Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Improving fog resource utilization with a dynamic round-robin load balancing approach

Entisar S. Alkayal [1,*], Nesreen M. Alharbi [2], Reem Alwashmi [2], Waleed Ali [1]

[1]*Information Technology Department, Faculty of Computing and Information Technology-Rabigh, King Abdulaziz University, Jeddah, Saudi Arabia*
[2]*Computer Science Department, Faculty of Computing and Information Technology-Rabigh, King Abdulaziz University, Jeddah, Saudi Arabia*

ARTICLE INFO

Article history:
Received 10 June 2024
Received in revised form
5 October 2024
Accepted 13 October 2024

Keywords:
Load balancing
Fog computing
Round-robin
Latency reduction
Resource optimization

ABSTRACT

In fog computing, load balancing is an important research problem. It focuses on efficiently assigning tasks to fog nodes and minimizing delay in real-time applications. The traditional round-robin algorithm assigns tasks in a rotating manner among fog nodes, but it can send tasks to the cloud too early, leading to increased delays. To solve this problem, this paper introduces an improved round-robin algorithm that takes a dynamic approach to balancing the use of fog resources. The new model aims to improve load balancing in fog computing by distributing tasks more evenly among fog nodes, reducing dependence on cloud computing, and making better use of fog resources. The improved algorithm helps fog computing systems run more efficiently, reduces delays in real-time applications, and lowers the costs associated with cloud use. The results show that the proposed load balancing algorithm is key to optimizing fog resource use, improving system efficiency, and reducing task completion times in distributed computing systems.

© 2024 The Authors. Published by IASE. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

The load balancing issue is an important area of study in fog computing (Sulimani et al., 2024). Resource allocation involves effectively mapping tasks to fog nodes to ensure optimal use of fog resources, aiming to minimize bandwidth usage, service request delays, and support real-time applications. Occasionally, fog resources within clusters are not fully utilized. Tasks that cannot be assigned to any available resource are sent to the cloud, potentially leading to increased latency. To improve the efficiency of fog node allocation, a robust allocation method to distribute the load among fog nodes is necessary (Ogundoyin and Kamil, 2021).

There are several load balancing algorithms that can be applied in the system to manage the resource allocation problem. Some of them are static, such as the round-robin method (Ali and Alubady, 2023). Other methods, such as the adaptive approach (Wang and Lu, 2022), are dynamic. It is essential to apply balancing algorithms that utilize the fog resources to the maximum level to decrease the number of jobs forwarded to the cloud (Ali and Alubady, 2023).

This paper focuses on improving the round-robin algorithm since the traditional round-robin algorithm, or the static round-robin (SRR) algorithm, is the most used technique because of its simplicity. It distributes the work between the network servers, where the jobs are assigned to the fog nodes in a cyclic manner before the cycle repeats. For example, in a fog system with (n) fog servers, the SRR sends the first job to the first node and the second job to the second fog node. When job n arrives, the algorithm will assign it to fog node n. The following job (n+1) will be assigned to fog node 1, and the algorithm restarts the order from the beginning (Hidayat et al., 2019). At any time, if the server in order reaches its full capacity, the next job will be routed to the cloud regardless of the load capacity of the other fog nodes, as shown in Fig. 1.

The greatest advantage of SRR is that it is easy to implement. However, if the load arriving at the cloud is different at each iteration, using SRR would not be efficient in balancing the load (Hidayat et al., 2019). To address SRR limitations, this paper proposes a

* Corresponding Author.
Email Address: ealkayyal@kau.edu.sa (E. S. Alkayal)
https://doi.org/10.21833/ijaas.2024.10.022
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-6617-1051
2313-626X/© 2024 The Authors. Published by IASE.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

dynamic approach involving the round-robin balancer (DRR) algorithm to increase the overall response time. This research questions whether applying the proposed DRR algorithm can increase fog resource utilization and decrease the number of jobs rerouted to the cloud to reduce the overall response time of the system.

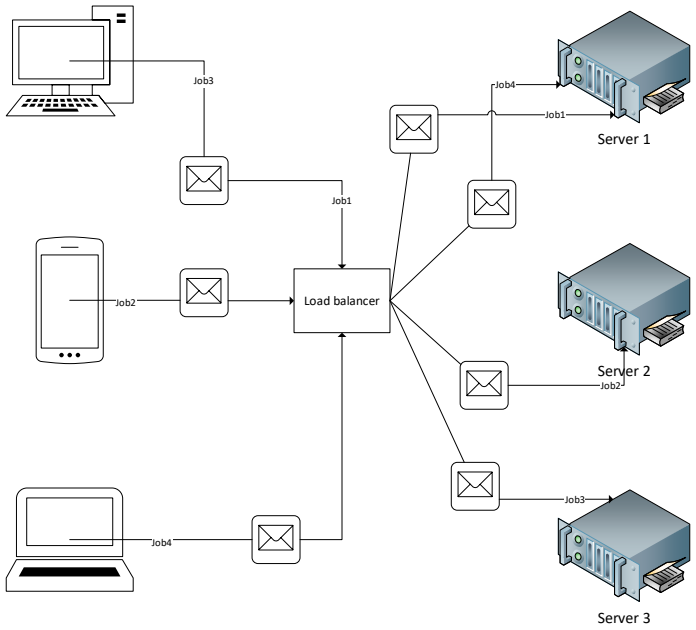

Fig. 1: Load balancing method based on the static round-robin algorithm

The main contributions of this paper are summarized as follows:

- A novel dynamic round-robin load balancing (DRR) algorithm is proposed to improve the utilization of fog resources.
- A load balancing manager is developed in the proposed DRR algorithm to effectively distribute the load over the available fog nodes and reduce the need to access cloud computing.
- The proposed DRR can play a key role in reducing the response time and, therefore, reducing the cost of using clouds. This is because the proposed DRR method can effectively schedule jobs between fog resources to improve the efficiency of the fog resources and reduce dependency on the cloud.

