Task Scheduling on Cloudlet in Mobile Cloud Computing with Load Balancing

Poonam[1†] and Suman Sangwan[2††],

Deenbandhu Chhotu Ram University of Science and Technology, Haryana, India

Summary
The recent growth in the use of mobile devices has contributed to increased computing and storage requirements. Cloud computing has been used over the past decade to cater to computational and storage needs over the internet. However, the use of various mobile applications like Augmented Reality (AR), M2M Communications, V2X Communications, and the Internet of Things (IoT) led to the emergence of mobile cloud computing (MCC). All data from mobile devices is offloaded and computed on the cloud, removing all limitations incorporated with mobile devices. However, delays induced by the location of data centers led to the birth of edge computing technologies. In this paper, we discuss one of the edge computing technologies, i.e., cloudlet. Cloudlet brings the cloud close to the end-user leading to reduced delay and response time. An algorithm is proposed for scheduling tasks on cloudlet by considering VM's load. Simulation results indicate that the proposed algorithm provides 12% and 29% improvement over EMACS and QRR while balancing the load.

Keywords:
Cloud computing, Cloudlet, Edge Computing, Mobile Cloud computing

1. Introduction

According to World Advertising Research Center (WARS), around 2 billion people currently access the internet through their mobile phones, which is expected to reach 3.7 billion by 2025. Nowadays, mobile phones are used for all kinds of applications like Augmented Reality, file editing, chatting, video streaming, and gaming. Mainly mobile phones are used for interactive applications. Although recent advances in technologies, mobile phones still have insufficient resources due to restrictions on size, weight, battery, and memory, which requires a technology that can meet all mobile users' demands without compromising resources, processing speed, and delay. Using the cloud for executing mobile applications gives new directions, leading to the concept of mobile cloud computing. Applications of mobile devices can be offloaded on cloud servers for computations and processing at a lower cost. Mobile users experience large elasticity in the utilization of resources on an on-demand basis [1]. Applications can be rapidly provisioned on-demand basis with minimal management efforts. As a result, mobile cloud computing is introduced by incorporating mobile infrastructure and cloud computing.

Mobile cloud computing can be defined as an environment where all processing, storage, and execution of mobile applications are done outside the mobile devices, somewhere on the external cloud. The use of the cloud leads to ample storage and processing capabilities for mobile devices through fully utilizing cloud resources. However, distant locations of cloud servers lead to increased network latency, processing time, and power consumption for mobile users. A new paradigm, cloudlet architecture, has been proposed by [2] initially. Cloudlet brings the cloud at a one-hop distance from the user leading to improved network latency, processing time, and power consumption. Mobile users' applications can now be offloaded to cloudlets instead of servers [3].

Limited computation and battery life of mobile devices lead to the transfer of tasks to the remote cloud to improve performance and reduce energy has captivated the interests of the MCC community. Various scheduling methods have been addressed in previous studies [4]. Depending on different architectures, scheduling methods provide improved performance and save energy for mobile devices. Balancing the load of the mobile cloud is also a significant research area. Load balancing is a way of distributing offloaded tasks over all nodes of the cloud uniformly to improve the overall performance of the cloud [5].

Mobile cloud computing is one of the top research areas as it enhances mobile devices' processing capabilities by integrating them with the cloud. The high points of this paper are as follows:

1. Study of existing scheduling and load balancing techniques in MCC.
2. Study of different edge computing techniques.
3. Propose a novel dynamic scheduling technique with load balancing for task scheduling on MCC.
4. The mathematical formulation of load, execution time, and cost.
5. Comparison of the proposed technique with QRR and EMACS.

This study has been structured as follows. Section 2 describes related work, and section 3 discusses cloudlet. Section 4 and 5 introduce the mathematical model and proposed work, respectively, and Section 6 discusses the

Manuscript received October 5, 2023
Manuscript revised October 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.10.10

simulation results. Lastly, section 7 concludes this paper with future directions.

2. Related Work

Cloudlet is one of the mobile cloud computing architectures whose concept was firstly given by Mahadev Satyanarayanan [2]. It provides a mechanism for extending the state of mobile cloud computing. It is a middle-tier in three-tier mobile computing architecture. This edge computing technology resolves mobile cloud computing technology by bringing cloud resources close to mobile users [6].

