International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.2 (2025) pp. 3291-3300
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Optimizing Task Scheduling and Resource Allocation Using Multi-Criteria Framework in Fog-Assisted IoT Networks with Preemption

D.Deepakraj[1]*, V. Saidulu[2], Azham Hussain[3], S. Muruganandam[4], Sumit Kumar[5], J RajaSekhar[6]

[1] Dept. of Computer and Information Science, Annamalai University, Tamilnadu.
* **Corresponding Author Email:** deepakraj0708@gmail.com - **ORCID:** 0000-0003-0740-1969

[2] ECE Department, Mahatma Gandhi Institute of Technology, JNTUH University, Hyderabad.
Email: vsaidulu_ece@mgit.ac.in - **ORCID:** 0009-0000-5370-8668

[3]School of Computing, Universiti Utara Malaysia, 6010 UUM Sintok, Kedah, Malaysia
Email: azham.h@uum.edu.my- **ORCID:** 0000-0001-9169-7621

[4] Department of Computer Science and Business Systems, Panimalar Engineering College (Autonomous), Chennai
Email: murugan4004@gmail.com - **ORCID:** 0000-0001-8813-0456

[5]Dept. of CSE, Haridwar University, Roorkee
Email: dr.sumitcse@huroorkee.ac.in- **ORCID:** 0000-0002-1906-5539

[6] Department of IoT, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur Dist, AP, India
Email: rajasekharemb@gmail.com- **ORCID:** 0000-0003-3480-080X

Article Info:

DOI: 10.22399/ijcesen.2134
Received : 02 March 2025
Accepted : 05 May 2025

Keywords :

optimization,
Multi-Criteria Framework
IoT

Abstract:

The Internet of Things (IoT) can be developed further using fog and cloud computing environments. Task scheduling is highly effective for carrying out user requests in these settings, and the IoT-fog-cloud system's productivity is increased when IoT task requests are scheduled well. To address challenges such as latency, bandwidth overhead, and resource management, this paper proposes the Optimized Scheduling and Cluster-based Resource Allocation (OSCRA) model. The OSCRA model introduces a multi-criteria task scheduling with preemption using: (i) expectation-maximization (EM) clustering to group jobs by priority and deadline, (ii) a heap-based optimizer to schedule jobs based on SLA and QoS restrictions, and (iii) distributed resource management to assign resources effectively. Experimental results using iFogSim demonstrate that combining MSCFS and OSCRA enhances server utilization, reduces latency, improves throughput, and shortens response time, outperforming existing models.

1. Introduction

Computing is widely employed in many fields. Nevertheless, a number of problems with remote computing have emerged with the expansion of the Internet of Things (IoT). One of the most important enabling technologies in terms of applicability in smart cities is the internet of things (IoT), which is basically the use of internet networks to connect and interlink computers and other technologically-enabled items [1]. In the Internet of Things, data is collected in real-time from several physical sensors and devices and then disseminated via wireless networks. Applications of the Internet of Things IoT, including smart homes, driverless cars, and healthcare, have been growing steadily in recent years. With the help of cloud and fog networks, IoT provides an effective resource environment for industrial automation applications. Users have public access to resources in the cloud environment. Numerous applications that produce enormous volumes of data and have strict latency requirements can be created with millions of sensors and intelligent devices [2]. Fog computing is a sophisticated cloud computing environment that is faster, closer to users geographically, and uses a lot fewer resources for computation than the environment of the cloud. As a result, compared to cloud computing, it can offer reduced traffic and

delay in the IoT-fog-cloud network. Figure 1. depicts the IoT-fog-cloud network's three-layer design.

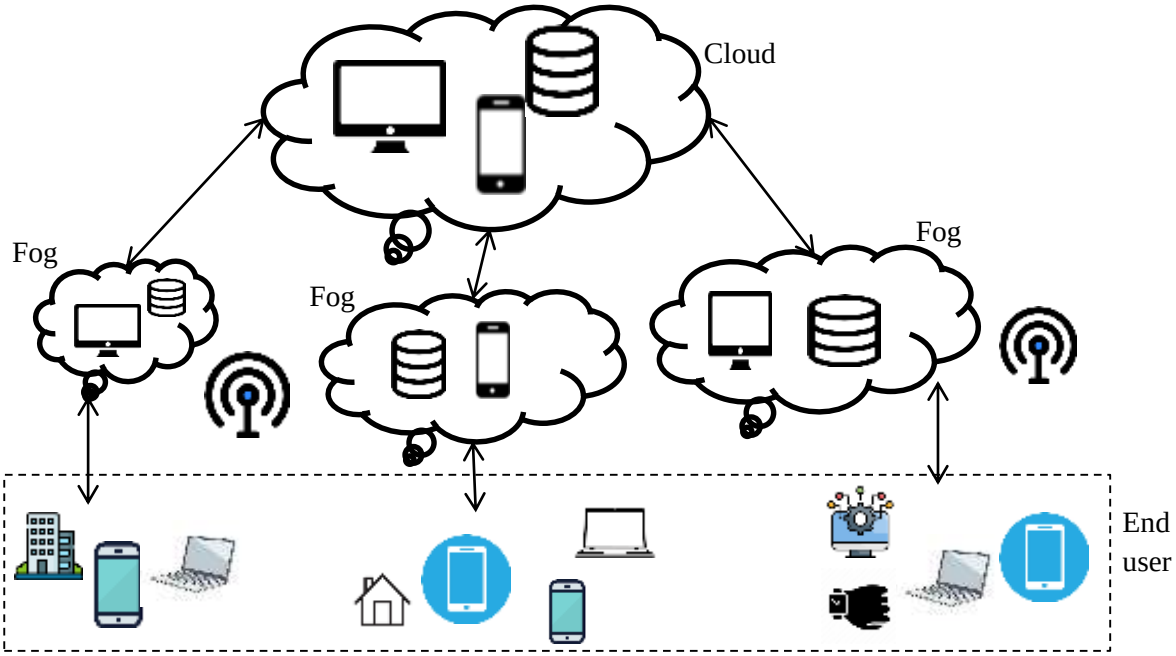

Figure 1. *Architecture of IoT-Fog Cloud network*

The cloud environment's servers, which can store and analyze large volumes of user data, are at the top layer. The network's edge and fog nodes make up the intermediate layer, which also contains smart gateways and mini-servers. Laptops, cellphones, automobiles, personal computers, sensors, and other IoT devices and end systems are all part of the lowest layer, known as the outer boundary of the IoT-fog-cloud network [3].

