks) under the simulation platform. The task's length is generated randomly below an upper threshold of 1000,000 Million Instructions (MI). Processor speed, available memory space, and bandwidth determine the acceptable workload for each VM. The parameters required for the setting of CloudSim are summarized in Table 2 and 3 below.

TABLE 3. Hardware requirements.

Component	Specification
Operating System	Windows (X64 based Processor) 64-bit OS
Processor	Intel® Core ™ CPU @ 1.00 GHz 1.19 GHz
RAM	16.0 GB

B. PERFORMANCE METRICS

The performance of the proposed LB algorithm was analyzed based on three parameters under the cloud environment. The following performance matrix is used to measure and evaluate the performance:

1) Makespan (MT): it is the total time taking to get a cloudlet scheduled. This is mostly used to measure scheduling algorithms' efficiency with respect to time [34], [35]. It should be reduced to allow efficient execution of tasks and to release the resources for other tasks. It is measured by using the equations below proposed by authors in [36], where *CT* denotes cloudlet completion time, and *n* denotes the number of Virtual Machines.

$$MT = Max(CT)$$

$$MT_{avg} = \left(\frac{\sum Max(CT)}{n}\right)$$

2) Execution Time (ExT): it is the exact time taken to execute the given tasks (cloudlets) on a virtual machine [16]. This metric should be reduced to achieve better performance of the algorithm. It is measured by using the equations below proposed by authors in [36], where *AcT* denotes Cloudlet Actual CPU Time and *n* denotes the number of Cloudlets.

$$ExT = AcT$$

$$ExT_{avg} = \left(\frac{\sum AcT}{n}\right)$$

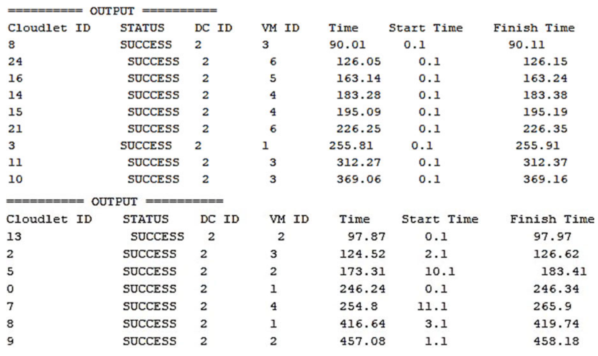
3) Resource Utilization (RU): this is another quantitative metric that depends on the above metrics. It is measured to increase the efficiency in utilizing the resources in the cloud environment. It is calculated using the equations below proposed by authors in [37], where *ExT* denotes total execution time, and *MT* denotes total Makespan. The average resource utilization can determine how efficient the proposed algorithm in terms of utilizing the CPU. The range of this metric is 0 to 1, the maximum value is the best case, which is 1, this indicates 100% resource utilization, and the worst-case value is 0, which means the resources are in ideal condition.

$$RU = \left(\frac{ExT}{MT}\right)$$

$$RU_{avg} = \left(\frac{ExT}{MT}\right) \times 100$$

VI. RESULTS & DISCUSSION

The purpose of carrying this experiment is to prove the reduction in Makespan, execution time, and the increasing of resource utilization in a dynamic cloud environment. During the testing of the algorithm, we have considered preemptive scheduling of tasks. This means the task can be interrupted during execution if the workload violates SLA, it can be migrated to another resource to complete execution, as shown in figure 5. During the scheduling process, several QoS performance parameters of cloudlets are considered, such as:



```
========== OUTPUT ==========
Cloudlet ID    STATUS    DC ID    VM ID    Time     Start Time    Finish Time
8              SUCCESS   2        3        90.01     0.1           90.11
24              SUCCESS   2        6        126.05    0.1           126.15
16              SUCCESS   2        5        163.14    0.1           163.24
14              SUCCESS   2        4        183.28    0.1           183.38
15              SUCCESS   2        4        195.09    0.1           195.19
21              SUCCESS   2        6        226.25    0.1           226.35
3              SUCCESS   2        1        255.81    0.1           255.91
11              SUCCESS   2        3        312.27    0.1           312.37
10              SUCCESS   2        3        369.06    0.1           369.16
========== OUTPUT ==========
Cloudlet ID    STATUS    DC ID    VM ID    Time      Start Time    Finish Time
13              SUCCESS   2        2        97.87     0.1           97.97
2              SUCCESS   2        3        124.52     2.1           126.62
5              SUCCESS   2        2        173.31     10.1           183.41
0              SUCCESS   2        1        246.24     0.1           246.34
7              SUCCESS   2        4        254.8      11.1          265.9
8              SUCCESS   2        1        416.64     3.1           419.74
9              SUCCESS   2        2        457.08     1.1           458.18
```

FIGURE 5. Same arrival time & random arrival time.