The remaining sections of the paper are organized as follows. Section 2 reviews the previous load balancing methods used in fog resources and background information about cloud computing, fog computing, and load balancing. Section 3 presents the architecture of the proposed dynamic round-robin load balance method. Section 4 explains the experimental setup and discusses the results of the proposed algorithm. Section 5 summarizes the conclusions and offers future directions.

2. Literature review

Cloud computing refers to the use of remote resources to save and manipulate data via the internet. It can be divided into two components: the platform and the applications. The cloud platform includes cloud operating systems and hardware or virtual servers that provide services to several types of cloud applications. The platform distributes the work from the cloud applications among the servers dynamically. On the other hand, cloud applications include applications hosted by cloud servers and are available to any user who has access to the cloud.

Cloud computing includes several computer resources that accept different workloads. With the support of the cloud's virtual and physical nodes, this workload is quickly deployed and scaled out within the cloud. The infrastructure of the cloud increases the efficient usage of hardware and software resources. This can be achieved by grouping the system entities into one regardless of the location of each entity. Overall, cloud computing is a real-time, load balancing, virtualized system that reduces management complexity and increases system responsiveness (Afzal and Kavitha, 2019; Prakash et al., 2017).

One of the advantages of cloud computing is resource scalability, where the users can add or remove resources to their cloud based on their needs. Another benefit is that using cloud resources helps the user decrease the cost needed for upfront infrastructure investments. In addition, the users can rely on the availability of cloud resources such as data centers since the services provided by the cloud are always redundant and packed up, which increases the cloud's reliability and availability. Moreover, an important advantage of cloud computing is the user's ability to access the cloud system remotely and at any time, allowing the users of the cloud to collaborate and share resources between them (Ameen and Begum, 2022; Prakash et al., 2017).

On the other hand, cloud computing has several disadvantages, including the need for internet connectivity to access the cloud. Additionally, sharing resources between several users may threaten the security and privacy of the data if the service provider does not provide a good security system on the cloud. Another disadvantage is that the cloud infrastructure is under the control of the service provider, which decreases the degree to which the user controls his or her resources. Another major problem facing the cloud is its performance and response time. As the cloud infrastructure and the number of users who execute the cloud resources increase, the cloud may take longer to serve its users. This results in decreased cloud performance and an increased time needed to finish jobs from the users (Ameen and Begum, 2022; Prakash et al., 2017).

The cloud infrastructure increases the cost since the users need to execute their jobs directly on the cloud without benefiting from their local resources if needed. One of the challenges faced in cloud computing is that the central server is responsible for processing the data in cloud computing. This consumes time when the data must be moved from one node to another until it reaches the central server and then moves back to the sending node (Ameen and Begum, 2022; Prakash et al., 2017). Therefore, fog computing was introduced to address the limitations of cloud computing, as discussed in the next section.

Fog computing is an extended paradigm of the cloud infrastructure. Fog resources are distributed among the network and assigned at its edge closer to the users. They allow the users to run their jobs using the closest resource instead of migrating the jobs to the cloud. This will decrease the time needed to process the data and increase the overall cloud performance (Harish et al., 2019). Fig. 2 illustrates the layers of fog computing and the relationship between them and the cloud computing infrastructure.

Fog computing expands the computing in the cloud to the network edge. The jobs are processed inside the node if no higher computing service is required because the fog infrastructure is limited. Furthermore, tasks can be executed partially between the node and the central server if needed. This reduces the amount of time needed to transfer the task from/to a node to the central server. Typically, fog computing is useful for real-time applications since it helps reduce the latency time and improves the response time (Harish et al., 2019).

Even though fog computing is an extension of cloud computing, some issues must be addressed in fog computing. When enormous numbers of jobs come from its users, the fog must control which fog resource to assign to each user. Assigning the users' jobs to the fog resources must be balanced; therefore, fog computing needs a good load balancing algorithm to ensure that the performance of fog computing for every node in the system is effectively balanced. The fog must be able to efficiently balance the workload and schedule the resources to take advantage of the fog computing benefits. These benefits include decreasing the latency time and the cost of using resources and increasing the reliability of the overall cloud system (Hidayat et al., 2019).