Limited computation and battery life of mobile devices lead to the transfer of tasks to the remote cloud to improve performance and reduce energy has captivated the interests of the MCC community. Various scheduling methods have been addressed in previous studies. Depending on different architectures, scheduling methods provide improved performance and save energy for mobile devices. Balancing the load of the mobile cloud is also a significant research area. Load balancing is a way of distributing offloaded tasks over all nodes of the cloud uniformly to improve the overall performance of the cloud.

Wei et al. [7] proposed an algorithm that minimizes energy consumption while maximizing profit. This algorithm provides approximately 60% better load variation than the random selection scheme in the case of light load. In heavy loads, it selects applications with minimum energy and maximum profit. Lin et al. [8] proposed a task scheduling algorithm in a mobile cloud environment that offloads tasks on local cores of mobiles and the cloud. Simulation results show that this algorithm minimizes delay with significant energy reduction, and tasks are completed within deadline constraints.

L. Shakkeera [9] proposed energy-aware application scheduling and consolidation algorithm. A hybrid cloud model is proposed to utilize idle resources of nearby mobile devices. This algorithm provides a significant decrease in energy consumption by application consolidation. Idle servers are shut down for energy saving. This algorithm minimizes the response time and costs while improving the quality of service parameters.

Offloading methods have been addressed on several offloading architectures, such as Honeybee and COSMOS [10, 11]. Based on these architectures, offloading methods are used to enhance the mobile device's performance and save energy. Due to the trade-off of the parameters in the offloading process, the QoS-related offloading methods include network bandwidth, deadline, and power consumption. [12]. Various offloading methods like Artificial intelligence-based applications, Directed Acyclic Graphs (DAG) scheduling, Game theory, Lyapunov optimization, Markov Decision Process, and deep learning methods [13, 14] have been applied in various areas. X. Wei et al. [15] proposed HACAS, which balances the system's load in both cases when the load is high, and the load is low, with maximum profit and minimum energy consumption.

3. Cloudlet

Mobile devices are mainly used for interactive, real-time applications like augmented reality. As mobile devices are poor in resources, their execution and storage need to be done outside, like in the cloud. Cloud provides a resource-rich infrastructure on an on-demand basis, eliminating the resource poverty of mobile devices [4]. The main limitation in the use of the cloud by mobile users is long WAN latencies due to multi-hop distance. Cloudlets can solve this problem without being WAN limited. Cloudlet is one of the edge computing technologies [5, 6]. The difference between different edge computing technologies, cloudlet, fog computing, and mobile edge computing, is given in Table 1. The concept of cloudlet was first introduced by Mahadev Satyanarayanan [2].

Table 1: Comparison of Edge Computing Technologies

S. NO.	***Edge Computing → Implementation / Parameters ↓***	***Cloudlet***	***Fog Computing***	***Mobile Edge Computing***
1.	Devices	Diverse	Dedicated	Dedicated
2.	Service Access	One-Hop	One-Hop	One-Hop
3.	Architecture	Decentralized	Decentralized	Decentralized
4.	Network	WI-FI	Bluetooth, WI-FI, Mobile Network	Mobile Network
5.	Storage	High	Low	High
6.	Computation	High	Low	High
7.	Non-IP-Support	No	Yes	No
8.	Context Awareness	Low	High	Low
9.	Inter-Node Communication	Low	High	Low
10.	Power Consumption	Low	Low	High
11.	Computation Time	Low	High	Low
12.	Location	Local/ close to ending device	Anywhere between end device and cloud	Co-located with Radio network controller

A cloudlet is a reliable, resource-rich cluster of computers connected to the internet to provide full-fledged services to proximate mobile users. It is considered a proxy of the central cloud, located somewhere in the middle of the cloud, and mobile users, as in Figure 1 [16].

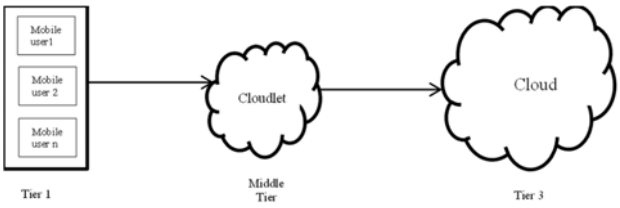

Fig. 1 Three-tier architecture.