With several data management options and advancements in transmission technologies, using cloud infrastructure to store the massive amounts of data from numerous machines and devices as well as data analytics became inevitable [4]. The data from its surroundings is collected and sent by the IoT devices to be stored on the cloud. IoT devices benefit from the cloud, but at the expense of lengthy transmission delays. Intelligent gadgets that perform operations in near real time require resources immediately and cannot wait for them. Fog Computing has arisen to meet the demands of such devices [5].

IoT devices can access storage and computation services from fog devices. However, the fog nodes' limited resources and high request volume necessitate the use of an optimal scheduling technique. While fog computing supplements the cloud paradigm by addressing the delay barriers for work with a deadline, cloud computing addresses the issues of resource availability. [6] A hidden Markov model (HMM) is applied in a study that projects the presence of fog sources. IoT jobs have also been scheduled using the DO-HHO hybrid method, which stands for discrete opposition-based Harris Hawk Optimization (HHO). A scalable approach for scheduling time-sensitive jobs is presented in research [7].

In cloud computing and fog environments, work scheduling is regarded as an NP-hard problem [8]. This is why the relevant work section has suggested various task scheduling techniques that have been implemented utilizing artificial intelligence algorithms. [9] This combination is intended to solve the task planning issue in the IoT-fog-cloud networking and decrease the task makespan time, allowing jobs to be finished as quickly as possible and decreasing the fitness function to arrive at the problem's ultimate solution. The fitness function in this paper is makespan time. Two datasets are used for the experiments. Other algorithms are compared to the suggested approach, and the suggested method outperforms the compared algorithms [10].

2. Related works

Huang et al. [11] designed A multi-criteria study of decision-making proficiency in student's employability for multidisciplinary curriculums. There are four very important and useful discoveries and conclusions. First, self-efficacy (SE), self-control (SC), and self-regulation (SR) of the autonomy-learning performance of social

learning theory (SLT) all had a direct impact on judgeability, the most important decision-making employability factor. Because most employers require graduates of higher education programs who can evaluate, revise, and justify their self-action capacity in thinking, motivation, feeling, and cognition, as well as have the behavior from the interdisciplinary curriculum instilled that allows them to cultivate their self-observing experience, it is clear that the "AU" of the technological TAM model also had an impact on the "SR" of autonomy-learning performance.

Liu et al. [12] introduced An optimal scheduling method in IoT-fog-cloud network using combination of Aquila optimizer and African vultures optimization. Task scheduling is a very effective way to carry out user requests, and the IoT-fog-cloud system is more productive when IoT task requests are scheduled optimally. The Aquila Optimizer (AO) and African Vultures Optimization approach (AVOA) are combined in this research to create AO_AVOA, a hybrid meta-heuristic (MH) approach for scheduling IoT requests on IoTfog-cloud networks. By employing AO operators to identify the best solution while determining the ideal scheduling solution, AO_AVOA enhances the AVOA exploration phase. For both datasets, AO_AVOA enhances the makespan time by 8.36%, 2.61%, 104.38%, 17.56%, and 94.05%, respectively, in comparison to the AO, AVOA, PSO, HHO, and FA algorithms.

Xia et al. [13] proposed Optimized multiple-attribute group decision-making through employing probabilistic hesitant fuzzy TODIM and EDAS technique and application to teaching quality evaluation of international Chinese course in higher vocational colleges. The expansion of vocational education globally, the improvement of training for international students, and the supply of talent assistance for both domestic and international businesses. To handle the MAGDM under PHFSs, the probabilistic hesitant fuzzy TODIM-EDAS (PHF-TODIM-EDAS) technique is developed in this study. Lastly, the PHF-TODIM-EDAS technique is demonstrated using a numerical example for teaching quality evaluation of foreign Chinese courses in higher vocational colleges. This paper's primary contributions are as follows: (1) the TODIM technique, which is based on the EDAS technique, has been extended to PHFSs based on the CRITIC technique; and (2) weight numbers under PHFSs are derived using the CRITIC technique. In order to manage the MAGDM under PHFSs, the PHF-TODIM-EDAS approach was developed. (4) A numerical case study for assessing the quality of instruction of foreign Chinese courses in higher vocational institutions is provided, along with some comparative analysis to support the suggested PHF-TODIM-EDAS technique.

Zhao et al. [14] designed Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network. Virtual machine-based services are provided by modern cloud frameworks. The following problems were brought on by these frameworks: lengthy boot times, overhead, and needless expenses to run IoT apps.To execute mobility and delay-sensitive applications at the lowest possible cost, we suggest a novel architecture based on the Microservice Container Fog System (MSCFS). The Cost Aware Computational Optimization and Task Scheduling (CACOTS) framework, which breaks down task scheduling into many parts, is introduced in the paper. The suggested MSCFS and CACOTS schemes can improve server usage, according to the testing results. Reduce service latency, more efficiently average service bootup time, and minimize expenses.

