IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 14, 2020, accepted March 14, 2020, date of publication March 19, 2020, date of current version March 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2981860

Microservice Based Computational Offloading Framework and Cost Efficient Task Scheduling Algorithm in Heterogeneous Fog Cloud Network

XUEHUA ZHAO[1] AND CHANGCHENG HUANG[2]
[1]School of Digital Media, Shenzhen Institute of Information Technology, Shenzhen 518172, China
[2]Department of Computer Science, Wenzhou University, Wenzhou 325035, China

Corresponding author: Changcheng Huang (cchuang@126.com)

This work was supported in part by the Guangdong Natural Science Foundation under Grant 2018A030313339, in part by the Ministry of Education in China (MOE) Youth Fund Project of Humanities and Social Sciences under Grant 17YJCZH261, in part by the Scientific Research Team Project of Shenzhen Institute of Information Technology under Grant SZIIT2019KJ022, in part by the National Natural Science Foundation of China under Grant 71803136 and Grant 61471133, and in part by the Key Research and Development Program Projects in Zhejiang Province under Grant 2019C01041.

ABSTRACT Nowadays, the usage of the fog cloud-based Internet of Things (IoT) applications among users has been growing progressively. These applications may be E-Transport, E-Healthcare, Augmented Reality, and 3D-Game. Generally, contemporary cloud frameworks offer services based on virtual machines. These frameworks incurred with following issues such as long boot-time, overhead and unnecessary cost to run the IoT applications. We propose a new Microservice container fog system (MSCFS) based framework to run the mobility and delay-sensitive applications with minimum cost. The study the cost-efficient task scheduling problem in the heterogeneous fog servers. Furthermore, the study introduces the Cost Aware Computational Offloading and task scheduling (CACOTS)framework, which solves the task scheduling into multiple steps. Such as task sequencing step, resource matching step and scheduling step. The experimental results prove that the proposed MSCFS and CACOTS schemes can enhance server utilization. As decrease the services latency and average services bootup time more effectively, and minimize costs.

INDEX TERMS Task scheduling, fog servers, boot-up time, microservices, container.

I. INTRODUCTION

Recently, explosive growth in smart devices (e.g., mobile phones, sensors, and tablets), and the Internet of Things (IoT) based smart applications have been growing progressive [1]. The applications may be E-Healthcare, Augmented Reality, 3D-Gaming, and E-Transport. However, still smart devices face resource limitation issue (e.g., insufficient battery power, poor CPU speed, and inadequate bandwidth utilization) [2]. It is hard to execute the applications as mentioned earlier locally on smart devices [3]. Mobile Cloud Computing (MCC) is an assuring clarification to alleviate the resource-constraint issue of smart devices via computation offloading method [3]. Computation offloading allows smart devices to offload compute-intensive workloads to the rich resource cloud for execution. Furthermore, MCC classified into many types of paradigms such as Edge Computing (EC), Fog Computing (FC), and cloudlet. FC is an emerging paradigm introduced for IoT applications with small end to end latency and placed at the edge of the radio access network [4]. Many studies [5]–[8] introduced computational offloading frameworks for resource constraint devices in the literature. The main goal was to improve smart device battery life, enhance the performance of applications at the resource-constraint devices. These computational offloading frameworks offered virtual machine cloud services to the user applications. However, the bootup time of services at virtual machines is about 28 seconds and inter-process between virtual machines incurred with heavyweight overheads. These limitations cannot meet the requirements of fine-grained lightweight delay-sensitive IoT applications during offloading.

Motivation: Smart devices based Internet of Things applications are demanding by the users and growing progressively. These applications are composed of lightweight operations and required thin services to execute them with minimum delay. Another hand, existing MCC paradigm exploited heavyweight virtual machine to offer assistance to

The associate editor coordinating the review of this manuscript and approving it for publication was Amir Masoud Rahmani.

the user applications. These services are paid based on as you go, pay model, therefore, cost, interactivity, and mobility are the key challenges in the existing MCC paradigms. Furthermore, the IoT application, such as E-Business (e.g., Ticket application), requires many different services from different vendors to meet the requirements. Therefore, cost-efficient resource scheduling is another critical challenge in the MCC for IoT applications.

This paper considers the cost-efficient task scheduling problem in heterogeneous fog cloud network for IoT applications. The goal is to minimize the costs of the services to user applications. These costs determined by the communication cost and computation cost during offloading and scheduling. Every IoT application has a set of independent fine-grained tasks. The fine-grained means, each task has its own data and attributes and run independently. Each task has a set vector attribute such as data size, require CPU instruction for execution, and deadline. We consider the various fog cloud servers that are distinct by their speed and prices. In this paper, the following questions are taking into consideration during task scheduling. (i) How to choose an ideal and cost-efficient fog server for each task, that meets the requirements of the task. (ii) How to sequence offloaded tasks in a cost-efficient way before task scheduling. (iii) How to do optimal task scheduling which reduces the costs of tasks and execute them under their deadlines.

Summary, the paper makes the following contributions.