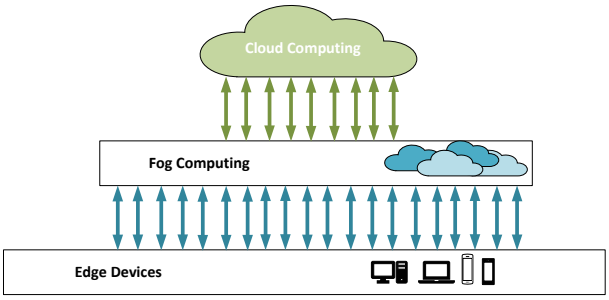

Fig. 2: Fog computing architecture

Load balancing algorithms can be static or dynamic. Dynamic load balancing algorithms distribute the tasks while considering the current load of each server in the cloud, whereas static load balancing algorithms do not take similar considerations when distributing the incoming tasks. The load balancer can be chosen based on the system needs. If the system is homogeneous or heterogeneous, applying the dynamic approach will be more suitable. However, using the dynamic approach increases the system overhead. If the degree of overhead is significant to the system, the use of static approaches would be more suitable than the use of dynamic approaches (Afzal and Kavitha, 2019).

The round-robin algorithm in the load balancing technique is a classic algorithm that has been widely used because of its simplicity. The round-robin algorithm has an essential role in distributing the client requests among the cloud servers one at a time following an ordered list starting from the beginning of the list (Wang et al., 2016). Once a request is assigned to the last server in the list, the process of setting client requests starts again from the first node. The main advantage of the round-robin algorithm is that it is easier to implement than other load balancing algorithms. However, when there are different loading requests to be distributed into the server, the loads cannot be balanced effectively (Ghosh and Banerjee, 2018). The algorithm is based on the priority of the different processing times by measuring the time parameter. Different time parameters are associated with the round-robin algorithm, such as the burst time and time quantization (Xu et al., 2016). The burst time is the time required to finish executing a service request, whereas the time quantum is the time assigned to a service so that it can access a virtual machine.

Researchers have identified a set of disadvantages with respect to the static round-robin algorithm (Choudhary and Kothari, 2018). Some of these disadvantages are the increase in the system waiting time, response time, and context switch number, as well as the decrease in the system

throughput in some stances. In addition, one of the major flows of the traditional round-robin algorithm is the nature of the quantum of time.

There are many popular load balancing algorithms that can be applied to improve cloud computing performance (Shafiq et al., 2022). These algorithms can be classified into three types according to the operation environment on which they depend; hence, they can be static, dynamic, and nature-inspired algorithms. The process of load balancing algorithms in a static environment depends on previous knowledge of the capabilities, features, and state of the system. For example, the system information includes the memory, storage capacity, and processing needed to specify the load of the system. On the other hand, a dynamic change in the load during runtime has not been considered in the static load balancing algorithms. Thus, a major drawback of the static algorithm is its intolerance to sudden changes in load (Mishra et al., 2020).

The dynamic algorithms used for load balancing are more efficient and adaptable than the static algorithms. Unlike static algorithms, dynamic algorithms are more flexible for distributed cloud systems since they do not require any prior knowledge (Shafiq et al., 2022). Haryani and Jagli (2014) and Sharma et al. (2022) reported that, compared with static algorithms, dynamic algorithms have greater runtime complexity and eliminate the overhead from the previous state storage of the system.

Recently, job scheduling in a cloud computing environment has been discussed as a critical research issue. Accordingly, the system's performance needs an efficient task-scheduling algorithm to increase the network capacity and reduce the response time. Moreover, many load balancing algorithms that enhance cloud application performance have been introduced by Ameen and Begum (2022), who presented a new Median-Average round-robin (MARR) scheduling algorithm, and they confirmed that when the time quantum has been adjusted dynamically, it performs better than the static round-robin. With the proposed algorithm, the average turnaround (ATT) and waiting time (AWT) achieved the best results on all the performance metrics; nevertheless, the number of context switches increased with the algorithm.

Furthermore, divisible weighted and dynamic load balancing approaches have been used in other studies, such as Sharma et al. (2022), due to their effectiveness in reducing response times and achieving load balancing without causing delays. Devi and Uthariaraj (2016) introduced a model that allocates incoming tasks within a virtual machine (VM) environment, incorporating scheduling, load balancing, resource monitoring, and task management algorithms. In this model, the scheduler assigns incoming tasks to the VM with the fewest jobs at the time of arrival. The load balancing algorithm then redistributes tasks among available VMs to maintain an even workload distribution. This paper proposed an improved weighted round-robin algorithm, called IWRR, as a load balancing method. The effectiveness of IWRR was assessed using metrics such as response time, the number of migrated tasks, cumulative idle time for all tasks, and the total number of delayed tasks. IWRR was compared to RR and WRR algorithms across various job lengths, demonstrating superior performance in terms of response time, a key quality of service (QoS) metric.

Noman and Jasim (2021) analyzed the performance of static and dynamic load balancing, emphasizing network throughput. They examined static techniques, including random and weighted methods, finding minimal changes with round-robin and weighted methods. However, compared to the random technique, the weighted round-robin method achieved a 6% improvement. In dynamic load balancing, comparing the least connection-based and least bandwidth-based techniques showed less than a 1% difference in average network throughput. Overall, the dynamic methods outperformed static ones. When compared to static round-robin, weighted round-robin, and random methods, connection-based techniques improved throughput by up to 2.2%, 2%, and 7%, respectively. Similarly, the least bandwidth-based methods achieved throughput improvements of up to 3.3%, 2.5%, and 8% over these static methods.