Hassan et al. [15] suggested Internet of Vehicles (IoV)-Based Task Scheduling Approach Using Fuzzy Logic Technique in Fog Computing Enables Vehicular Ad Hoc Network (VANET). The infrastructure of the fog processes, which functions as a cloud extension, is near VANET, creating an atmosphere that is conducive to smart cars with IT hardware and efficient job management supervision. In VANET, there are limitations on vehicle processing power, bandwidth, time, and high-speed mobility. In order to reduce latency and enhance response time when offloading duties in the Internet of Vehicles, we suggested a fuzzy logic-based task scheduling system in VANET. Lastly, the study will create best practices and recommendations for implementing fog computing-based applications that successfully strike a compromise between performance metrics and energy consumption.

3. Proposed system

Due to several commonalities in task scheduling, the scope of this study extends beyond the cloud and fog computing paradigms. However, a number of factors contribute to the overall variances in work scheduling across the two systems; these distinctions are noted throughout the article where needed.

As seen in Figure 1, the suggested system is composed of three layers: the client application layer, the control layer, and the resource layer. In most cases, the client application layer creates loading tasks for arrive at the control layer at random from various IoT apps. On the other hand,

four modules make up the control layers, which process the output tasks at the different resource layers. With the help of the scheduling and task sequence modules, the Fog Cloud Agent (FCA), an orchestrator, is in charge of overseeing and controlling the tasks. FCA is a centralized controller that sits between system resources and user applications. FCA keeps an eye on and gathers data from entities, including metrics, configuration details, and logs. In a fog system network, these things are housed on hosts or virtual hosts.

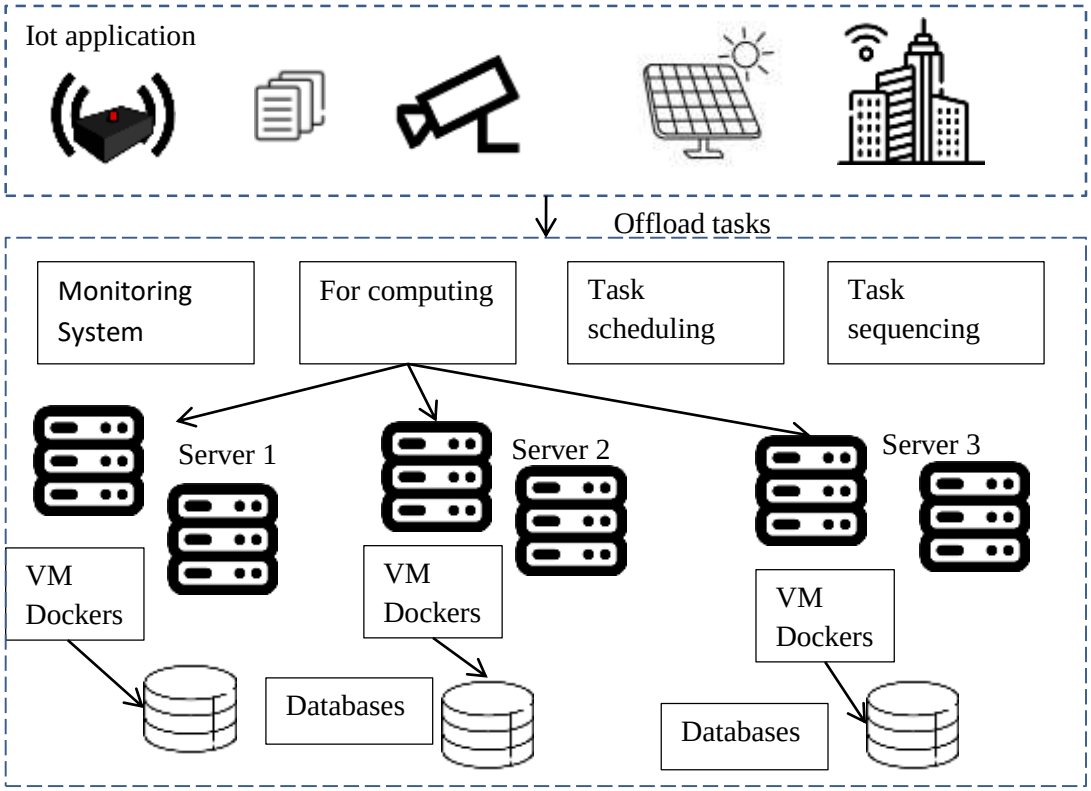

Figure 2. *Proposed architecture*

We introduced a multi-criteria task scheduling with a preemption technique to be planned with minimal cost within their deadlines in order to meet the requirements of the tasks. This model is called the Optimized Scheduling and Cluster-based Resource Allocation (OSCRA) model. Heterogeneous fog servers with a small number of homogenous virtual machines make up the resource layer. Internal docker-engine, which adds and removes container microservices for the fine-grained of loaded tasks, as the top layer of resources. Each of these jobs operates separately and needs a different set of resources to complete. Prior to job scheduling at the fog system, each task has vector attributes such data size, CPU requirements, and deadline. We suggested task sequence rules using an alternative technique to guarantee that the sorted algorithm is planned with the least amount of expense under their deadlines in order to satisfy the job requirements. The lookup table for monitoring system management contains the list of tasks for apps and the resource status. The lookup table is updated. The resource becomes available once an event, such finishing a task, has taken place. Through job swapping between servers, initial work scheduling will further enhance the resource's cost-function by distributing duties among many fog servers.

With the help of the scheduling and task sequence modules, the Fog Cloud Agent (FCA), an orchestrator, is in charge of overseeing and controlling the tasks. FCA is a centralized controller that sits between system resources and user applications. We integrated every fog cloud service at the wireless network's edge. Containers, a lightweight approach to virtualizing applications, have recently gained popularity in the fog cloud paradigm, particularly for IoT applications [23]. Usually, managing clusters of containers becomes essential, and coordinating development and deployment becomes a major challenge.