- Computational offloading is a method between resource constraint devices and the rich resource fog computing model. We propose a novel microservices container-based fog system (MSCFS). It improves heavyweight virtual machines based frameworks with the docker container at the resource level. MSCFS gains lower boot-up time, less overhead among services.
- The paper considers a set of different fog cloud servers with different attributes. However, each task of the application has different vector set QoS requirements. To meet the demand of a task, we require to choose an optimal fog server, among others. We propose the resource matching algorithm which matches each server to the task requirements.
- In our problem, tasks are arrived randomly to the system and followed by the Poisson process. The offloaded tasks are submitted randomly in a set without any sequence. So we need to sequence these tasks first. A three rule-based task sequencing method is employed. The given deadline, size and slack time are critical to sequence the task; therefore, we develop the three rules to sort the submitted responsibilities concerning three features. These rules are the Earliest Due Date (EDD), Lateness Time First (LTF), and Shortest Size First (SSF).
- Schedule, a set of fine-grained tasks onto a network of heterogeneous fog servers. With communication time and computation costs, the scheduling becomes a challenging job. To deal with the problem, We propose a cost-efficient task scheduling. The derived scheme iteratively allocates the tasks onto servers to reduces overall costs.

The rest of the paper is organized as follows. Section 2 shows the existing studies related to the task scheduling problem. Section 3 describes and formalizes the problem under study. Section 4 describes the proposed algorithms CATSA and their components. Section 5 evaluates the performance. Section 6 is the conclusion.

II. RELATED WORK

With the explosive growth of the Internet of Things in practice, computational offloading has gained a lot of popularity among users. Computation offloading allows resource-constrained devices offload resource-intensive tasks of applications to the cloud servers for the execution. This way, the battery power of the device could be improved, as well as application performance could be improved. Many efforts have made to improve the device and application performances by different studies.

The paper [5] presented the CloneCloud framework, which goals at increasing the battery life and execution on the mobile phone. The study exploited offloading method to offload compute-intensive tasks to centric cloud servers. The paper [6] presented a computational offloading framework that focuses on energy saving on mobile phones when execute the compute-intensive applications on it. MAUI is a dynamic computational offloading framework, which exploited different profiling technologies. The goal was to make an offloading decision whether a task runs locally or offload to the server at the runtime of the application. ThinkAir [7] presented a computational offloading framework that allows smartphone applications to offload their workload to the cloud. The framework uses the idea of mobile phone virtualization at the centric cloud and gives method level computation offloading. In addition, some meta-heuristic algorithms are also applied to scheduling problems [8], [9].

Computation offloading to the centric cloud is not evermore a resolution, because of the high wireless access network latencies. Therefore the centric cloud resources to be brought proximity to the mobile-user in the design of cloudlets. Satyanarayanan *et al.* imply in [10] a VM based cloudlet framework. Cloudlets are virtual-machine based on provider scalability, mobility, and elasticity. They are placed in single-hop proximity to mobile phones. The Rattrap [11] presents Container Android Cloud-based framework for smartphone computation offloading that replaces virtual machines with containers. The goal was to decrease the boot time of the monolithic services in the centric cloud environment. However, the aforementioned computational offloading framework cannot meet the requirements of fine-grained IoT applications.

The paper [12] investigates the offloaded tasks of mobile devices at heterogeneous cloudlets while improving the total cost-efficiency. The communication time and computation time, along with costs under deadline, are taken into consideration by the study. Energy-efficient offloading and

cost-efficient task scheduling type problems investigated in these works [13]–[15]. They focused on improving battery energy via computational offloading to cloud computing. Resource cost includes storage and computing model reduced via task scheduling inhomogeneous cloud environment. The studies [16], [17] were investigated by resource models in the mobile cloud system for mobile applications. Whereas, the different pricing models discussed in these studies. For instance, on-demand, on-revered and spot instance. The goal was to decrease the renting costs of resources and execute all applications under their deadlines. These studies [18]–[20] considered the real-time and cost-efficient task scheduling problem to minimize application costs. The main goal is to enhance the performance for cloud service platforms by reducing uncertainty propagation in scheduling workflow applications that have both uncertain task execution time and data transference time. Additionally, and these costs include the communication cost and computation cost during scheduling in the mobile cloud system.

To the best of our knowledge, microservices containers based fog cloud framework for fine-grained delay-sensitive has not been suggested. We propose a novel MSCFS framework, which execute all tasks of applications with minimum costs. To ensure the Quality of Service (QoS), we propose the cost-aware computational offloading task scheduling (CACOTS) framework. The CACOTS provide the applications QoS, minimize the cost of the resources and execute them under their deadlines [21].

III. PROBLEM DESCRIPTION

We study, the cost-efficient task scheduling problem for IoT applications in the heterogeneous fog cloud servers. The costs determined by the communication cost and computation cost during offloading and scheduling at the MSCFS. The goal of the paper is to reduce applications costs at the MSCFS during processing at the heterogeneous servers. The proposed architecture MSCFS describes in detail in the next subsection.

A. PROPOSED MICROSERVICES FOG-CLOUD ARCHITECTURE

The proposed microservices container-based fog system (MSCFS) consists of a client application layer, the control layer, and the resource layer, as shown in Figure 1. Generally, the client application layer generating offloading tasks of different IoT applications and randomly arrive at the control layer. Whereas, the control layers consists of four modules to process the offload tasks at the various resource in the resource layer. The Fog Cloud Agent (FCA) is orchestrator, which is accountable for managing and handling the tasks with assists of task sequence module and scheduling module. FCA is a centralized controller, and it exists between user applications and system resources. FCA monitor and collects data (for example, metrics, configuration information, and logs) from entities. These objects reside on hosts or virtual hosts in a fog system network. These are Monitoring System,