1) Arrival Time: indicates the time cloudlets arrive or when the algorithm receives the user request. This is known as the cloudlet start time in the CloudSim environment. In CloudSim, by default, all cloudlets arrive at the broker at the same arrival time. In this experiment, this has been modified to make changes for postponing the submission of cloudlets; this is known as a random Arrival Time parameter. The broker will then assign the cloudlets in a random order to the VMs based on the code implemented in this method. Using this parameter, we can design an algorithm to function in a dynamic environment where the arrival time can be different for each request.

2) Task Length: identifies the size of tasks in bytes; smaller tasks lead to more resource utilization. In CloudSim, each Cloudlet must have a length value that indicates the cloudlet type, whether it is a heavy request, light, or medium. In this experiment, length has been identified and assigned randomly to each Cloudlet. All the cloudlets should have random values to differentiate the client requests from each other. This

can be done by defining the length as a random value to represent the cloud environment's total workload. The length parameter is an essential input in the experiment to determine the load for each Virtual Machine. Based on this parameter, the Time to Complete requests in each VM can be identified. Based on this, we can determine if there's a violation in SLA.

3) Deadline: the maximum amount of time given to the task to execute. It is one of the most important aspects considered by CSPs in SLA. In this experiment, each Cloudlet has a different deadline value, which means each client gets a different SLA contract based on their needs and service expectations from the cloud providers. Thus, it is suggested to use random deadline value instead of static. Deadline is an important parameter as it represents SLA; if the Time to Complete requests exceeds the Deadline, we can identify that there is a violation in the SLA.

Sample values that make up the workload based on the above parameters (Task Length & Deadline) are illustrated in Table 4 below.

TABLE 4. CloudSim simulator requirements.

Type	Parameters	Value
Cloudlet (Task)	Length of tasks (in bytes)	Random < upper threshold (1000000)
	Total number of tasks	1-40
Virtual Machine (VM)	Number of VMs	2-6
	Processor speed	9980-15000 MIPS
	RAM in a single VM	512 Mb
	Bandwidth	1000 Mb
	Cloudlet Scheduler	Time Shared
	Number of processor elements (PEs) requirement	1
	VMM	Xen
Data Center	Number of Data Centers	2
	Number of Hosts	1
	VmScheduler	Time Shared

In this section, the performance of the proposed LBA algorithm was recorded by taking three different test cases: (1) 2 Virtual Machines with 10 to 40 cloudlets; (2) 4 Virtual Machines with 10 to 40 cloudlets; (3) 6 Virtual Machines with 10 to 40 cloudlets. The increment of these variables in simulation can enhance the scheduling process and workload migration among different VMs.

The average Makespan, Execution time, and resource utilization was recorded in each case as different values are considered for the parameters of tasks such as Deadline, arrival time, and length of the task. The experiment is carried out in a homogenous cloud environment whereby all VMs have the same capacity in each test case. The total MIPS in table 3 is used to set the CPU that is equally shared among VMs. Based on the violation of a VM, this is then adjusted to reallocate the resources efficiently.

The results are recorded in Table 4 to VI to highlight the experiment's achieved values in each iteration of increasing the cloudlet and VMs variables.

TABLE 5. A sample of some task properties.

Cloudlet ID	Task Length	Deadline	File Size	Output Size	No. of CPU
0	350483	219	300	300	1
1	48295	657	300	300	1
2	363835	1995	300	300	1
3	752057	710	300	300	1
4	896159	317	300	300	1
5	560818	654	300	300	1
6	764075	1543	300	300	1
7	578332	1966	300	300	1
8	295922	1837	300	300	1
9	735446	1249	300	300	1

As shown in Table 4, the algorithm's performance varies in each case the cloudlets are increased. The results obtained for 2 VMs reveal the algorithm has a minimum and maximum Makespan of 261 ms and 893 ms, respectively, whereas the Execution time is 196 ms and 607 ms, respectively.