We assume a set of delay-sensitive tasks in the fog cloud system.

$$T = \{t_1, t_2, t_3 \dots\dots\dots\dots\} \quad (1)$$

We assumed that there are M heterogeneous fog servers in the fog cloud system. Each node has attributes including memory size, storage capacity, network bandwidth, and CPU processing rate (measured in millions of Procedures 2023, 11, 1162

5 of 18 instructions per second, or MIPS). Consequently, the ECT matrix, which has a size of f1*f2, is used to indicate the expected computation time (ECT) for task requests on nodes for n1 tasks and f2 computing nodes.

$$F = \{f_1, f_2 \dots\dots\dots\dots, M\} \quad (2)$$

Each fog server f_j possesses the following characteristics:

$$f_j = \{B_j, C_j, W_j, V_j\} \quad (3)$$

On the other hand, during the loading operation, B_j displays the bandwidth between the fog cloud server and the centric fog cloud agent. S shows the overall capacity (such as storage) of fog server j in the system, while ~j shows the compute rate of the j^{th} fog server. The number of virtual machine Docker deployments for microservices with the same capabilities in the fog server j is indicated by V_j.

The fog server precisely schedules one task at a time, much like the assignment issue, where each work is assigned to a single f_j. Here is how we indicate that task $t_i$has been assigned to fog server j:

$$\sum_{j=1}^{M} y_{a,b} = 1 \quad (4)$$

Because fog server j has limited resources, it has a limited number of virtual machines that docker can use to provide microservices for every task. As a result, less virtual machine capacity must be used for the specified job demand. This phrase is explained as

$$\sum_{i=0}^{N} x, w = V_j \quad (5)$$

Because a task must be sent to the fog server for processing, it receives additional communication during loading and is returned by the fog server as

$$RTT = \left(\frac{data^i}{Bw_{ij}^{up}} \pm \frac{data^j}{Bw_{ij}^{down}}\right) \quad (6)$$

As a result, each task's bandwidth demand is calculated as follows:

$$X_i\left(RTT + T_j^x\right) \le d_i \quad (7)$$

The first part is fog server resource matching, which uses a pairwise method to match each job to the appropriate fog server. The task sequence module is an essential module that arranges the tasks into various sequences so that the scheduler may do the best possible scheduling on them. Several components must process the IoT apps in order for them to be executed. For example, an algorithm designed to obtain the best optimal scheduling in the heterogeneous fog servers specifies the complete application process. Because the cost optimization challenge involves diverse fog servers. Thus, it is essential to determine how to choose edge servers to handle the sequential jobs. Remember that our job scheduling approach chooses the server with the lowest unit cost in order to reduce the fog computing system's expenses.

After scheduling certain activities, we refer to the server f_j's remaining resources as qfj. The task that maximizes the dot product is the largest, according to our definition [25]. We calculate hi as follows: After scheduling certain activities, we refer to the server F_j's remaining resources asQf_j. The task that maximizes the dot product is the largest, according to our definition [25]. Here's how we determine H_i:

$$h_i = S_i^j q + V_i^l + n_{ij} b_i^j q$$

$$qf_j = S_i^j q + V_i^l + n_{ij} b_i^j q$$

$$= pf_j - \sum P_{t*j} \quad (8)$$

The process of choosing the most suitable resource for every task on the heterogeneous fog servers is known as resource matching. Nonetheless, every task has vector qualities (like workload, deadline, and data size), and every resource has vector attributes (like cost, storage, bandwidth, and virtual machine capacity).

The Poisson process follows tasks that are introduced into the system at random in our situation. The tasks are submitted at random and in no particular order. Therefore, we must first order these tasks. Sequencing the work depends on the specified deadline, size, and free time. In order to categorize the submitted obligations according to three features, we thus establish the three rules. Offloaded tasks arrived at the fog cloud system at random. Using task sequencing, FCA put them in order of priority. To complete the execution in a way that is economical, the FCA takes advantage of the suggested sequence rules in a certain order. As a result, we will select an ideal order of activities

that fulfills the problem's constraints and objective function.

The preparatory task scheduling approach is obtained via task sequencing and resource matching. In terms of cost, task scheduling is not the best option for IoT applications. The initial does not stay constant because of seasonal variations in network contents and variations in cloud resources. Tasks t1 and t2 have different resource demand characteristics: data size: 10 MB, CPU required: 10, and deadline: 20; and data size: 30 MB, CPU required: 30, and deadline: 40. Conversely, servers $K_1$and K_2 may have the following resource attributes, in order: S: 15, VM_j: 4, B_j: 10, and S_j: 20, VM_j: 8, B_j: 20. Therefore, in order to identify our optimization problem and increase the application cost, we suggest using the task scheduling technique.

We break down the CACOTS framework for time complexity into several parts. (I) Resource Matching: To match each task to the various servers, we utilize the TOPSIS and AHP techniques. O(T * M) represents the time complexity.

Result

We produced real-world outcomes from trials on several IoT application benchmarks by the system in order to assess the performance of the suggested OSCRA architecture. The experimental setup used in this work was broken up into distinct components. (i) OSCRA implementation portion; (ii) Metric parameter and component calibration. (iii) Evaluation of computational offloading frameworks (iv) Comparison of algorithms and task scheduling. The experiment considers and compares the following current computational loading framework methodologies. In contrast, the workload analysis of IoT apps is displayed in Table 1.

Table 1. *Workload analysis*

Workload	EM	N	Communication cost
Augmented reality	64.6	5764	5G:7.4$
E- transport	24.5	2437	4G:6.3$
Heathcare	74.1	3241	3G:9.2$
3D- game	44.5	8273	Cellular:2$

The Simulation parameters and for servers specification are compared with workload. The following current methods for work scheduling are compared and examined.