In a study by Waghmode and Patil (2023), dynamic load balancing in cloud computing was presented with a focus on enhancing network performance. The weighted round-robin algorithm is used to achieve load balancing and to allocate jobs effectively to maximize processing speed. Pakhrudin et al. (2023) focused on enhancing cloud service analysis via the round-robin algorithm for task placement for Internet of Things (IoT) services. The round-robin technique is optimized by adjusting the parameter values to provide high accuracy and low cost. The results demonstrated improvements in response times and cost efficiencies for virtual machines.

The aim of the experiments conducted by Sinha and Sinha (2020) was to improve the weighted round-robin (WRR) load balancing algorithm to increase the efficiency and effectiveness of resource utilization in cloud computing. Both static and dynamic load balancing algorithms have been utilized to allocate VMs based on their processing capacity and to dynamically adjust the load on each VM at runtime. This approach involves considering the capacity of each VM to determine the optimal allocation of jobs to the appropriate virtual machines. Researchers have concentrated on optimizing the performance of virtual machines by selecting an algorithm that prioritizes job duration, resource capacity, and the interdependency of multiple tasks. The enhanced WRR load balancing algorithm overcomes the drawback of the RR load balancing algorithm, which fails to consider the number of user requests when assigning the appropriate virtual machine to incoming requests. Instead, it simply makes decisions based on rotation.

Garcia-Carballeira et al. (2021) implemented the shortest queue algorithm based on randomization techniques, and static local balancing was implemented based on a round-robin policy. The results of this new version demonstrated an enhanced performance compared with the traditional solution across all scenarios, even in systems operating at 99% capacity. A key benefit of this approach is the reduced likelihood of selecting a server that was recently chosen. Furthermore, simulations were conducted to validate the theoretical approximation proposed in the study. These simulations revealed that the probability of selecting an empty server is 60% higher for high service rates.

These works provide good results in balancing loads in cloud computing paradigms. This paper focuses on disturbing the load among fog resources and reducing the need to use cloud computing resources, especially for real-time applications that need lower response times with shorter latencies.

3. The proposed dynamic round-robin load balanced algorithm

In fog computing, each cluster contains fog resources that handle computational requests from edge devices within the cluster. If a fog resource has sufficient capacity, it processes tasks locally within the fog nodes. Otherwise, tasks are forwarded to cloud computing. At times, fog resources may remain idle if there are no computational requests from edge devices during specific time slots. To address this issue, the proposed algorithm aims to balance the load among fog resources and minimize the number of tasks sent to the cloud, thereby reducing processing time. The architecture of the proposed system consists of four layers, as illustrated in Fig. 3.

- Edge device layer: This layer includes N users connected to the fog resource (FR) in a cluster; these users send jobs to the FR for processing at each slot of time.
- Fog resource (FR): In this layer, there are M clusters in the setup, and each cluster has FR. If FR receives jobs from connected users (N), it processes the request based on several criteria. If the load at the FR exceeds its capacity, then it sends jobs to the load balancer manager to schedule with the nearby FR or cloud server.
- Load balancer manager (LBM): In this layer, the LBM has a list of FRs and obtains the status details of the FR at every slot. Its main goal is to pass jobs from FRs that have more work to those with less workload. In the worst case, if the LBM cannot find any FR that has a lower load, the LBM will pass those jobs to the cloud server, which has unlimited resources. Then, the cloud server performs the remaining task within that slot. There will be M FRs managed by the LBM; by trying to find a suitable FR to avoid an unbalanced job distribution before the jobs are sent to the cloud, the LBM will manage to increase the FR utilization, decrease the cloud utilization, and improve the completion time of the jobs.
- Cloud resources: In this layer, the cloud servers have unlimited resources to process the jobs.

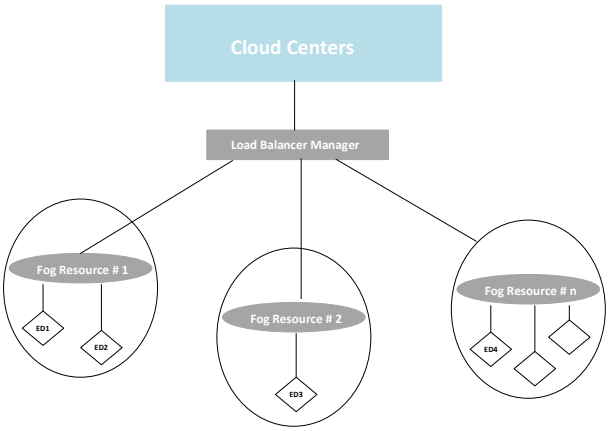

Fig. 3: Architecture of the proposed DRR load balancer

The proposed dynamic round-robin load balancing algorithm includes the following steps:

1. Edge devices submit tasks to the cluster heads of the fog resources.
2. Jobs sent by the edge devices to the fog resources are accepted by the fog resources.
3. If the number of jobs exceeds the capacity of the fog resource, then excess jobs are sent to the load balancer manager to schedule them to other fog clusters.
4. If the number of jobs submitted is less than the capacity of the fog resource, then the fog resource informs the load balancer manager about its extra capacity to receive jobs from other clusters that have more jobs.
5. The manager balances the load on the fog resources at all the clusters, while any jobs left out are sent to the cloud for execution.
6. At the end of each slot, no job is left without completion.