The numbers of servers are few in cloudlet as compared to the cloud as they are created and used locally. Therefore a cloudlet can be defined as a data center in a box, which is self-managed, trusted, well connected to the internet, and consumes less power. The main objective of cloudlet is to bring the cloud close to mobile users [17]. Cloudlets have decentralized architecture and are dispersed widely. They can be used to cater to the need of nearby mobile devices such as coffee shops and hospitals. Cloudlets need no fixed infrastructure; they can be formed by using nearby resources like nearby mobile phones, and laptops, providing a dynamic infrastructure.

Fog Computing is a decentralized platform where data, applications, compute, and storage is retained somewhere between source and cloud. It includes bringing intelligence and processing nearer to where the data is produced, leading to improved productivity; still, it might also be done for safety and compliance reasons [5]. Edge Computing (EC) offers a highly distributed computing environment that can be used to organize applications and services as well as to store and process content close to mobile users. EC would empower applications to be fragmented into small tasks, with some of the tasks accomplished at the native or provincial clouds as far as the latency and accuracy are conserved.

4. Mathematical Problem Formulation

This section discusses the mathematical model of the problem. Mobile users' requests are offloaded on cloudlets which the cloudlet manager handles. Cloudlet manager monitors all VMs and evaluates their load status based on utilization of network bandwidth, RAM, and CPU processing [18]. For the mathematical formulation of the problem, let us consider n tasks arriving for execution on p cloudlets and m VMs. Let us consider that all the tasks have the same priority and are independent. Tasks are represented as $T_1, T_2, T_3 \dots \dots . T_n$, VMs as $VM_1, VM_2, VM_3 \dots \dots . VM_m$. Let $tl_1, tl_2, tl_3 \dots \dots . tl_n$ represents task lengths for tasks $T_1, T_2, T_3 \dots \dots . T_n$ respectively.

4.1 Load

System load is an indicator of the utilization of system resources, and it is defined in terms of network bandwidth, RAM, and CPU processing [19]. Load is evaluated as shown in (1.

$$Load\_VM, L = \frac{1}{1 - BW} * \frac{1}{1 - RAM} * \frac{1}{1 - CPU} \quad (1)$$

BW, RAM, and CPU represent bandwidth, memory, and CPU utilization in percentage, which are used uniformly for load evaluation. L also represents load in percentage, i.e., on a scale of 0 to 1. Extensive utilization of Bandwidth (BW) of the network leads to network overloading, and that of RAM leads to memory overloading. Similarly, if the number of processes running on the CPU is larger than the threshold, it results in CPU overloading. Based on the load status of VMs, the arrived tasks are assigned to VMs.

4.2 Execution Time of Task

The execution time is the time to execute a task on a VM. It is calculated by using the following (2.

$$ET\_Task = \frac{tl}{ps * \alpha} \quad (2)$$

Where tl is the task length of a task, ps is the processing speed of the CPU and α represents the number of processing units.

4.3 Execution Time of VM

VM execution time is then given by summing up the minimum execution time of all tasks running on that VM, given by the following (3.

$$ET_{VM} = min \sum ET\_Task \quad (3)$$

4.4 Makespan

Makespan is one of the essential criteria that show the highest finishing time among all tasks. Therefore, a low value of makespan means that the task scheduling algorithm is successful in the efficient allocation of tasks to VMs. Generally, makespan is computed as given by (4 [20].

$$Makespan, M = max\{TF_i \mid \forall\, i \in List\ of\ Tasks\} \quad (4)$$

Where TF_i is the finishing time of the i^{th} task.

4.5 Cost

It is the total cost incurred for the execution of tasks on cloudlet. It is calculated using the following (5:

$$Total\_Cost, TC = Data\_Transfer\_Cost + VM\_Cost \quad (5)$$

Where $Data\_Transfer\_Cost$ is the cost incurred in transferring data from mobile to cloud, and VM_Cost is the cost incurred in executing tasks on VM.

4.6 Fitness Function

The fitness function for this scheduling problem can be formed using (1, (4, and (5. Mathematically fitness function is represented using (6.

$$Fitness\ Function, F(x) = \min(a_1 * L + a_2 * M + a_3 * TC) \quad (6)$$

Where $L, M,$ and TC represent load, makespan, and Total cost, respectively. a_1, a_2 and a_3 are constants responsible for optimizing fitness function such that $a_1 + a_2 + a_3 = 1$. Values for these constants are considered as $a_1 = 0.5$, $a_2 = 0.25$ and $a_3 = 0.25$ [21].