As shown in Table 6, the results obtained for 4 VMs reveal that the algorithm has a minimum and maximum Makespan of 271 ms and 895 ms. In contrast, the Execution time is 206 ms and 615 ms, respectively.

As shown in Table 7, the results obtained for 6 VMs reveal the algorithm has a minimum and maximum Makespan of 304 ms and 896 ms, respectively. In contrast, the Execution time is 252 ms and 639 ms, respectively.

The results are graphically represented in figure 6 (a) and (b), where the x-axis represents the number of cloudlets (tasks) and the y-axis represents the Makespan or Execution time in milliseconds. The graphs conclude that these parameters (execution time and Makespan) are affected when the number of cloudlets and VMs increases. However, it does not result in a huge difference, which shows the proposed LB algorithm's stable performance in such conditions. It is concluded that Makespan increases with the number of cloudlets. In contrast, Execution time depends on the cloudlet Actual CPU time, which is the total execution time of the Cloudlet in a cloud resource (aka. VM). Hence, it can fluctuate in each case based on this.

The proposed algorithm also improves resource utilization in the cloud environment. As shown in figure 7 below, the algorithm results in an average of 77% RU for 6 VMs and 40 tasks. The RU value can vary in each case due to the different Maksepan and Execution time.

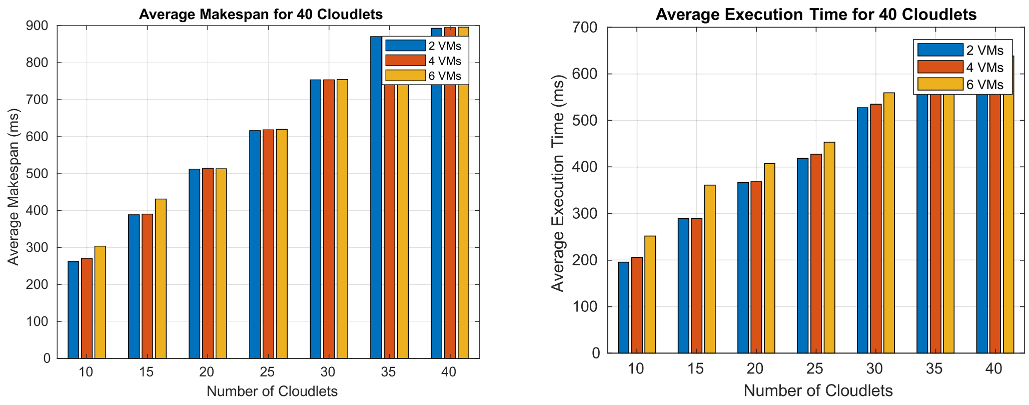


FIGURE 6. (a) Average Makespan for 40 tasks; (b) Average Execution time for 40 tasks.

TABLE 6. Results were obtained for 2 VMs with 10 To 40 tasks.

Number of cloudlets	Average Makespan (ms)	Average Execution Time (ms)	Resource Utilization (%)
10	261.4400938	195.9367204	75
15	388.8291983	289.1025092	74
20	512.0973245	366.7708264	72
25	616.2059895	418.7846801	68
30	753.4639850	527.36626	70
35	870.4258042	618.4638792	71
40	892.7444428	607.1213388	68

TABLE 7. Results were obtained for 4 VMs with 10 To 40 tasks.

Number of cloudlets	Average Makespan (ms)	Average Execution Time (ms)	Resource Utilization (%)
10	270.6467798	205.598844	76
15	390.3435289	289.4019536	74
20	514.1474467	368.5742998	72
25	618.2197241	427.3171608	69
30	753.4639850	534.8518396	71
35	870.0429147	630.6373105	72
40	894.851356	614.8845139	69

TABLE 8. Results were obtained for 6 VMs with 10 To 40 tasks.