The DRR algorithm steps are shown in Algorithm 1, which is based on the dynamic job distribution method in a fog computing environment and is designed to optimize the utilization of resources across fog clusters. Initially, it sets up the necessary data structures: S for tracking the simulation slot number, JOB_LIST, and DIFFER_JOB_LIST for holding jobs, and CLUSTER_LIST for listing available cluster fog resources. The algorithm operates within a loop, constrained by a maximum simulation time (MAX_SIM_TIME). In each simulation slot, JOB_LIST is populated with jobs submitted by edge devices. Each job in JOB_LIST is then processed. If the designated cluster fog (ClusterFog) has the capacity, the job is executed immediately. Jobs that cannot be processed due to capacity constraints are moved to DIFFER_JOB_LIST for subsequent handling.

In processing deferred jobs from DIFFER_JOB_LIST, the algorithm checks each cluster in CLUSTER_LIST for available capacity. If a cluster

can take on a job, it is assigned and removed from the deferred list. This process continues until either a cluster is found for each deferred job or no suitable cluster is available. In the latter case, the job is sent to the cloud for processing (send_job_toCloud). After all the jobs are processed, the lists are reset, and the simulation moves to the next slot (S ->S+1). This loop of job assignment, execution, and reassignment (if needed) continues until the simulation reaches its time limit. The algorithm efficiently manages the fog computing resource allocations, dynamically adjusts to varying workloads, and ensures the optimal use of the fog and cloud resources.

Algorithm 1: Algorithm of the proposed DRR

```
S = 1
initialize JOB_LIST and DIFFER_JOB_LIST to empty job list
initialize CLUSTER_LIST with list of clusters_fog_resources
while Time(S) < MAX_SIM_TIME
JOB_LIST ->list of jobs, submitted by edge devices
//received and added to JOB_LIST, at slot S
for each JOB in LIST
if ClusterFog (JOB) has capacity to process
ClusterFog (JOB) execute the task/job
Else
JOB needs co-ordination and added to DIFFER_JOB_LIST.
for_end
for each JOB in DIFFER_JOB_LIST //deferred jobs for loop
Boolean NO_CLUSTER_FREE = true
for ClusterFog in CLUSTER_LIST //cluster for_loop
if ClusterFog has capacity // has idle time
ClusterFog (JOB) execute JOB
set NO_CLUSTER_FREE = false
break (exit cluster for_loop)
for_end
if (true = NO_CLUSTER_FREE) send_job_toCloud (JOB)
for_end
initialize JOB_LIST and DIFFER_JOB_LIST to empty
S->S+1 // set next slot no
end_repeat
```

4. Experimental implementation and evaluation

4.1. Simulation setup

To implement and evaluate the proposed load balancing algorithm in fog computing, a simulation-controlled and flexible environment was developed in the Java language. The components of the fog and cloud computing elements, such as the edge devices, fog nodes, and cloud servers, were simulated. The algorithm was then programmed by defining the logic for task submission, job acceptance, capacity management, manager intervention for load balancing, and cloud computing resources. The proposed work was then configured, and the parameters for evaluation, such as the number of edge devices, the capacity of the fog resources, the network latency, and the processing power, were set. These parameters should reflect realistic scenarios for accurate testing.

4.2. Evaluation metrics

Once the environment and algorithm are implemented, performance metrics such as task completion time, resource utilization efficiency, and load distribution are evaluated. To demonstrate the effectiveness of the proposed algorithm, performance comparisons were conducted with and without its application. The following steps were used to assess the proposed algorithm:

- A simulated environment with edge devices, fog resources (clusters), a manager, and a cloud is set up. This involves deploying and configuring the necessary hardware and software components.
- Generate a set of test tasks or jobs that represent different workloads. These tasks should vary in size and complexity to simulate real-world scenarios.
- Implement the algorithm steps in your system's codebase based on the description provided. It is important to ensure that the code accurately represents the logic and flow of the algorithm.
- Test cases that cover different scenarios and edge cases are created.
- Execute the test cases by running the algorithm implementation on the test environment. Then, the results, including job assignments, load balancing decisions, and job completions, are monitored and recorded.
- The test results are analyzed and compared against the expected outcomes. The algorithm behaves as intended and meets the desired objectives, such as load balancing, efficient resource utilization, and job completion within time slots.

In this work, three metrics, namely, the average completion time of the job, the fog resource CPU utilization of the fog resource, and the cloud utilization, are used to evaluate the performance of the proposed algorithm. These metrics are described as follows.

Fog resource utilization (FRU): In the proposed work, only the CPU utilization, which is based on computational tasks, is used. Therefore, the CPU utilization is computed via Eq. 1.

$$FRU\ (fs) = \frac{UsedC(fs)}{TotalC(fs)} * \ 100 \quad (1)$$

where, $UsedC$ is the used CPU for all tasks executed in server fs, and $TotalC(fs)$ is the total CPU of fog fs.

Cloud utilization (CU): This paper focuses on the cloud utilization of computing resources. Therefore, the cloud utilization is computed via Eq. 2.

$$CU\ (fs) = \frac{UsedC(fs)}{TotalC(fs)} * 100 \quad (2)$$

where, $UsedC$ is the used CPU for all the tasks executed in server fs, and $TotalC(fs)$ is the total CPU of fog fs.

Average completed time (ACT): The average completed time of a job is measured by computing the difference between the time of submitting the job

to the system and the time of finishing the execution of the job, as shown in Eq. 3.

$$ACT\ (fs) = FinExct(t) - Subt(t) \quad (3)$$

where, $FinExcT(t)$ denotes the finish time of the execution of job t, and $SubT(t)$ denotes the submission time of job t.