5. Proposed Work

A dynamic algorithm has been proposed for scheduling tasks on VMs along with balancing the load. The proposed algorithm consists of two modules, Load_Evaluator and VM_Allocation, for scheduling and balancing the load on the cloudlet. The flowchart of the proposed algorithm is shown in Figure 2. Tasks arrive at cloudlet broker randomly. The resource manager evaluates a load of all VMs on each cloudlet using (1. Execution time of each VM is also evaluated using (3. Based on the load and execution time of VMs, the cloudlet is selected to execute the tasks.

5.1 Load Evaluator

The primary function of the load evaluator module is to calculate the load of VMs. Load is evaluated by considering bandwidth, memory, and CPU utilization, as shown by (1. It is used to check whether VM is overloaded or not. This algorithm is executed repeatedly for auto-generation of balanced VM's list, therefore considering system dynamics. The pseudocode for load evaluation is given in Figure 3. A list of load-balanced VMs is created for task assignment. The load evaluator is a part of the cloudlet structure, as shown in Figure 4.

The resource manager evaluates the load status of all VMs based on RAM CPU and Bandwidth utilization. The resource manager runs at regular intervals and updates the load status of all VMs. Interval should be neither too large nor too small, avoiding system instability. The load evaluator uses the current load statistics and decides whether VM is overloaded or underloaded. Task schedular then allocates tasks to a VM on the cloudlet selected by the Load Evaluator.

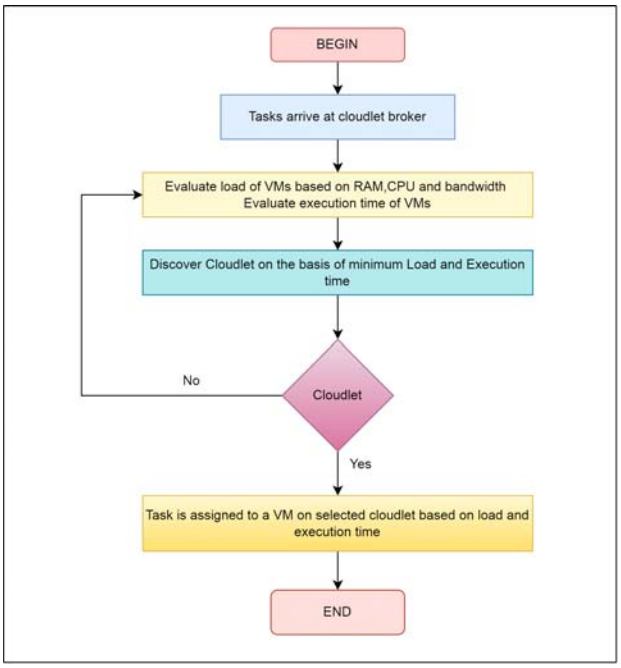

Fig. 2 Proposed Algorithm Flowchart.

Load_Evaluator()
Input: VM's status
Output: List_BVM
List_VM: Active VM's
$Load\_VM = \frac{1}{1-BW} * \frac{1}{1-RAM} * \frac{1}{1-CPU}$
List_OVM: List of overloaded VMs
List_BVM: List of balanced VMs
Step 1: **For** i=1 to m in List_VM **do**
Find Load_VM[i];
If Load_VM[i]> Threshold **then**
add VM[i] to List_OVM;
Else
add VM[i] to List_BVM;
End if
End for
Step 2: Sort List_BVM in ascending order of Load_VM
Go to step 1

Fig. 3 Load Evaluator.

5.2 Task Allocation

The Load Evaluator module returns a list of VMs to which tasks can be allocated. Tasks are sorted in increasing

order of their task length and allocated to the best fit VM based on VM's load status and execution time. Execution time is calculated using (2 and (3. In case all the VMs are overloaded, a new VM is created, and arriving requests are allocated. The pseudocode for Task allocation is given in Figure 5.