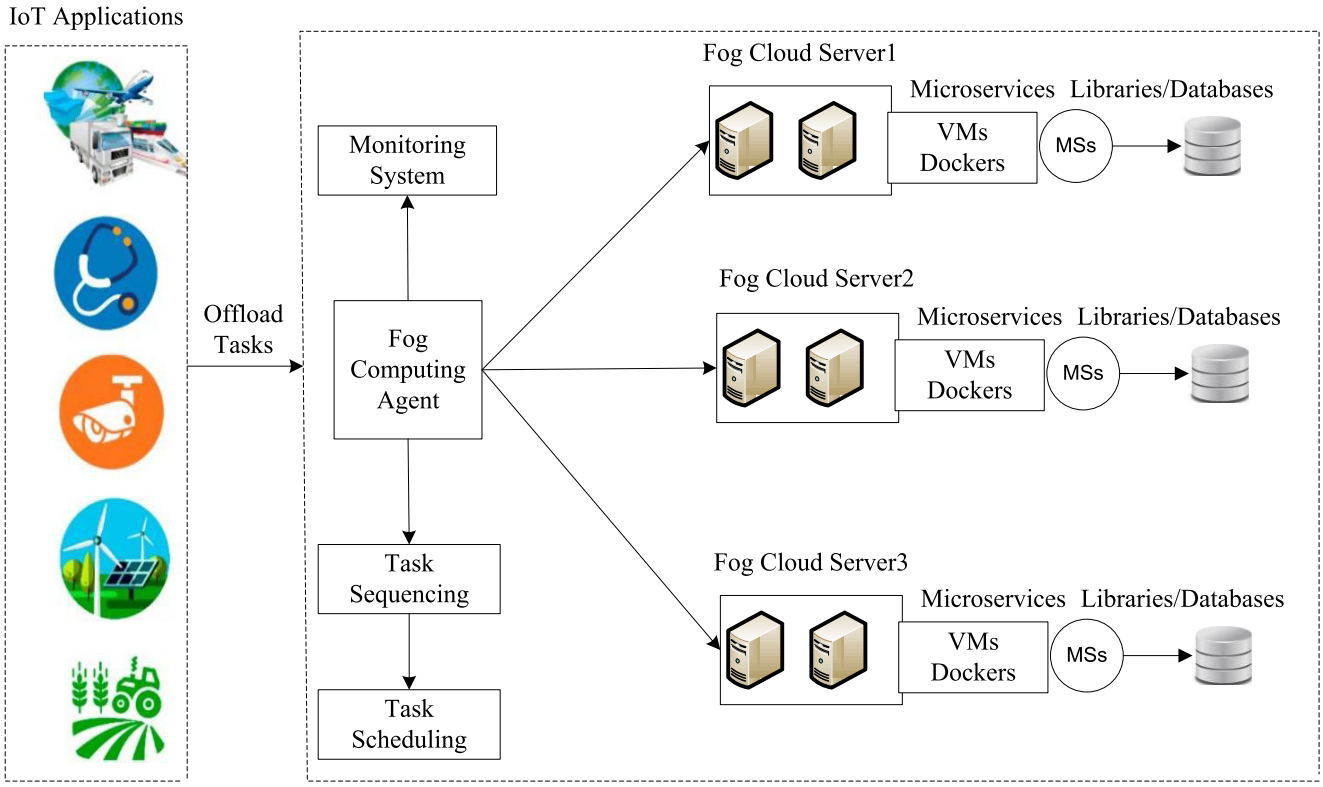


FIGURE 1. Proposed CACOTS system architecture.

Task Sequencing, and Task Scheduling. FCA utilizes all components together to execute requested workloads and measure the performances of the resources.

Initially, we arrange the offloaded tasks into the topological order by the known sorting algorithm [19]. Furthermore, each task has a set of vector attributes such as data size, required CPU instruction, and deadline. To meet the requirement of tasks, we proposed task sequence rules with a different method to ensure sorted algorithm to be scheduled with minimizing cost under their deadlines. Monitoring system manages lookup-table, which includes the list applications tasks, status of available resources. The lookup-table updates after the event occur, such as completion of a task, the resource is free after getting the job done. Due to swapping the tasks between servers, initial task scheduling will further improve costs function of the resource by swapping tasks among different fog servers.

The resource layer is composed of heterogeneous fog servers with a limited set of homogeneous virtual machines. We consider virtual machine at the top layer of resources, internally we deployed docker-engine which add/remove container microservices for the fine-grained offloaded tasks. Each container holds one business microservice for each fine-grained tasks, and it can communicate with microservice via REST API.

B. TASK CHARACTERIZATION

The IoT application made up of independent fine-grained tasks [22]. The fine-grained means here, all tasks have their data in the application. These tasks run independently and require different resources for execution. Whereas, every task has vector attributes such as data size, required CPU and deadline before task scheduling at the fog system. These tasks arrive at the fog system based on the random method. The preemption state during scheduling is not permitted.

C. MOBILITY MANAGEMENT

At the edge of the wireless network, we integrated all of the fog cloud services. Mobility module in FCA allows service networks to locate the attachment point of a mobile subscriber to deliver data packets (i.e. location management) and to maintain the connection of a mobile subscriber as it continues to change its attachment point (i.e. handoff management). This segment addresses the problems and functionalities of those operations. Handoff management is the mechanism through which a mobile node triggers its connection while moving from one access point to another. A handover process involves three phases. Next, either the mobile device, or a network agent, or the changing network conditions cause the activation of handoff. The second stage is for a new connection process, where the network needs to find new handoff connection tools and perform additional routing operations. Finally, data-flow control must ensure data delivery from the old connection path to the original connection path according to the QoS support granted upon.