4.3. Performance results of the proposed method

To assess the proposed algorithm, experiments were conducted using 3, 4, and 5 clusters with varying numbers of edge devices to measure its effectiveness. The results are presented in Tables 1 and 2.

Table 1 displays the average completion time for assigned tasks, both with and without applying the DDR algorithm, across different scenarios. The findings indicate an improvement in completion time when using the DDR algorithm. Table 2 presents the CPU utilization results for 3, 4, and 5 clusters, comparing scenarios with and without DDR. The results demonstrate an increase in CPU utilization when the DDR algorithm is applied.

Table 1: The average time of completed tasks (seconds)

Clusters	Applying DRR	Edge devices	Number-of-tasks	Average time
3	Yes	15	2456	7.8257
3	No	15	2503	9.1014
4	Yes	22	3621	8.2239
4	No	22	3603	9.7643
5	Yes	30	5007	8.7926
5	No	30	4947	10.7558

Table 2: CPU utilization results with DRR and without DRR in 3, 4, and 5 clusters

Clusters	Applying DRR	CPU utilization 3 clusters	CPU utilization 4 clusters	CPU utilization 5 clusters
FR1	True	75.24	85.1297	95.9081
FR1	False	63.473	62.275	63.772
FR2	True	79.341	83.5329	90.01
FR2	False	74.351	75.648	73.253
FR3	True	84.1317	86.3273	92.415
FR3	False	85.7285	82.834	82.135
FR4	-	-	91.2175	92.415
FR4	-	-	89.121	89.021
FR5	-	-	-	95.409
FR5	-	-	-	92.814

The CPU utilization comparison in different cluster configurations, referred to as "cluster FRs," with varying numbers of edge devices is shown in Fig. 4. It compares scenarios with and without the proposed load balancing manager. Unlike Tables 1 and 2, which showed that the load balancing manager reduces resource utilization or task completion times, this chart indicates that CPU utilization is greater with the proposed load balancing manager across all the cluster configurations. This means that the proposed load balancing manager can distribute tasks in a way that utilizes CPU resources more completely, which could be an indication of improved efficiency or a more evenly distributed workload among the available CPUs. However, the increase in utilization is not drastic, indicating a balanced approach to load management.

Table 3 shows the results of the comparison of cloud utilization in scenarios with and without applying the proposed load balancing manager across different numbers of clusters. As shown in Fig. 5, the proposed load balancing manager effectively reduces cloud utilization across all the cluster configurations. The trend suggests that, as the number of clusters and edge devices increases, the cloud utilization also increases when there is no load balancing manager. However, the presence of the proposed load balancing manager consistently lowers the utilization, indicating a more efficient distribution of computing tasks across the fog nodes, preventing an overreliance on the cloud.

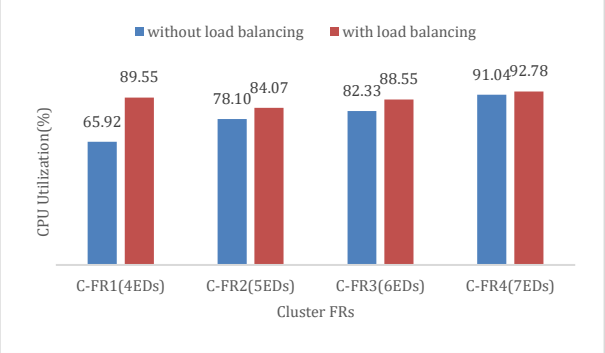

Fig. 4: Comparison of the CPU utilization results before and after applying the proposed DRR

Table 3: Cloud utilization with DRR and without DRR with 2, 3, or 4 clusters

Clusters	Applying DRR	Cloud computing
FR3	T	6.3999
FR3	F	26.3
FR4	T	15.19
FR4	F	49.8
FR5	T	33.6
FR5	F	92.89

Fig. 6 compares the average time in seconds to complete the tasks in a computing environment with different numbers of clusters and edge devices, specifically, the effect of using the proposed load balancing manager. The chart shows that in all the

cluster configurations, the presence of the proposed load balancing manager reduces the average time required to complete the tasks. This indicates that the proposed load balancing manager contributes to increasing efficiency and reducing the completion time of the tasks as the number of clusters and edge devices increases. The trend shows that the time savings are more significant as the system scales up, highlighting the effectiveness of load balancing in larger and more complex environments.

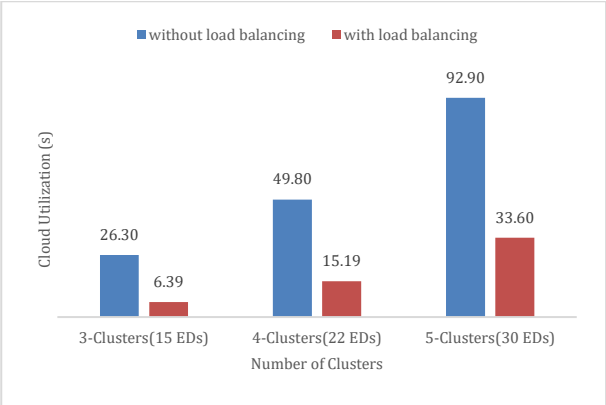

Fig. 5: Comparison of the cloud utilization results before and after applying the proposed DRR