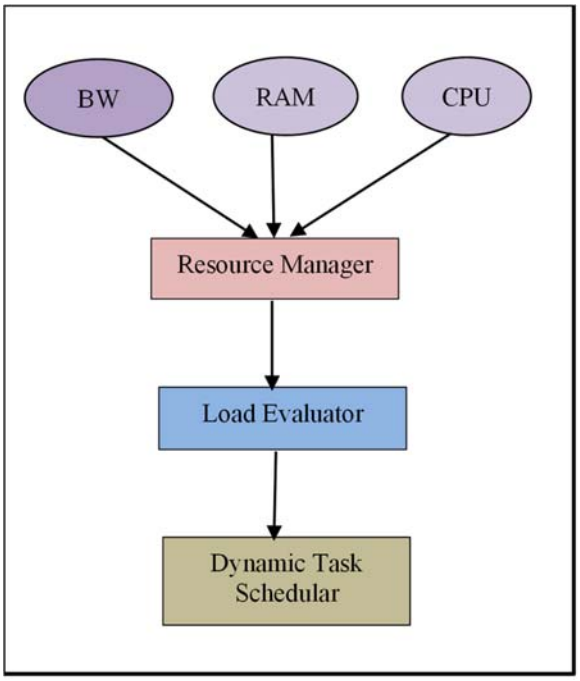

Fig. 4 Structure of Cloudlet Manager.

```
Task_Allocation()
Input: List_BVM, List_Task
Output: Allocation of Tasks on VMs
List_Task: List of n Tasks
Step 1: If List_BVM≠0 then
        For i=1 to m in List_BVM do
            Calculate Execution Time of VM[i];
            ET_VM[i]= Execution Time of VM[i];
        End for
        List_Sort= sort ET_VM in ascending order;
        Sort List_Task in increasing order of task length;
        For all Ti∈List_Task and VM∈ List_Sort do
           Allocate Tasks to VMs in First come first serve order;
        End for
Step 2: Else
        Create a new VM;
        Allocate Task to new VM;
      End if
```

Fig. 5 Task Allocation.

The proposed algorithm pseudocode is given in Figure 6. Both Load_Evaluator and Task_Allocation modules are called in this algorithm to carry out the task scheduling while uniformly balancing the load on VMs.

```
Dynamic_Scheduling()
Input: List_Task, List_VM
Output: Tasks Allocation, Balanced VMs
List_Task: List of n Tasks
List_VM: List of m VMs
Step 1: For each task in List_Task do
        For each VM in List_VM do
            Call Load_Evaluator();
            Call Task_Allocation();
        End for
        End for
```

Fig. 6 Dynamic Scheduling Algorithm with Load Balancing.

6. Results and Discussions

A dynamic task scheduling algorithm with load balancing is proposed whose objective is to reduce makespan and cost of task allocation while balancing the load in a cloudlet environment. In this section, the simulation setup, dataset, and results are discussed.

6.1 Simulation Setup

Simulation has been carried out on a 64-bit Windows 8 machine having Intel Core i3 and 4 GB RAM using the Cloud Analyst tool with Eclipse Java Neon.3 IDE. Parameters that are considered for simulation are shown in Table 2. The simulation setup used consists of a set of physical machines or hosts, VMs, and users. Tasks are considered to be mutually independent; they have no constraints. Three VMs per cloudlet are created for the execution of the different number of tasks which vary from 50 to 250. The length of tasks may vary from 100MI to 500MI. The policy used for execution is space shared. Tasks are scheduled based on the proposed algorithm, and results are compared with QRR and EMACS algorithms [22]. Simulation is carried out on the CLARKNET dataset [23]. These datasets are offered by "Ake Sandgren, Bill Nitzberg, and Victor Hazlewood" in the standard workload format (.swf) recognized by the Cloud Analyst tool [24].

6.2 Makespan

Makespan is the highest finishing time among all tasks. The makespan of all three algorithms is shown in Figure 7. The proposed algorithm shows an improved makespan compared to others as the number of tasks increases. First, The proposed algorithm allocates tasks to the cloudlet, which are only one hop distance to the user, which leads to reduced makespan in contrast to when tasks are allocated to VMs on cloud servers.