Baseline 1: In the experiment section, we use the current cost-effective static task scheduling techniques and evaluate how well they perform in terms of application costs when contrasted with the suggested plan.

Baseline 2: We examine the effectiveness of the suggested system in terms of application costs by implementing the current cost-effective dynamic task scheduling algorithms in the experiment section.

Baseline 3: We examine the performance of the suggested scheme in terms of application costs by implementing the current cost-effective static task scheduling solutions without task scheduling in the experiment section.

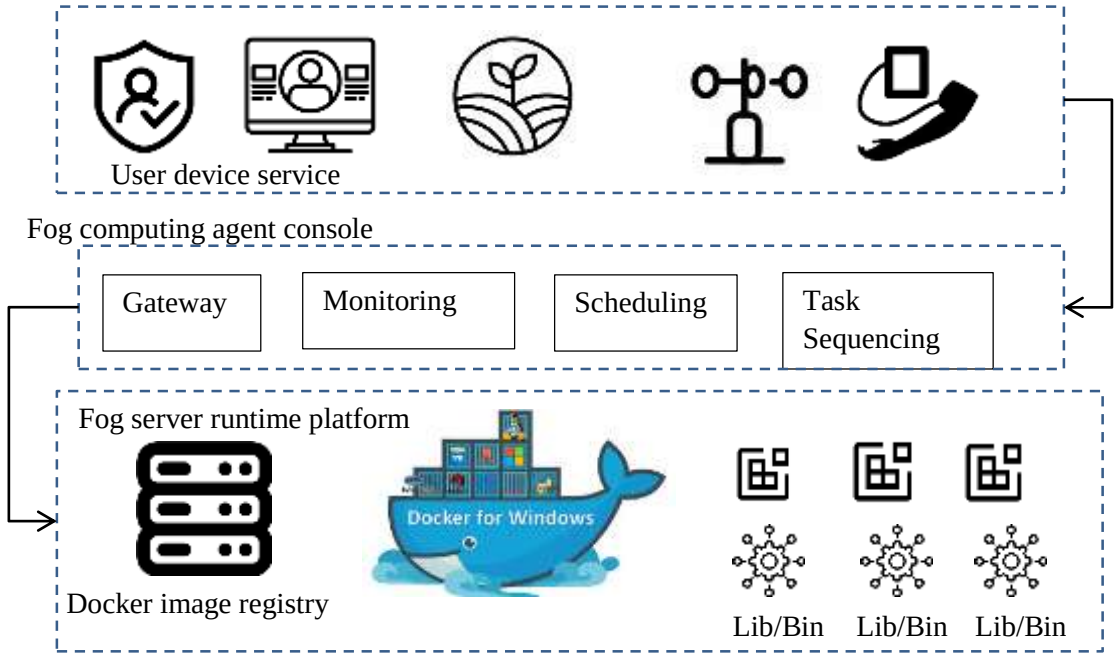

Figure 3. *Proposed system implementation*

As seen in Figure 2, we create Internet of Things (IoT) apps from the user's point of view using Android Studio and incorporate the GenyMotion emulator for testing. The elements calibrated for task organization for scheduling are the suggested task sequencing rules (i.e., EDD, SPF, and SSTF). The mean plot of the suggested task sequence rules with 95.0% Tukey HSD spaces is shown in Figure 3. As shown in Figures, the RPD significance of the EDD is significantly lower than that of the SPF and SSTF. The task sequence, shown in Figure 4, is used to map or arrange the activity. tasks that failed because of fog servers with limited resources. This, in a heterogeneous environment, results in lower delay fog clouds due to the EDD rule. As a result, EDD has been selected for the Mob-Cloud's job sequencing component.

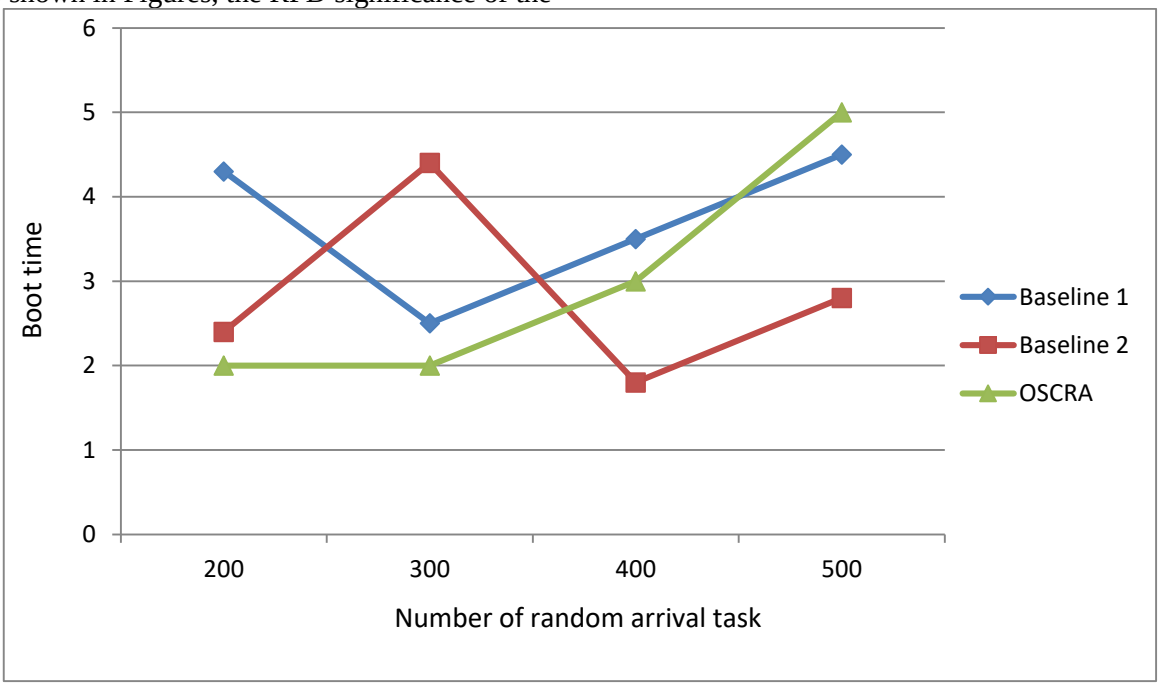

Figure 4. *Boot-time*

Figure 4 illustrates how the suggested technique has cut down on time. Four distinct numbers of applications are considered for testing in the experiment. The following formula is used in the study to determine the deadline for tasks for the various types of deadline constraints.