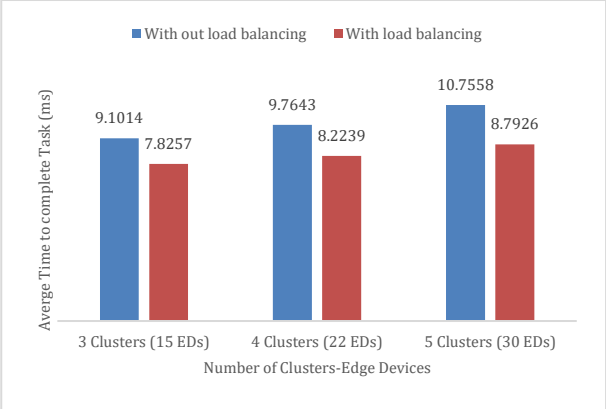

Fig. 6: Comparison of the average time in seconds after applying the proposed DRR

4.4. Discussion of results

The analysis of CPU utilization, cloud utilization, and average task completion times across different cluster configurations, with and without the proposed load balancing manager, provides valuable insights into the effectiveness and impact of the DRR load balancing approach.

The comparison of CPU utilization across various cluster configurations demonstrates improved efficiency and a more balanced workload distribution among available CPUs when using the load balancing manager. Regarding cloud utilization, the analysis shows that the DRR load balancing manager effectively reduces cloud usage in all cluster configurations. The trend indicates that without a load balancing manager, cloud utilization increases as the number of clusters and edge devices grows. In contrast, the presence of a load balancing manager consistently lowers cloud usage by distributing computing tasks more efficiently across fog nodes, thus reducing dependency on the cloud.

The analysis of average task completion times further highlights the benefits of load balancing. In all configurations, the load balancing manager decreases the average time needed to complete tasks. This indicates that load balancing enhances overall system efficiency, particularly in larger and more complex environments where time savings are more pronounced.

This model has several advantages and some limitations. Most of the advantages are described as follows:

- Improving CPU utilization: By ensuring an effective distribution of the incoming work along the available CPU resources, this algorithm manages optimal utilization of the available CPU resources, which reduces the possibility of having some overloaded CPUs and other idle resources at the same time. Similarly, (Devi and Uthariaraj, 2016) highlighted the importance of applying the IWRR-optimized scheduling algorithm to improve the efficiency and utilization of fog and cloud resources.
- By reducing cloud utilization by approximately 70%, the algorithm first ensures that any incoming job will be assigned to the fog resources. For any reason, if the next candidate's fog resource is overloaded, the algorithm first searches within the other fog resources for any available resource to assign the job to instead of assigning the job to the cloud. This reduces the number of jobs sent to the cloud and minimizes cloud utilization. This finding is consistent with the results of Noman and Jasim (2021), who reported that the overall performance of dynamic and static techniques was better than that of static techniques. The DDR results are also consistent with the findings of Sinha and Sinha (2020) and Garcia-Carballeira et al. (2021), which show an enhancement of the improved round-robin techniques in terms of the efficiency and effectiveness of resource utilization in cloud computing.
- By increasing the average time of completing the tasks by approximately 15% for any incoming job, the algorithm ensures assigning the job to the fog resources that are available instead of assigning it to the next inline resource, regardless of whether this resource is overloaded or not.

In summary, the results of applying DDR in a fog environment support the advantages of improving the load balancing and scheduling techniques, which constitute the core of the DDR algorithm. The proposed load balancing algorithm plays a crucial role in optimizing fog resource utilization, improving efficiency, and reducing completion times in distributed computing environments. Compared with the traditional SRR algorithm, one of the limitations is that the new approach may suffer from an additional overhead needed by the load balancing manager to add extra computations to decide which

resource it will assign the task to. This may also increase the overall complexity of the algorithm since the load manager must monitor all available resources at every assigning cycle to ensure even load balancing.

5. Conclusion and future work

To improve the performance of the traditional static round-robin algorithm, this paper proposes a dynamic load balancing approach involving the round-robin balancer algorithm to increase the utilization of fog resources. The proposed algorithm develops a load balancing manager that can effectively distribute the load over available fog nodes to reduce the need to access cloud computing. The simulation results demonstrated that the proposed load balancing manager of DRR contributed to enhancing the utilization of fog resources in terms of CPU utilization, cloud utilization, and the average time to complete tasks in the different cluster configurations. By effectively distributing tasks among available resources, load balancing enables a more balanced workload and prevents overreliance on the cloud, resulting in improved performance and scalability. To further improve the performance of the proposed algorithm, heuristic and evolutionary algorithms can be applied to schedule and optimize tasks/jobs across the fog resources.

The model can be improved by applying machine learning to help determine the initialization parameters on the basis of the previous data, which will improve and optimize the results.