D. RESOURCE CHARACTERIZATION

In the fog cloud paradigm, the containers as a lightweight methodology to virtualise applications have newly been thriving, especially to run IoT applications in the fog cloud [23]. Typically, the administration of groups of containers grows crucial, and the orchestration of the development and deployment grows a fundamental problem. Microservices are small autonomous services that communicate over well-defined APIs. Small, self-contained teams own these services in the MSCFS. We consider heterogeneous fog server, at the top of the layer we exploited virtual machines. However, internally, we deployed the docker engine, which adds and remove the containers microservice easily. A heterogeneous cluster of fog servers contains processors and devices with different bandwidth and computational capabilities, runtime, virtual machines (e.g., container-microservices) and costs.

E. MICROSERVICES RUNTIME IN MSCFS

Microservices are a technique of runtime software development a variation of the service-oriented structured architecture model that arranges an application as a series of loosely coupled services. Services are fine-grained in a microservice architecture, and the protocols are lightweight. In MSCFS, each fog server made of a single virtual machine (e.g., on-demand). Whereas, each virtual machine can support many container-microservices at a time. Each container-microservice can run one task at a time. For instance, requested doctor searching computational task of healthcare application requires a searching service in fog server. FCA assigns cost-efficient microservice to those as mentioned earlier requested computational tasks to run with the efficient result. The microservice (e.g., doctor searching service) is a business goal which holds its resources and libraries to run any task. Therefore, it is an effective way to handle fine-grained tasks of application during offloading in the system.

F. PROBLEM FORMULATION

In the fog cloud system, we assume a set of delay-sensitive tasks $T = \{t_1, t_2, \ldots, N\}$ to be offloaded and schedule to the fog servers. Each task t_i holds different attributes and denoted by $t_i = \{W_i, data_i, d_i, s_i\}$. W_i denotes the workload of a task, $data_i$ shows the data size of a task during transmission, d_i illustrates task deadline of a task, and s_i demonstrates storage requirements of a task.

We assumed that, the fog cloud system consists of M heterogeneous fog servers, i.e., $F = \{f_1, f_2 \ldots, M\}$. Each fog server f_j has the following attributes, i.e., $f_j = \{Bw_j, \zeta_j, S_j, V_j\}$. Whereas, Bw_j shows the bandwidth between centric fog cloud agent and fog cloud server during task offloading. ζ_j illustrates the computing rate of the j^{th} fog server, S_j demonstrates the total capacity (e.g., storage) of the fog server j in the system. V_j denotes the number of deployed virtual machine docker for microservices with same capability in the fog server j. Each V_j made up of different containers and to be

executed many microservices for the offloaded tasks. Each microservice can run autonomously with its libraries and database during task execution.

Moreover, we show b_{ij} is a bandwidth demand of a task t_i when it schedules at fog server j. C_j denotes resources (i.e., RAM, Storage, Bandwidth, etc) cost of server j when it executes the offloaded task t_i and state remains on i.e., $y_j = 1$. The paper has limited page space. Therefore the remaining notations are listed in Table 1. We describe a binary variable x_{ij} to show either a task t_i to be scheduled to a fog server f_j or not as

$$x_{ij} = \begin{cases} 1, & t_i \leftarrow f_j \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

TABLE 1. Mathematical notation.

Notation	Description
N	A set of Tasks of all IoT applications
M	The Number of fog cloud servers
x_{ij}	The task i is assigned to fog cloud f_j
y_j	Denotes whether to f_j server is on or off
t_i	The i^{th} task of a any application
f_j	The j^{th} fog server in the system
s_i	The storage demand of a task t_i
d_i	The deadline constraint of a task t_i
W_i	The computation data/workload of a task t_i
$data_i$	Data size of a task t_i during transmission
b_{ij}	Bandwidth demand when a task t_i schedule to f_j
S_j	The capacity of the j^{th} fog cloud server
V_j	The number of virtual machine docker deployed at f_j
ζ_j	The computing speed of each virtual machine under f_j
Bw_j	Bandwidth between fog computing agent and f_j
C_j	The cost of j^{th} fog server in the system

It is similar to the assignment problem, each task is exactly assigned to the one f_j, and similarly the fog server exactly schedule one task at a time. We denote assignment of task t_i to the fog server j as follows

$$\sum_{j=1}^{M} x_{ij} = 1. \tag{2}$$

Each fog server has limited resource capability, so that the offloaded task execution requirement must not be exceed capability of fog server. We show this expression as follows

$$\sum_{i=1}^{N} x_{ij} s_i \leq S_j. \tag{3}$$

Due to the resource-constraint of fog server j, it has limited virtual machine docker for offering microservice to each task. Thus, the requested tasks workload must be less deployed virtual machine capacity. This expression is described as

$$\sum_{i=1}^{N} x_{ij} w_i \leq V_j. \tag{4}$$

Each offloaded task to be scheduled to any optimal server j, that is decided by the fog cloud agent where to schedule task t_i. We measure the task execution to server j as

$$T_i^e = \sum_{j=1}^{M} x_{ij} \frac{w_i}{R_j}. \tag{5}$$

Since, a task t_i to be offloaded to the fog server for processing, therefore a task gains some extra communication during offloading and get back its by fog server as

$$RTT = \left(\frac{data_i^{in}}{Bw_{ij}^{up}} + \frac{data_i^{out}}{Bw_{ij}^{down}} \right). \tag{6}$$