$$D_{a,i} = F_i + \gamma + F_{a.i} \quad (9)$$

The earliest finishing time and a specific percentage of the early finish time are used to determine the task's deadline (D). We used the parameter's range of values to demonstrate that it can handle the job deadline's tightness specifically, γ= f:2; :4; :6; :8; 1.

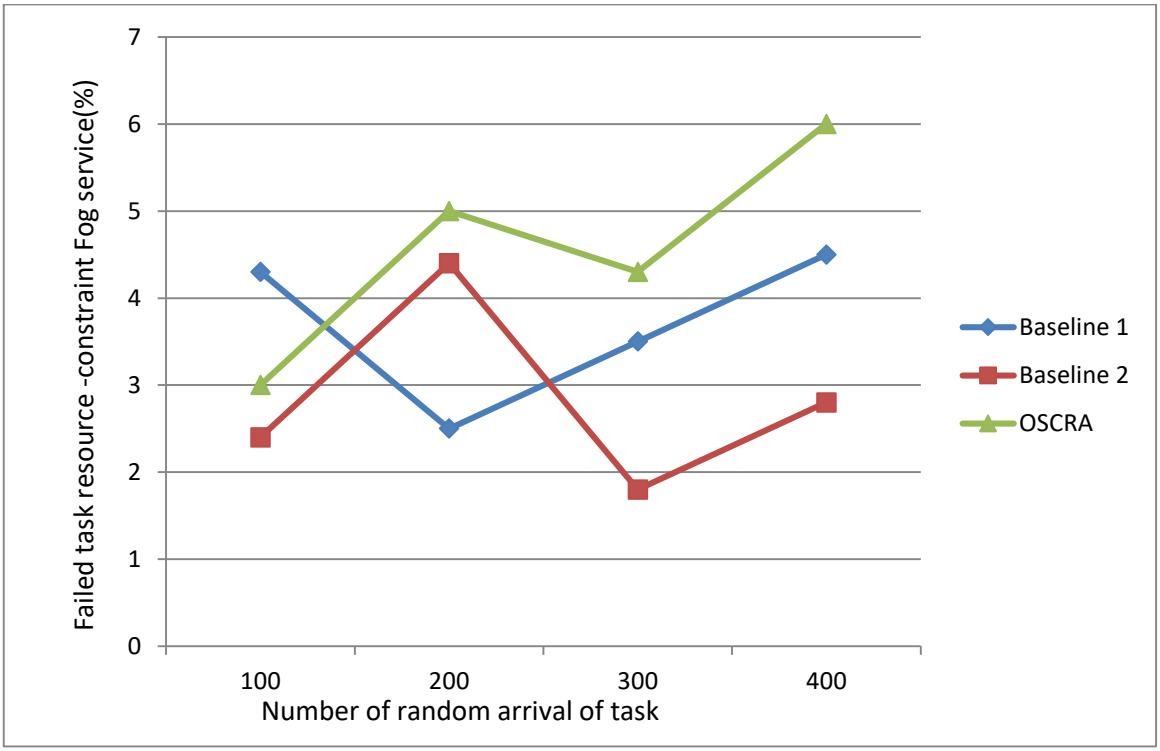

Figure 5. *Failed task resource-constraint Fog service*

The study makes use of RPD (Relative Percentage Deviation) statistical analysis to calculate the recital of the VFCN, MTOP, and CTOS. It assesses how much power is used by various parameters, the framework, and algorithm permutations across the component calibration parameter. We create Internet of Things (IoT) apps from the viewpoint of the user using Android Studio, and we integrate the GenyMotion emulator for testing. The OSCRA is made up of three primary layers: the resource layer, fog control agent layer, and user device layer. IoT applications use the Representational State Transfer (REST) API to load their tasks at random to the Fog Computing Agent Console (FCA).

Following the trial, Figure 5 shows that the suggested scheme's task failure ratio is lower than that of the baseline method. As a result, the suggested dynamic OSCRA approach improves costs and meets deadlines while being effective in a dynamic setting.

Device services inform FCA of the type of service needed to complete a task. To determine if the system is stable or not, the monitoring system keeps track of a look-up table of tasks and resources. The FCA's Task Sequence and Scheduler methods arrange the tasks in a certain order and schedule their processing on the heterogeneous fog server. The environment in which the system will function is called the runtime. Java Runtime Virtual (JVM), for example, is capable of efficiently running a dot class of Java programs. Compared to the current heavyweight virtual machine-based architecture, the suggested microservices container fog system-based computation loading has resulted in a shorter bootup time. On the other hand, we enhanced the suggested system's service resource use.There are several containers in use, and in order to use them all, they must be registered through registry services. Microservices can communicate with one another via REST APIs with less overhead.