Acknowledgment

This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (GPIP: 528-865-2024). The authors gratefully acknowledge the DSR for their technical and financial support.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Afzal S and Kavitha G (2019). Load balancing in cloud computing–A hierarchical taxonomical classification. Journal of Cloud Computing, 8: 22. https://doi.org/10.1186/s13677-019-0146-7

Ali S and Alubady R (2023). RWRR: Remind weighted rounding robin for load balancing in fog computing. In the 7th International Symposium on Innovative Approaches in Smart Technologies, IEEE, Istanbul, Turkey: 1-7. https://doi.org/10.1109/ISAS60782.2023.10391499

Ameen JN and Begum SJ (2022). Evolutionary algorithm based adaptive load balancing (EA-ALB) in cloud computing framework. Intelligent Automation and Soft Computing, 34(2): 1281-1294. https://doi.org/10.32604/iasc.2022.025137

Choudhary R and Kothari DA (2018). A novel technique for load balancing in cloud computing environment. International Journal of Software and Hardware Research in Engineering, 6(6): 1-5.

Devi DC and Uthariaraj VR (2016). Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. The Scientific World Journal, 2016: 3896065. https://doi.org/10.1155/2016/3896065 **PMid:26955656 PMCid:PMC4756214**

Garcia-Carballeira F, Calderon A, and Carretero J (2021). Enhancing the power of two choices load balancing algorithm using round robin policy. Cluster Computing, 24(2): 611-624. https://doi.org/10.1007/s10586-020-03139-6

Ghosh S and Banerjee C (2018). Dynamic time quantum priority based round robin for load balancing in cloud environment. In the 4th International Conference on Research in Computational Intelligence and Communication Networks, IEEE, Kolkata, India: 33-37. https://doi.org/10.1109/ICRCICN.2018.8718694 **PMCid:PMC6191404**

Harish G, Nagaraju S, Harish B, and Shaik M (2019). A review on fog computing and its applications. International Journal of Innovative Technology and Exploring Engineering, 8(6C2): 2278-3075.

Haryani N and Jagli D (2014). Dynamic method for load balancing in cloud computing. IOSR Journal of Computer Engineering (IOSR-JCE), 16(4): 23-28. https://doi.org/10.9790/0661-16442328

Hidayat T, Azzery Y, and Mahardiko R (2019). Load balancing network by using round robin algorithm: A systematic literature review. Jurnal Online Informatika, 4(2): 85-89. https://doi.org/10.15575/join.v4i2.446

Mishra SK, Sahoo B, and Parida PP (2020). Load balancing in cloud computing: A big picture. Journal of King Saud University-Computer and Information Sciences, 32(2): 149-158. https://doi.org/10.1016/j.jksuci.2018.01.003

Noman HM and Jasim MN (2021). A comparative performance analysis for static and dynamic load balancing techniques in software defined network environment. Journal of Physics: Conference Series, 1773: 012010. https://doi.org/10.1088/1742-6596/1773/1/012010

Ogundoyin SO and Kamil IA (2021). Optimization techniques and applications in fog computing: An exhaustive survey. Swarm and Evolutionary Computation, 66: 100937. https://doi.org/10.1016/j.swevo.2021.100937

Pakhrudin NSM, Kassim M, and Idris A (2023). Cloud service analysis using round-robin algorithm for quality-of-service aware task placement for Internet of Things services. International Journal of Electrical and Computer Engineering, 13(3): 3464-3473. https://doi.org/10.11591/ijece.v13i3.pp3464-3473

Prakash P, Darshaun KG, Yaazhlene P, Ganesh MV, and Vasudha B (2017). Fog computing: Issues, challenges and future directions. International Journal of Electrical and Computer Engineering, 7(6): 3669-3673. https://doi.org/10.11591/ijece.v7i6.pp3669-3673

Shafiq DA, Jhanjhi NZ, and Abdullah A (2022). Load balancing techniques in cloud computing environment: A review. Journal of King Saud University-Computer and Information Sciences, 34(7): 3910-3933. https://doi.org/10.1016/j.jksuci.2021.02.007

Sharma C, Sharma S, Kautish S, Alsallami SA, Khalil EM, and Mohamed AW (2022). A new median-average round robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time. Alexandria Engineering Journal,

61(12): 10527-10538.
https://doi.org/10.1016/j.aej.2022.04.006

Sinha G and Sinha D (2020). Enhanced weighted round robin algorithm to balance the load for effective utilization of resource in cloud environment. EAI Endorsed Transactions on Cloud Systems, 6(18): e4.
https://doi.org/10.4108/eai.7-9-2020.166284

Sulimani H, Sulimani R, Ramezani F, Naderpour M, Huo H, Jan T, and Prasad M (2024). HybOff: A hybrid offloading approach to improve load balancing in fog environments. Journal of Cloud Computing, 13: 113.
https://doi.org/10.1186/s13677-024-00663-3

Waghmode ST and Patil BM (2023). Adaptive load balancing in cloud computing environment. International Journal of Intelligent Systems and Applications in Engineering, 11(1s): 209-217.

Wang L and Lu G (2016). The dynamic sub-topology load balancing algorithm for data center networks. In the International Conference on Information Networking, IEEE, Kota Kinabalu, Malaysia: 268-273.
https://doi.org/10.1109/ICOIN.2016.7427075

Wang X, Sun Y, and Ding D (2022). Adaptive dynamic programming for networked control systems under communication constraints: A survey of trends and techniques. International Journal of Network Dynamics and Intelligence, 1(1): 85-98.
https://doi.org/10.53941/ijndi0101008

Xu R, Chen H, Liang X, and Wang H (2016). Priority-based constructive algorithms for scheduling agile earth observation satellites with total priority maximization. Expert Systems with Applications, 51: 195-206.
https://doi.org/10.1016/j.eswa.2015.12.039