Table 2: Simulation Parameters

Simulation Parameters	
No. of Hosts	3
No. of Cloudlets	10-15
No. of servers in Cloudlet	1-5
No. of VMs on Cloudlet	5-25
No. of Users	10-80
No. of tasks	50-250
Length of Tasks	100 MI to 500 MI
Storage	1 TB
RAM	2 GB
Type of Policy	Space shared
Type of VMM	Xen
Operating system	Linux
No of Processors	one each
Processing Speed	50-300 MIPS
Bandwidth	100-200 Mbps
Dataset	CLARKNET

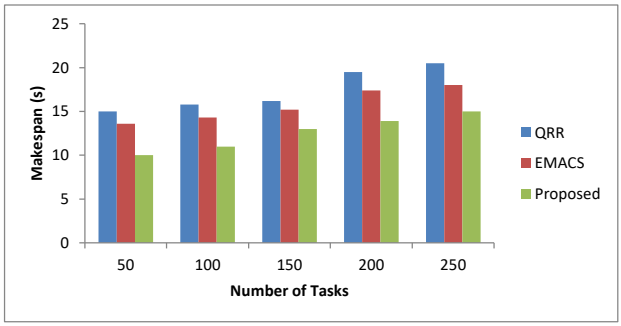

Fig. 7 Average Makespan.

Second, the proposed algorithm allocates tasks to the VM on a cloudlet whose load and execution time is minimum. Load is the primary parameter in the proposed algorithm. The load of a VM is calculated on the basis of CPU, RAM, and bandwidth utilization. The load is minimized if all the parameters are small. If one resource is highly utilized, the overall load value would be high. So proposed algorithm assigns tasks to the VM with a smaller value of load and execution time, leading to reduced makespan as compared to QRR and EMACS. All of this means that resource utilization of the proposed algorithm is more uniform than QRR and EMACS algorithms.

6.3 Load

System load represents the resource utilization of system resources. Load is the prime parameter in the proposed algorithm, which depends upon CPU, RAM, and bandwidth utilization. The load status of all three VMs for the proposed, QRR and EMACS algorithms is illustrated in Figure 8. Simulation results show that the proposed algorithm achieves better load balancing among VMs than the other two. Tasks are assigned to the VMs by checking their load status to avoid overloaded and underloaded states, thus leading to uniform load distribution among all VMs. EMACS is the second-best algorithm to balance the load among VMs.

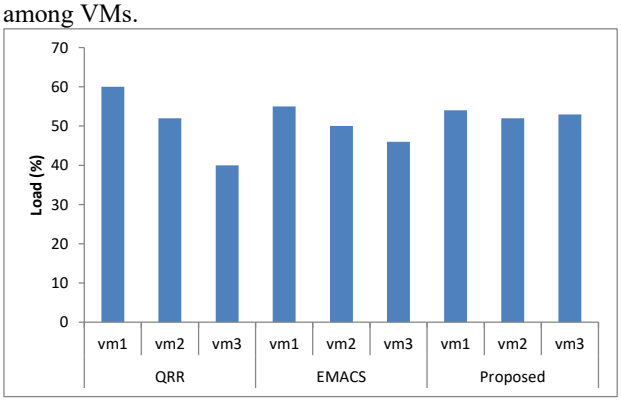

Fig. 8 Load Comparison on Three VMs.

6.4 Cost

Figure 9 illustrates the comparison of the cost of QRR and EMACS and proposed algorithms. The cost of task execution consists of the cost incurred in transferring the task from a mobile device to the cloudlet and the cost incurred in executing the task on a VM. As cloudlets are close to mobile users, the transfer time incurred is less than when tasks are offloaded to the cloud. Moreover, tasks are allocated to a VM only after verification of its load status leading to uniform load distribution among all VMs and reduced execution time. All this leads to reduced costs. It can be observed from Figure 9 that the proposed algorithm cost is considerably less than QRR and EMACS algorithms.

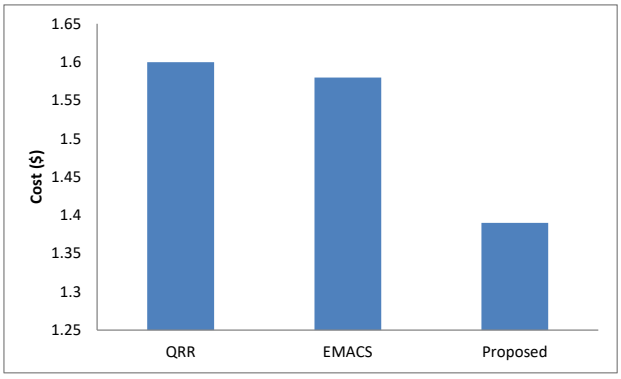

Fig. 9 Comparison of Cost.