$data_i^{in}$ shows input size of a task t_i and $data_i^{out}$ illustrates output of a task t_i after being processed at server j. Whereas, bw_{ij}^{up} and bw_{ij}^{down} show uplink and down link rate of bandwidth between application and server j during offloading and get back its result. RTT is the round-trip time which is equal to data sending and receiving time between application and server j for all tasks. Therefore, the bandwidth requirement of each task is measured as follows

$$x_{ij} \left(RTT + T_i^e \right) \leq d_i. \tag{7}$$

The inequality bandwidth of a task t_i can be obtained as

$$bw_{ij} > \frac{data_i}{d_i - \frac{W_i}{\zeta_j}} \tag{8}$$

To minimize the cost of each fog server j, we make sure all tasks are finished before their respective deadlines constraint. Then, the required bandwidth between application and fog server can be measured as

$$bw_{ij} = \frac{data_i}{d_i - \frac{W_i}{\zeta_j}} \tag{9}$$

Since, each fog server j has limited bandwidth, therefore the all scheduled tasks at server j must be less than its bandwidth constraint as

$$\sum_{i=1}^{N} x_{ij} Bw_{ij} \leq Bw_j. \tag{10}$$

Fog Computing Agent (FCA) is a orchestrator which communicates with each fog server and monitors them in the given interval. The cost of each fog server depends on two main elements such as On state and resources required to execute microservices for each offloaded task. The cost of each fog server not only depends on its On state, but FCA only charge user's only for requested functions rather than entire server cost. We use a binary variable y_j to denote show the on/off state of fog server j as

$$y_j = \begin{cases} 1, & f_j \leftarrow on \\ 0, & \text{off}, \end{cases} \tag{11}$$

G. COST MODEL

The price model is taken into consideration based on-demand method which is similar to the prevailing business enterprise application.

$$C_j = \mu_j . x_{ij} . T_i^e . \tag{12}$$

μ_j the unit price of each fog server as shown in Table 2. The resource constraint cost optimization problem of each fog server is formulated as

$$\min Z = \sum_{i=1}^{N} \sum_{j=1}^{M} y_j C_j \quad \forall i \in N. \tag{13}$$

$$\text{Subject to } \min Z = \sum_{i=1}^{N} \sum_{j=1}^{M} y_j x_{ij} . C_j \quad \forall i \in N. \tag{14}$$

$$T_{j,0} = 0, \quad \forall \{j = 1, \ldots, M\}, \tag{15}$$

$$T_{kj} = T_{kj} - 1 + \sum_{k=1}^{N} x_{kj} T_k^e \quad \forall \{j = 1, \ldots, M\}, \tag{16}$$

$$T_i^e = \sum_{j=1}^{M} x_{ij} \times \frac{W_i}{\zeta_j} \quad \forall \{i = 1, \ldots, N\}, \tag{17}$$

$$F_i = \sum_{j=1}^{M} T_{kj} x_{ij} \quad \forall \{i = 1, \ldots, N\}, \tag{18}$$

$$F_i + RTT \leq d_i, \tag{19}$$

$$\sum_{j=1}^{M} x_{ij} = 1 \quad \forall \{i = 1, \ldots, N\}, \tag{20}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall \{j = 1, \ldots, M\}, \tag{21}$$

$$\sum_{i=1}^{N} x_{ij} s_i \leq S_j \quad \forall j \in 1, 2, \ldots M, \tag{22}$$

$$\sum_{i=1}^{N} x_{ij} \leq V_j \quad \forall j \in 1, 2, \ldots M, \tag{23}$$

$$\sum_{i=1}^{N} x_{ij} b_{ij} \leq Bw_j \quad \forall j \in 1, 2, \ldots M, \tag{24}$$

$$x_{ij} = \{0, 1\}. \tag{25}$$

TABLE 2. Unit price of fog servers.

Fog Servers	On-Demand μ_j			State
	CostPerBw	StorageCost	CPUCost	y_j
f_1	0.5	0.3	2	1/0
f_2	0.7	0.5	4	1/0
f_3	0.9	0.7	6	1/0

IV. PROPOSED ALGORITHM FRAMEWORK CACOTS AND SYSTEM

Cost Aware Computational Offloading and task scheduling (CACOTS)framework consist of multiple components as depicted in Figure 2. The first component is fog server resource matching which matches each task to the different fog servers based on pair-wise approach [24]. Task sequence module is a crucial module which sorts the tasks into different sequence order in a way the optimal scheduling to be performed on the tasks by the scheduler. The sequence t_i to be scheduled to the fog server j if $x_{ij} = 1$, otherwise $x_{ij} = 0$. The process will continue until all requested works execute their execution under their deadlines at the MSCFS. The IoT applications to be processed by multiple components to get their executions. For instance, in Algorithm 1, we define the entire process of an application to achieve the best optimal scheduling in the heterogeneous fog servers.

Algorithm 1 CACOTS

Input : $G \in A, T \in G, \{t_1, t_2, \ldots, T\}, j \in M$
Output: $\min Z$;

```
Call Multi-criteria Decision Scheme;
Call Task Sequencing;
Call Task Scheduling;
PLIST[] ← null;
begin
    foreach (k as M) do
        foreach (G as A) do
            foreach (v_i as T) do
                if (T ≠ ϕ) then
                    Call Resource Matching Algorithm;
                    PList[] ← t_i ← j;
                if (PList[] ≠ ϕ) then
                    Call Task Sequence Component;
            Call Task Scheduling Component;
            Calculate Z* ← PList[] based on equation
            (13);
        Z ← Z*
    return Z;
End-Loop;
```

A. FOG SERVER RESOURCE MATCHING

Since the fog servers are heterogeneous in the cost optimization problem. Therefore, how to select edge servers to process the sequenced tasks is necessary. Recall the goal of our task scheduling strategy is to minimize the costs of the fog computing system, while selects the server with the smallest unit cost μ_j as shown in the equation (27). Where we know different fog servers are also unequal.