Number of cloudlets	Average Makespan (ms)	Average Execution Time (ms)	Resource Utilization (%)
10	303.5410115	251.6623927	83
15	430.8096146	361.2242922	84
20	512.9966944	407.6082266	79
25	577.6690028	423.5005818	73
30	754.0924138	559.1548208	74
35	869.503937	650.4350047	75
40	896.0297375	638.7887158	71

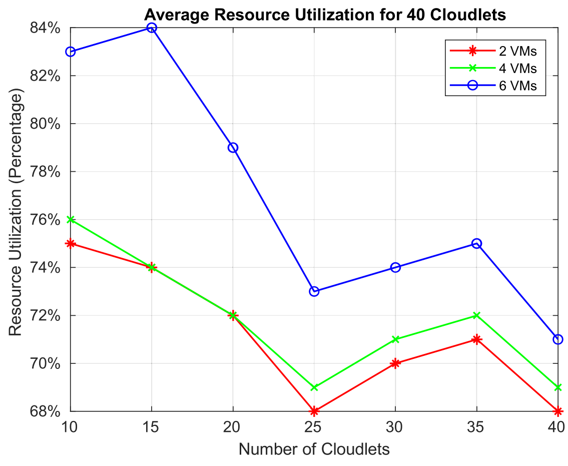


FIGURE 7. Average Resource Utilization for 40 tasks.

VII. RESULTS COMPARISON

To analyze the results of the recent research algorithm and the proposed algorithm. The main parameter considered for comparison in this research is Makespan Time. The main objective of the proposed Load Balancing algorithm is to enhance the utilization and allocation of cloud resources and minimize the time taken to schedule a task for improving the performance of the cloud applications. This section provides a general comparison of the existing related work and the proposed algorithm in this research.

The proposed work has been compared with the Dynamic Load Balancing algorithm proposed in [14]. This algorithm was developed in 2017 to efficiently allocate the cloudlets to

Virtual Machines and optimize the Makespan. It efficiently utilizes the resources in the cloud environment. This algorithm is chosen for comparison since it is closely related to the objectives of this research and the implementation setup. Part of the future work for the Dynamic LB algorithm was to consider QoS parameters or priority; our algorithm considered such parameters to illustrate the difference in results if these parameters are used. Both algorithms have used parameters such as cloudlet Length and Completion Time. However, the algorithm has utilized the First Come First Serve (FCFS) method for scheduling tasks, resulting in higher waiting time for tasks as it does not provide any priority. The proposed LB algorithm considers different arrival times and deadlines to follow up with the SLA document in line with QoS parameters for better service in cloud applications.

The results are compared based on Makespan, as shown in figure 8, where the y-axis represents the Makespan value in milliseconds, and the x-axis represents the number of cloudlets (tasks). As can be seen, results are obtained for 40 tasks in total. The graph shows that Makespan in our proposed algorithm increases in the case of 25-40 tasks; this is due to the huge range of task length considered in the experiment. The proposed LB algorithm handles requests of larger tasks of 1000,000 MI length, whereas Dynamic LBA is only within 400,000 MI. Since Makespan depends on the load of the VMs, increasing the task length will increase the Makespan as well. However, if a smaller size is considered in our proposed algorithm, it will reduce Makespan compared to Dynamic LBA for the case of 25-40 tasks.

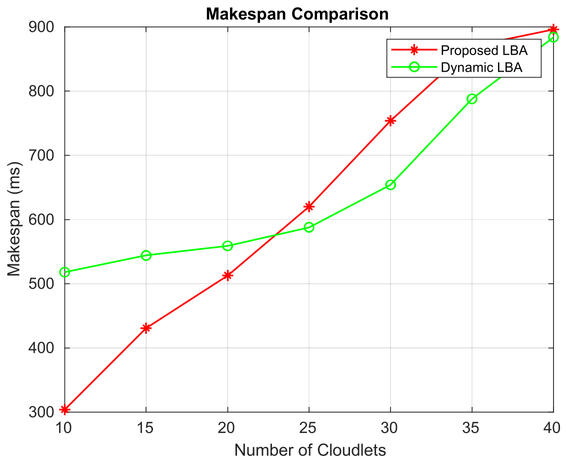


FIGURE 8. Experimental results comparison for Makespan time at 6 VMs with existing Dynamic LBA algorithm.