6.5 Drop Rate

Figure 10 represents the drop rate of QRR, EMACS, and the proposed algorithms.

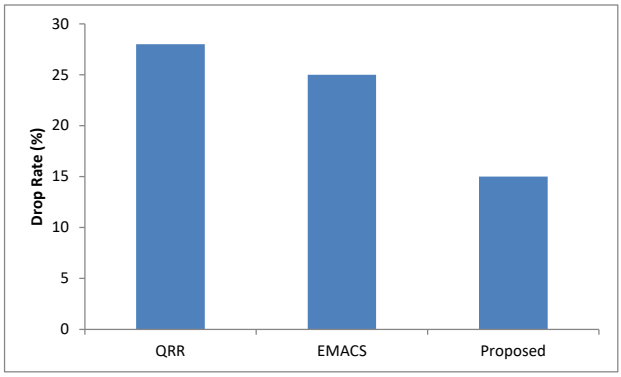

Fig. 10 Drop Rate.

It is visible that the drop rate of the proposed algorithm is much lower than QRR and EMACS as the proposed algorithm allocates tasks based on the remaining capacity of VMs, which leads to a reduced drop rate. Uniform load distribution on all the VMs is one of the crucial factors for reduction in drop rate.
The simulation results show that the proposed algorithm excels EMACS at about 12% and QRR at about 29%, respectively.

6. Conclusion and Future Work

A novel Dynamic task scheduling technique with load balancing is proposed in the cloudlet environment. This technique schedule tasks on the appropriate VM by considering their load based on memory, bandwidth, and CPU utilization, which are significant resources. Simulation results indicate that under all possible situations proposed technique gives reduced makespan and cost while balancing load than the QRR and EMACS algorithms. It also permits enhancing the balancing of load among VMs. The proposed technique provides 12% and 29% improvement over EMACS and QRR algorithms. This work can further be extended as follows: firstly, by offloading tasks on the cloud when the number of tasks is enormous to be executed on cloudlet. Secondly, task priority can be considered, which matters in some applications. Third, it can be further optimized by applying any optimization technique.

References

[1] Dinh, H. T., Lee, C., Niyato, D., Wang, P.:*A survey of mobile cloud computing: architecture, applications, and approaches.* In: Wireless communications and mobile computing, vol. 13, no. 18, pp. 1587-1611 (2013).

[2] Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: *The case for VM-based cloudlets in mobile computing.* In: IEEE Pervasive Computing, vol. 8, no. 4, pp. 14-23 (2009).

[3] Somula, R., Anilkumar, C., Venkatesh, B., Karrothu, A., Kumar, C. P., Sasikala, R.: *Cloudlet services for healthcare applications in mobile cloud computing.* In: Proceedings of the 2nd international conference on data engineering and communication technology, Springer, Singapore, pp. 535-543 (2019).

[4] Singh, S.: *Load balancing algorithms in cloud computing environment.* In: International Journal of Advanced Research in Computer Science, vol. 9, no. 2 (2018).

[5] Dolui, K., Datta, S. K. (2017).: *Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing.* In: 2017 Global Internet of Things Summit (GIoTS), pp. 1-6 (2017).

[6] Nayyer, M. Z., Raza, I., Hussain, S. A.: *A survey of cloudlet-based mobile augmentation approaches for resource optimization.* In: ACM Computing Surveys (CSUR), vol. 51, no. 5, pp. 1-28 (2018).

[7] Wei, X., Fan, J., Lu, Z., Ding, K.: *Application scheduling in mobile cloud computing with load balancing.* In: Journal of Applied Mathematics, *(2013).*

[8] Lin, X., Wang, Y., Xie, Q., Pedram, M.: *Energy and performance-aware task scheduling in a mobile cloud computing environment.* In: 2014 IEEE 7th international conference on cloud computing, pp. 192-199 (2014).

[9] Shakkeera, L., Tamilselvan, L.: *Energy-Aware Application Scheduling and Consolidation in Mobile Cloud Computing with Load Balancing.* In: Emerging Research in Computing, Information, Communication and Applications, Springer, New Delhi, pp. 253-264 (2016).