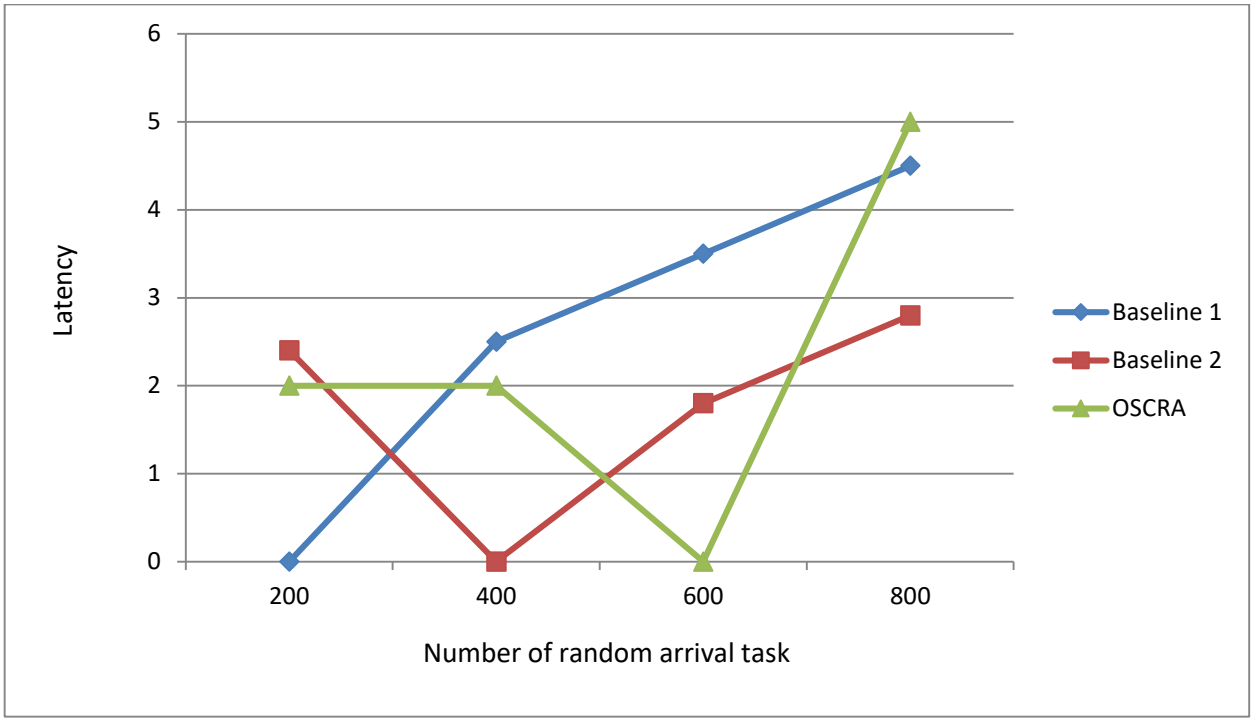

Figure 6. *Comparing latency*

Following the trial, Figure 6 shows that the suggested scheme's latency is lower than that of the baseline method. As a result, the suggested dynamic OSCRA approach improves costs and meets deadlines while being effective in a dynamic setting. The secret to IoT applications is cost-effective task scheduling. The main factors that are taken into account during scheduling are the application costs (such as communication and computation costs) and task deadlines. We take into account the various fog servers for the task scheduling issue. The projects with the shortest deadlines should be given priority over all others.

We take into account the various fog servers for the task scheduling issue. Our goal is to complete applications on time and with the least amount of expense. As stated in the equations, we take into account the various deadlines. Figure 7 shows that all programs run under their deadlines and experience reduced throughput under the suggested OSCRA strategy. As stated in the equations, we take into account the various deadlines. The primary explanation is that the suggested approach continuously refines neighbour space solutions until the ultimate ideal solution is obtained. Comparing our task scheduling strategy

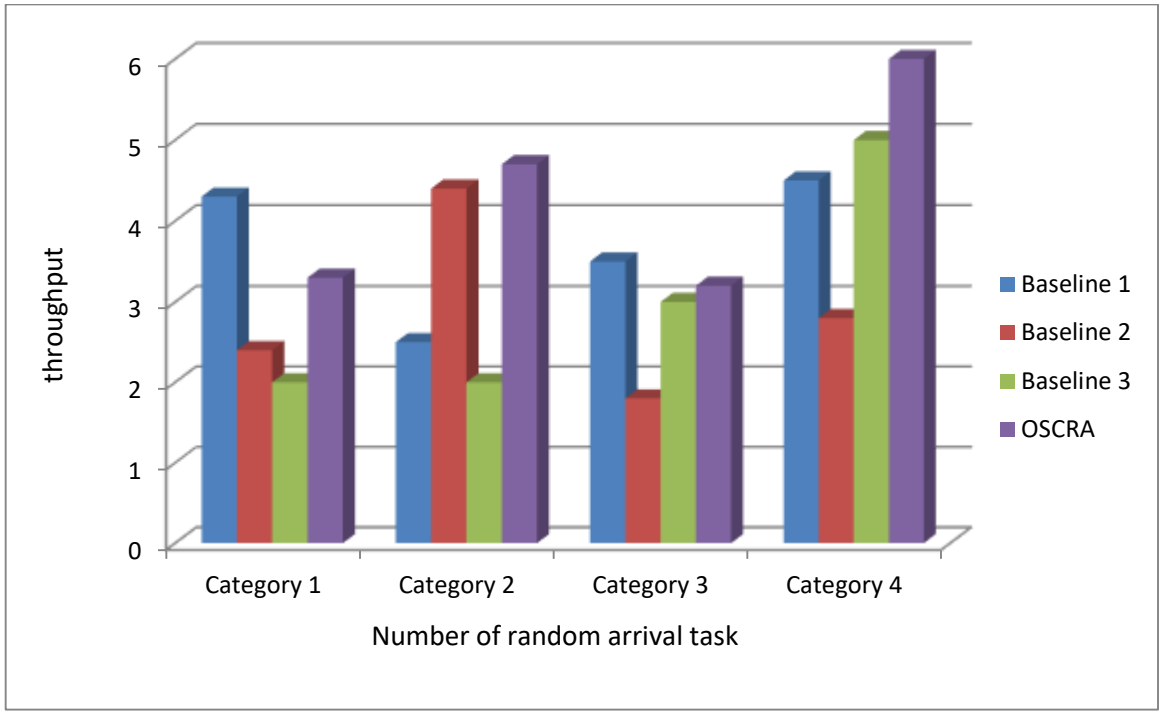

Figure 7. *Throughput comparison*

to previous research, we increase the task failure ratio. As of right now, baseline methods only take into account the first answer. As a result, the suggested dynamic OSCRA approach improves costs and meets deadlines while being effective in a dynamic setting.