$$\mu_j = \frac{C_j}{z_j}. \tag{26}$$

where z_j denote the size of fog server f_j, which can be determined as

$$z_j = \frac{MS_j}{\sum_{i=1}^{M} S_j} + \frac{MV_j}{\sum_{i=1}^{M} V_j} + \frac{MB_j}{\sum_{i=1}^{M} B_j}. \tag{27}$$

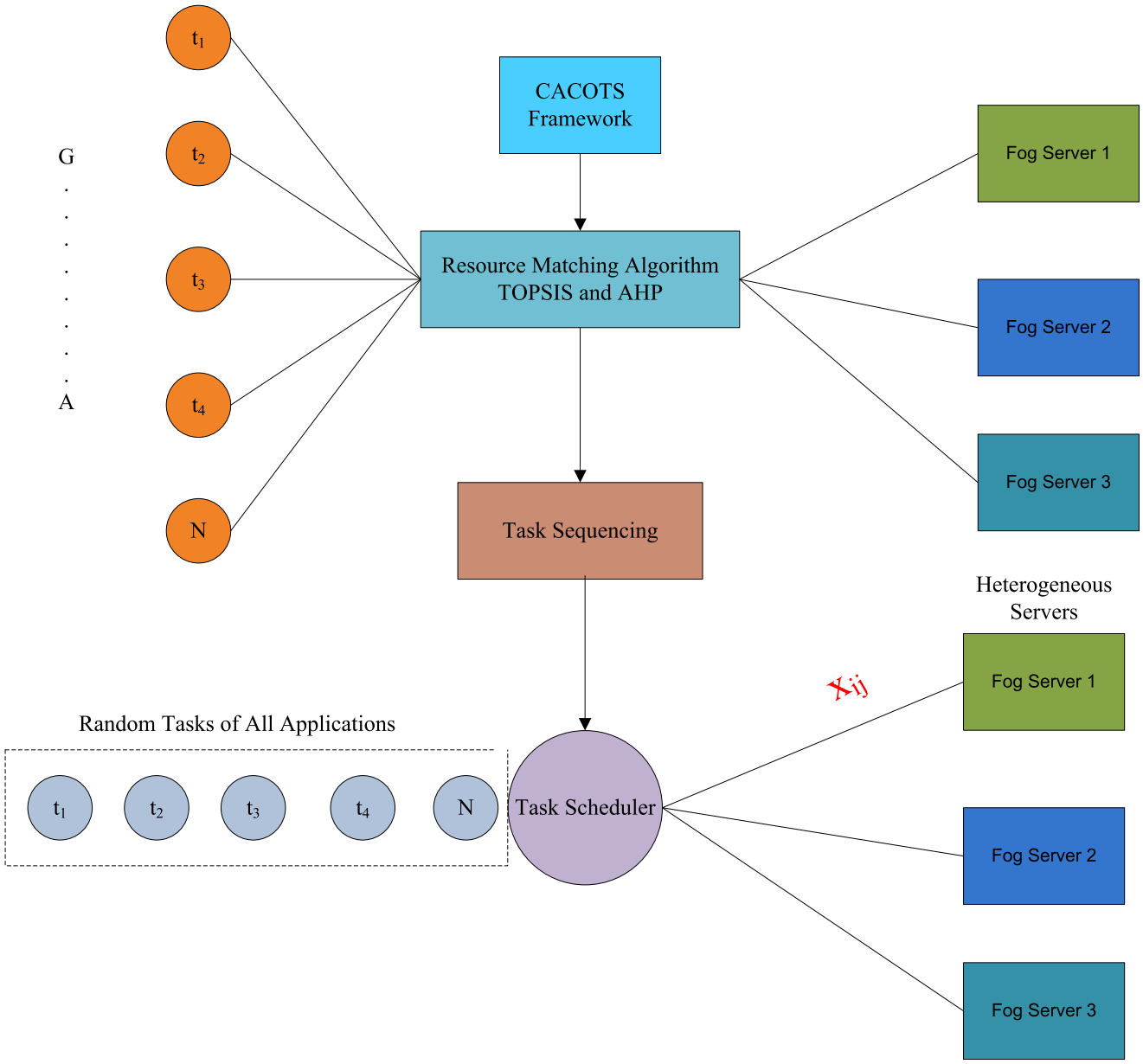


FIGURE 2. CACOTS framework.

We name $\overline{q_{fj}}$ as the remaining available resources of server f_j after scheduled some tasks. We define the largest to be the task that maximizes the dot product h_i [25]. We determine h_i as follows We name $\overline{q_{fj}}$ as the remaining available resources of server f_j after scheduled some tasks. We define the largest to be the task that maximizes the dot product h_i [25]. We determine h_i as follows

$$\begin{aligned} h_i &= \overline{q_{fj}}\overline{P_{t_{ij}}} \\ &= s_i S_j^l q + \mathcal{V}_j^l q + b_{ij} B_j^l q \end{aligned} \tag{28}$$

$$\begin{aligned} q_{f_j} &= (S_j^l q, V_j^l q, B_j^l q) \\ &= P_{f_j} - \sum Pt_{*j} \end{aligned} \tag{29}$$

Pt_{*j} denote the resource requirements of all tasks which are to be scheduled at fog server f_j.