[10] Sangwan, S.: *A comparative study of various load balancing algorithms in cloud computing environment*. In: IJARET, vol. 11, no. 12, pp. 2735-2760 (2020).

[11] Tapale, M.T., Goudar, R.H., Birje, M.N., Patil, R.S.: *Utility based load balancing using firefly algorithm in cloud.* In: Journal of Data, Information and Management, vol. 2, no. 4, pp. 215-224 (2020).

[12] Liao, Z., Ma, Y., Huang, J., Wang, J. Wang, J.: *HOTSPOT: A UAV-assisted dynamic mobility-aware offloading for mobile-edge computing in 3-D space.* In: IEEE Internet of Things Journal, vol. 8, no. 13, pp. 10940-10952 (2021).

[13] Haris, M., Zubair, S.: *Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing.* In: Journal of King Saud University-Computer and Information Sciences (2021).

[14] Lu, J., Hao, Y., Wu, K., Chen, Y. Wang, Q.*: Dynamic offloading for energy-aware scheduling in a mobile cloud*. In: Journal of King Saud University-Computer and Information Sciences (2022).

[15] Sangwan, S.: *Fuzzy firefly based intelligent algorithm for load balancing in mobile cloud computing.* In: Computers, Materials & Continua, vol. 74, no.1, pp. 1783–1799 (2023).

[16] Satyanarayanan, M., Lewis, G., Morris, E., Simanta, S., Boleng, J., Ha, K.: *The role of cloudlets in hostile*

environments. In: IEEE Pervasive Computing, vol. 12, no. 4, pp. 40-49 (2013).
[17] Verbelen, T., Simoens, P., De Turck, F., Dhoedt, B.: *Cloudlets: Bringing the cloud to the mobile user*. In: Proceedings of the third ACM workshop on Mobile cloud computing and services, pp. 29-36 (2012).
[18] Chen, C., Bao, W., Zhu, X., Ji, H., Xiao, W., Wu, J.: *AGILE: A terminal energy efficient scheduling method in mobile cloud computing*. In: Transactions on Emerging Telecommunications Technologies, vol. 26, no. 12, pp. 1323-1336 (2015).
[19] Chabbouh, O., Rejeb, S. B., Agoulmine, N., Choukair, Z.: *Service scheduling scheme based load balancing for 5G/HetNets Cloud RAN*. In: 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), pp. 843-849 (2017).
[20] Mansouri, N., Zade, B. M. H., Javidi, M. M. (2019).: *Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory*. In: Computers & Industrial Engineering, vol. 130, pp. 597-633 (2019).
[21] Walia, N. K., Kaur, N., Alowaidi, M., Bhatia, K. S., Mishra, S., Sharma, N. K., Kaur, H.: *An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments*. In: IEEE Access, vol. 9, pp. 117325-117337 (2021).
[22] Zhang, C., Yang, Z., He, X., Deng, L.: *Multimodal intelligence:Representation learning, information fusion, and applications*. In: IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 3, pp. 478-493 (2020).
[23] Rashidi, S., Sharifian, S.: *A hybrid heuristic queue based algorithm for task assignment in mobile cloud*. In: Future Generation Computer Systems, vol. 68, pp. 331-345 (2017).
[24] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., Buyya, R.*: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*. In: Software: Practice and experience, vol. 41 no. 1, pp. 23-50 (2011).

Ms. Poonam is a Research Scholar in Deptt. Of Computer Sci. and Engg. at Deenbandhu Chottu Ram University of Science and Technology, Murthal, Sonepat, India. She holds a MCA and BSc Computer Science degree from Kurukshetra University, Kurukshetra, India. She has also qualified UGC-NET and JRF. Her research interests include Cloud Computing, Mobile Cloud Computing, and Networking.

Dr. Suman Sangwan is working as Professor in CSED in Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Haryana, India. She has been into teaching and research for about 17 years. She did her Ph.D. from Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Haryana, India. Her research areas include Heterogeneous Wireless Networks, Wireless Sensor Networks, Cloud Computing, Internet of Things(IoT) and VANETs. She did her M.Tech in Computer Science & Engineering from Kurukshetra University, Kurukshetra, India. She has published more than 60 papers in various journals of repute.