4. Conclusion

The cost-effective job scheduling issue in preempting fog servers is examined in this work. Under time and failure-aware limitations, the suggested algorithm framework completed every task with various components. Furthermore, we provide the OSCRA framework, which breaks down task scheduling into several steps: Optimized Scheduling and Cluster-based Resource Allocation. Such as the scheduling, resource matching, and task sequencing steps. According to the performance evaluation, OSCRA performs better than any current scheduling and joint offloading issues in the dynamic environment. According to the results provided, the suggested work successfully implemented industrial automation applications on the cooperative fog cloud network. The outcomes of the experiments demonstrate that the suggested OSCRA strategies can improve server usage. Throughput is increased as service latency and average bootup time are decreased more efficiently. In order to operate IoT applications on hybrid service platforms like Amazon, Azure, and Google jointly, we will concentrate on services composition in the future. Task scheduling and system loading take security and temporary failure into consideration.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Abdelmoneem, R. M., Benslimane, A., & Shaaban, E. (2020). Mobility-aware task scheduling in cloud-fog IoT-based healthcare architectures. *Computer Networks*, *179*, 107348. https://doi.org/10.1016/j.comnet.2020.107348

[2] Alsadie, D. (2024). Advancements in heuristic task scheduling for IoT applications in fog-cloud

computing: Challenges and prospects. *PeerJ Computer Science*, *10*, e2128. https://doi.org/10.7717/peerj-cs.2128

[3] Abualigah, L., & Diabat, A. (2021). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Computing*, *24*(1), 205–223. https://doi.org/10.1007/s10586-020-03176-7

[4] Alsammak, I. L. H., Alomari, M. F., Nasir, I. S., & Itwee, W. H. (2022). A model for blockchain-based privacy-preserving for big data users on the Internet of Things. *Indonesian Journal of Electrical Engineering and Computer Science*, *26*(2), 974–988. https://doi.org/10.11591/ijeecs.v26.i2.pp974-988

[5] Lakhan, A., Mohammed, M. A., Abdulkareem, K. H., Jaber, M. M., Nedoma, J., Martinek, R., & Zmij, P. (2022). Delay optimal schemes for Internet of Things applications in heterogeneous edge cloud computing networks. *Sensors*, *22*(16), 5937. https://doi.org/10.3390/s22165937

[6] Al-Maytami, B. A., Fan, P., Hussain, A., Baker, T., & Liatsis, P. (2019). A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing. *IEEE Access*, *7*, 160916–160926. https://doi.org/10.1109/ACCESS.2019.2951760

[7] Khan, Z. A., Aziz, I. A., Osman, N. A. B., & Ullah, I. (2023). A review on task scheduling techniques in cloud and fog computing: Taxonomy, tools, open issues, challenges, and future directions. *IEEE Access*, *11*, 143417–143445. https://doi.org/10.1109/ACCESS.2023.3308305

[8] Ali, I. M., Sallam, K. M., Moustafa, N., Chakraborty, R., Ryan, M., & Choo, K. K. R. (2020). An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems. *IEEE Transactions on Cloud Computing*, *10*(4), 2294–2308. https://doi.org/10.1109/TCC.2020.3007621

[9] Adhikari, M., Mukherjee, M., & Srirama, S. N. (2019). DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. *IEEE Internet of Things Journal*, *7*(7), 5773–5782. https://doi.org/10.1109/JIOT.2019.2962370

[10] Amoon, M., Bahaa-Eldin, A. M., & El-Bahnasawy, N. A. (2023). Resource allocation strategy in fog computing: Task scheduling in fog computing systems. *Journal of Communication Sciences and Information Technology*, *1*(1), 1–11. https://doi.org/10.21608/jcsit.2023.XXXXX

[11] Huang, Y. M., Hsieh, M. Y., & Usak, M. (2020). A multi-criteria study of decision-making proficiency in student's employability for multidisciplinary curriculums. *Mathematics*, *8*(6), 897. https://doi.org/10.3390/math8060897

[12] Liu, Q., Kosarirad, H., Meisami, S., Alnowibet, K. A., & Hoshyar, A. N. (2023). An optimal scheduling method in IoT-fog-cloud network using combination of Aquila optimizer and African vultures optimization. *Processes*, *11*(4), 1162. https://doi.org/10.3390/pr11041162

[13] Xia, F. (2024). Optimized multiple-attribute group decision-making through employing probabilistic hesitant fuzzy TODIM and EDAS technique and application to teaching quality evaluation of international Chinese course in higher vocational colleges. *Heliyon*, *10*(4), e26616. https://doi.org/10.1016/j.heliyon.2024.e26616

[14] Zhao, X., & Huang, C. (2020). Microservice-based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network. *IEEE Access*, *8*, 56680–56694. https://doi.org/10.1109/ACCESS.2020.2982572

[15] Ehtisham, M., Hassan, M. U., Al-Awady, A. A., Ali, A., Junaid, M., Khan, J., ... & Akram, M. (2024). Internet of Vehicles (IoV)-based task scheduling approach using fuzzy logic technique in fog computing enables vehicular ad hoc network (VANET). *Sensors*, *24*(3), 874. https://doi.org/10.3390/s24030874