Results are also compared based on resource utilization, as shown in figure 9, where the y-axis represents the Resource utilization in percentage value, and the x-axis represents the number of cloudlets (tasks). In [14] Dynamic LBA, the figure shows average resource utilization of approximately 76% for 40 tasks in 5 VMs, whereas our proposed algorithm results in 78% utilization of resources in 6 VMs, which is slightly improved.

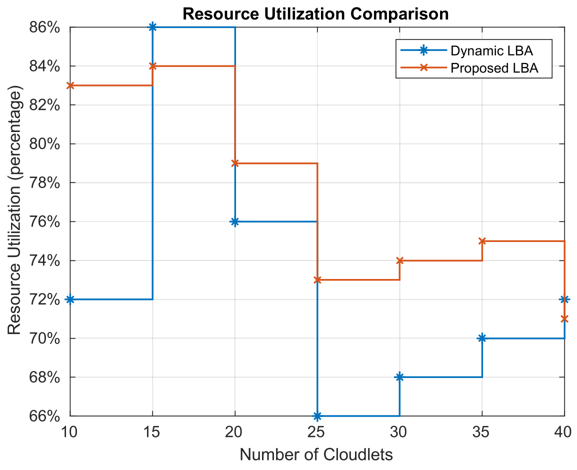


FIGURE 9. Experimental results comparison for Resource Utilization with existing Dynamic LBA algorithm.

In the conclusion of the results, it is proven that considering QoS parameters such as the Deadline can significantly improve the utilization of resources, reducing the Makespan and providing an efficient allocation technique in VMs. In addition, our proposed workload Balancing Algorithm for the Data Centers to Optimize Cloud Computing Applications could help to the different applications including location aware services [38], live streaming and recording cloud based [39] services, etc.

VIII. CONCLUSION & FUTURE WORK

This section concludes the paper by highlighting the findings and obtained results from the proposed LB algorithm.

As we saw from the literature, task scheduling highly contributes to balancing the load in a cloud environment. Improving the Load Balancing process through Task Scheduling can result in efficient utilization of cloud resources. The objective of this paper was to provide an enhanced Load Balancing algorithm. Results proved that our algorithm reduces Makespan and provide efficient resource utilization of 78% compared to existing Dynamic LBA. It also shows that the proposed algorithm can function in a dynamic cloud environment where user requests arrive in random order and where there are many changes in the length of the user requests. The algorithm is also able to handle large size requests compared to the existing approach. The algorithm address SLA violation of VMs by reallocating resources to execute tasks efficiently.

In the future, authors will work to optimize the cloud resources further and enhance cloud-based application performance, such as considering more SLA parameters. For example, the algorithm will be tested based on the number of violations and the migration count for better performance. Also, the algorithm will be comprehensively compared to other existing algorithms in the literature.

ACKNOWLEDGMENT

The authors would like to express their greatest gratitude and thanks to Taylor's University, School of Computer

Science & Engineering (SCE) for giving them the special opportunity to carry out this research successfully.

REFERENCES

[1] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "Cloud computing virtualization of resources allocation for distributed systems," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 3, pp. 98–105, Jun. 2020, doi: 10.38094/jastt1331.

[2] M. Agarwal and G. M. Saran Srivastava, "Cloud computing: A paradigm shift in the way of computing," *Int. J. Mod. Educ. Comput. Sci.*, vol. 9, no. 12, pp. 38–48, Dec. 2017, doi: 10.5815/ijmecs.2017.12.05.

[3] N. Zanoon, "Toward cloud computing: Security and performance," *Int. J. Cloud Comput.: Services Archit.*, vol. 5, no. vol. 5, nos. 5–6, pp. 17–26, Dec. 2015, doi: 10.5121/ijccsa.2015.5602.

[4] C. T. S. Xue and F. T. W. Xin, "Benefits and challenges of the adoption of cloud computing in business," *Int. J. Cloud Comput.: Services Archit.*, vol. 6, no. 6, pp. 1–15, Dec. 2016, doi: 10.5121/ijccsa.2016.6601.

[5] D. A. Shafiq, N. Jhanjhi, and A. Abdullah, "Proposing a load balancing algorithm for the optimization of cloud computing applications," in *Proc. 13th Int. Conf. Math., Actuarial Sci., Comput. Sci. Statist. (MACS)*, Dec. 2019, pp. 1–6, doi: 10.1109/MACS48846.2019.9024785.