Resource matching is a process of selecting the best and optimal resource for each task in the heterogeneous fog servers. However, each task t_i has vector attributes (e.g., data size, workload, and deadline) and each resource also has vector attributes (e.g., storage, cost, virtual machine capacity, and bandwidth). We exploit Analytic hierarchy processing (AHP) [24] and Technique for Order of Preference by Similarity to Ideal Solution (TOSS) method [26] to make the multi-criteria decision of resource matching. Algorithm 2 takes attributes of tasks, and resources as an input. $PList[]$ is a preference list array which stores match

Algorithm 2 Fog Server Resource Matching

Input : $G \in A, T \in G, t_i \leftarrow \{s_i, W_i, d_i\}$, $k_j \leftarrow \{S_J, V_J, B_J\}$;
Output: $PList[]$

```
1  begin
2      foreach (G   as A) do
3          foreach (T   as G) do
4              if (t_i ↔ k_j = true) then
5                  Apply AHP;
6                  Apply TOPSIS;
7                  Add PList[t_i, k_j]
8              else
9                  Repeat step-4 again;
10     return PList[t_i, k_j]
11 End-Loop;
```

of a task to the fog server k_j based on requirements that to be fulfilled by the resource k_j. Line-4 verifies that the needs of a task t_i are met by fog server, and condition returns true, and we add this matching to the preference list $PList[t_i, k_j]$. The process will repeat until all tasks are matched to the heterogeneous fog servers.

B. TASK SEQUENCING

In our problem, tasks are arrived randomly to the system and followed by the Poisson process. The offloaded tasks are submitted randomly in a set without any sequence. So we need to sequence these tasks first. A three rule-based task sequencing method is employed. The given deadline, size and slack time are critical to sequence the task. Therefore we develop the three rule to sort the submitted responsibilities concerning three features. These are defined as follows:

- Earliest Due Date (EDD): We sort the set of tasks based on their deadline. The small deadline task sort first and the bigger one later. If the deadline is the same the task with the smaller size is ranked with a higher priority. We determine the task lateness in the following way.

$$LTF = d_i - f_i. \tag{30}$$

- Lateness time first (LTF): The tasks are sorts according to the task lateness time. The task which has shortest lateness time, they schedule early.
- Shortest size First (SSF): The task are sequenced based on the size of the task, quickest task arranged first and the bigger one later.

Offloaded tasks randomly inter arrived at fog cloud system. FCA arranged them in the first in first order with task sequencing. The FCA exploits proposed sequence rules in a particular order to get the execution done in a cost-efficient manner. Figure 3 depicted the entire process from offloading to sequencing orders of tasks with different methods. Different task sequencing methods have different scheduling results. Therefore, we will choose an optimal task sequence of offloaded tasks which satisfy the constraint and objective function of the problem.

C. TASK SCHEDULING

After task sequencing and resource matching, we get the preparatory task scheduling method. However, initial task

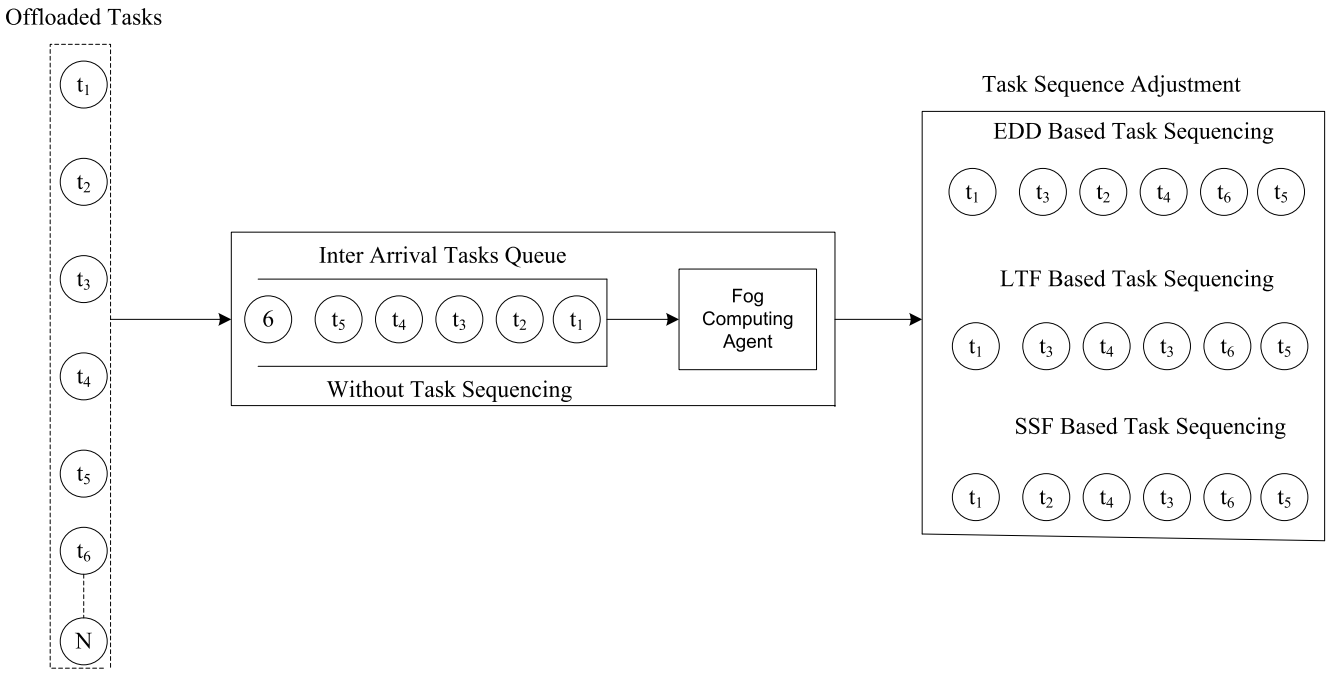


FIGURE 3. The task sequence adjustment of all applications.

scheduling is not the final solution to the IoT application in term of cost. Due to the seasonal changes in network contents and fluctuation in cloud resources, the initial does not remain stable. Therefore, we need to improve it further with new solutions. For instance, we consider there are two tasks t_1, t_2 to be executed onto two heterogeneous cloud servers k_1 and k_2. The resource demand attributes of tasks t_1, t_2 are (data size:10mb, CPU required: 10, and deadline: 20), and (data size:30mb, CPU required: 30, and deadline: 40), separately. On the other hand, the possible resource attributes of servers k_1 and k_2 are (S_j: 15, VM_j: 4, B_j: 10) and (S_j: 20, VM_j: 8, B_j: 20), sequentially.