[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, 2020, doi: 10.1016/j.jksuci.2018.01.003.

[7] I. Odun-Ayo, M. Ananya, F. Agono, and R. Goddy-Worlu, "Cloud computing architecture: A critical analysis," in *Proc. 18th Int. Conf. Comput. Sci. Appl. (ICCSA)*, Jul. 2018, pp. 1–7, doi: 10.1109/ICCSA.2018.8439638.

[8] A. Jyoti, M. Shrimali, and R. Mishra, "Cloud computing and load balancing in cloud computing -survey," in *Proc. 9th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2019, pp. 51–55, doi: 10.1109/confluence.2019.8776948.

[9] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS ONE*, vol. 12, no. 5, May 2017, Art. no. e0176321, doi: 10.1371/journal.pone.0176321.

[10] M. Adhikari and T. Amgoth, "Heuristic-based load-balancing algorithm for IaaS cloud," *Future Gener. Comput. Syst.*, vol. 81, pp. 156–165, Apr. 2018, doi: 10.1016/j.future.2017.10.035.

[11] B. Singh and G. Singh, "A study on virtualization and hypervisor in cloud computing," *Int. J. Comput. Sci. Mobile Appl.*, vol. 6, no. 1, pp. 17–22, 2018.

[12] 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, doi: 10.1016/j.jnca.2019.06.006.

[13] F. Zabini, A. Bazzi, B. M. Masini, and R. Verdone, "Optimal performance versus fairness tradeoff for resource allocation in wireless systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2587–2600, Apr. 2017, doi: 10.1109/TWC.2017.2667644.

[14] 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, doi: 10.1080/1206212X.2017.1404823.

[15] 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, 2015, doi: 10.1016/j.procs.2015.07.385.

[16] M. A. Alworafi, A. Dhari, A. A. Al-Hashmi, and A. B. Darem, "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, doi: 10.1109/ICEECCOT.2016.7955216.

[17] A. V. Lakra and D. K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," *Procedia Comput. Sci.*, vol. 48, pp. 107–113, 2015, doi: 10.1016/j.procs.2015.04.158.

[18] B. J. H. Shanthan and L. Arockiam, "Resource based load balanced min min algorithm (RBLMM) for static meta task scheduling in cloud," in *Proc. IC-ACT*, 2018, pp. 1–5.

[19] A. Thomas, G. Krishnalal, and V. P. Jagathy Raj, "Credit based scheduling algorithm in cloud computing environment," *Procedia Comput. Sci.*, vol. 46, pp. 913–920, 2015, doi: 10.1016/j.procs.2015.02.162.

[20] 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, doi: 10.1016/j.eij.2016.07.002.

[21] 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, doi: 10.1007/s13369-015-1626-9.

[22] R. Kaur and P. Luthra, "Load balancing in cloud system using max min and min min algorithm," in *Proc. Nat. Conf. Emerg. Trends Comput. Technol. NCETCT*, vol. 1, 2014, pp. 31–34.

[23] 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, doi: 10.1016/j.future.2018.09.014.

[24] P. Kathalkar, "Challenges & issues in load balancing in cloud computing," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 6, no. 4, pp. 963–968, Apr. 2018, doi: 10.22214/ijraset.2018.4163.

[25] S. Afzal and K. Ganesh, "A taxonomic classification of load balancing metrics: A systematic review," in *Proc. 33rd Indian Eng. Congr.*, Jan. 2019, pp. 85–90.

[26] S. Afzal and G. Kavitha, "Load balancing in cloud computing–A hierarchical taxonomical classification," *J. Cloud Comput.*, vol. 8, no. 1, p. 22, 2019, doi: 10.1186/s13677-019-0146-7.

[27] R. K. Naha and M. Othman, "Cost-aware service brokering and performance sentient load balancing algorithms in the cloud," *J. Netw. Comput. Appl.*, vol. 75, pp. 47–57, Nov. 2016, doi: 10.1016/j.jnca.2016.08.018.

[28] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–35, Feb. 2019, doi: 10.1145/3281010.

[29] A. Jindal, "Optimization of task scheduling algorithm through QoS parameters for cloud computing," in *Proc. ICAET*, vol. 57, 2016, pp. 1–4, doi: 10.1051/matecconf/20165702009.

[30] A. Semmoud, M. Hakem, B. Benmammar, and J. Charr, "Load balancing in cloud computing environments based on adaptive starvation threshold," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 11, pp. 1–14, Jun. 2020, doi: 10.1002/cpe.5652.

[31] 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, doi: 10.1007/s10586-017-1056-4.

[32] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, Jul. 2010, pp. 108–113, doi: 10.1109/ICCSIT.2010.5563889.

[33] A. Thakur and M. S. Goraya, "A taxonomic survey on load balancing in cloud," *J. Netw. Comput. Appl.*, vol. 98, pp. 43–57, Nov. 2017, doi: 10.1016/j.jnca.2017.08.020.

[34] J. K. 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. 1916–1926, 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, doi: 10.17485/ijst/2017/v10i11/110107.

[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, doi: 10.1109/ICCCNT.2017.8204070.

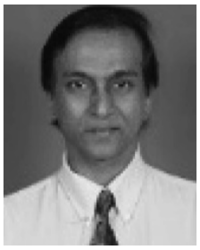
[37] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *J. King Saud Univ.–Comput. Inf. Sci.*, Oct. 2020, doi: 10.1016/j.jksuci.2020.10.016.

[38] Z. A. Almusaylim and NZ. Jhanjhi, "Comprehensive review: Privacy protection of user in location-aware services of mobile cloud computing," *Wireless Pers. Commun.*, vol. 111, pp. 541–564, Oct. 2019.

[39] Z. A. Almusaylim, N. Zaman, and L. T. Jung, "Proposing a data privacy aware protocol for roadside accident video reporting service using 5G in vehicular cloud networks environment," in *Proc. 4th Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Aug. 2018, pp. 1–5.

DALIA ABDULKAREEM SHAFIQ received the bachelor's degree in computer science. She is currently pursuing the master's degree in cloud computing with Taylor's University. She has great research and design skill. She is publishing with conferences and journal, and she has vast coaching experience as well.

NOOR ZAMAN JHANJHI is currently working as an Associate Professor with the School of Computer Science and Engineering, Taylor's University, Malaysia. Besides of teaching, research, and administrative responsibilities, he is heading the Cybersecurity Research Cluster, where he is supervising a great number of Postgraduate students mainly in the cybersecurity for data science. Cybersecurity research cluster has extensive research collaboration globally with several professional. A great number of postgraduate students are graduated under his supervision. He has high indexed publications in WoS/ISI/SCI/Scopus, and his collective research Impact factor is more than 150 points. He has international patents on his account, edited/authored more than 22 research books published by world-class publishers. He has a great experience of supervising and co-supervising postgraduate students, an ample number of Ph.D. and master's students graduated under his supervision. He is an external Ph.D./Master thesis examiner/evaluator for several universities globally. He has completed more than 21 international funded research grants successfully. He has served as a Keynote Speaker for several international conferences, presented several Webinars worldwide, chaired international conference sessions. He has achieved consecutively higher performance award. He is an Associate Editor and an Editorial Assistant Board for several reputable journals, including IEEE Access Journal, a PC member for several IEEE conferences worldwide, and a guest editor for the reputed indexed journals. He is an Active Reviewer of a series of Q1 journals, has been awarded globally as a top 1% reviewer by Publons (Web of Science).



AZWEEN ABDULLAH (Senior Member, IEEE) is currently working with Taylor's University, Malaysia. He is a Professional Development Alumni of Stanford University and MIT. His work experience includes 30 years as an academic in institutions of higher learning and as the Director of research and academic affairs at two institutions of higher learning, the Vice President of educational consultancy services, 15 years in commercial companies as a Software Engineer, a Systems Analyst, and a Computer Software Developer, and IT/MIS Consultancy and Training.

MOHAMMED A. ALZAIN received the bachelor's degree in computer science from King Abdulaziz University, Saudi Arabia, in 2004, the master's degree in information technology from La Trobe University, Melbourne, VIC, Australia, in 2010, and the Ph.D. degree from the Department of Computer Science and Computer Engineering, La Trobe University, in September 2014. He is currently an Associate Professor with the College of Computers and Information Technology, Taif University, Saudi Arabia. His research interests include cloud computing security, multimedia security, image encryption, steganography, and medical image processing.