At the initial scheduling, task t_i will be scheduled to fog server k_1, after that task t_2 to be scheduled to fog server k_2. In this way, the total cost of the application is the aggregate of two fog servers cost. Nevertheless, if all tasks are scheduled to fog server k_2 for executing, the cumulative cost of application is exclusively the cost of fog server k_2. Whereas, this scheduling policy reduce the computation cost of the applications. Figure 4 illustrates that in the solution space, we have more than one solutions. However, we accept some worst solution too during process. However, it is a challenge which optimal solution must be picked by the scheduler, which reduces the total costs of the system.

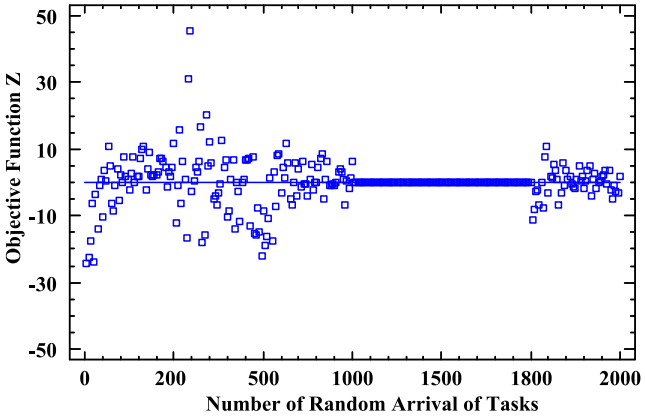


FIGURE 4. Different objective functions.

Whereas the example mentioned above showed, the fog server, which is picked by the scheduler, only has a small cost of resources for execution. Hence, at the initial scheduling, we require to additional optimize the scheduling method to reduce high cost. Whereas, initially selected fog servers will have a substantial cost of available resources. We introduce an optimization task scheduling method to improve resources utilization of the selected fog servers. The goal of the optimization is to reschedule the application tasks in the smallest cost fog servers from the initial task scheduling stage. This way, the task scheduler algorithm can improve the utilization of fog servers with the most expensive cost, and reduce extra cost, which was made by the scheduler at the initial stage. Hence, enhance the application cost, we propose the task scheduling algorithm to determine our optimization problem, as depicted in Algorithm 2. The input of the algorithm is the sequenced set of tasks to be scheduled to the heterogeneous fog servers.

The execution process of the Algorithm 3 is detailed below.

- We declare the different variables inline 1-5. All fog servers are sorted into the descending order according to μ_j and C_j based on equation (27) and (28).
- We read all applications tasks and their requirements along with resource specification inline 7-9.
- If the set T of application G has unpublished tasks, then choose smallest cost fog server μ_j. We pick the

Algorithm 3 Cost-Efficient Task Scheduling

Input : Set of applications: $G \in A$, Set of tasks T, Set of fog servers F
Output: Task Scheduling x_{ij}, state of fog servers y_j

1 Declaration all binary variable x_{ij} to 0;
2 Initial Solution Z;
3 Declaration of fog servers state y_j to 0;
4 Take the vector attributes $\overline{P}_{fj}$;
5 Determine the unit cost μ_j of fog servers $k \in M$;
6 Set of fog servers that are exploited $E = \{\}$;
7 **begin**
8 **foreach** *(G as A)* **do**
9 **foreach** *(t_i as T)* **do**
10 **foreach** *(f_j as F)* **do**
11 **if** *($T \in G \neq \phi$)* **then**
12 Choose smallest cost μ_j;
13 $t_i \leftarrow f_j$;
14 Compute h of all tasks to f_j;
15 Assign task t_i with the largest size of dot product in fog server f_j;
16 $T \leftarrow T \mid \{t_i\} \mid$;
17 Establish $x_{ij} = 1$;
18 Establish $y_j = 1$;
19 $E = E \cup \{f_j\}$;
20 $Z \leftarrow x_{ij}$ optimal assignment;
21 $F = F \cup \{f_j\}$;
22 Get selected fog server f_{g1} and server f_{g2} with small cost unit in E;
23 **if** *($f_{g1} > f_{g2}$ in E)* **then**
24 Swap: $t_i \leftarrow f_{g1}$ to $t_i \leftarrow f_{g1}$;
25 Assign $x_{ig1} = 1$;
26 $Z^* \leftarrow x_{ij}$ optimal assignment;
27 **else if** *($t_i \leftarrow f_{g2} = \phi$)* **then**
28 Set $E \leftarrow E \mid f_{g2} \mid$;
29 Set $y_{fg2} = 0$; Get the newest fog server f_{g2} with small cost in E;
30 End-Assignment;
31 End Application Assignment;
32 End-Loop;

fog server to execute unscheduled tasks based on their resource requirements and the available resources of the fog server. After that, we choose the server f_j with the smallest unit cost in the set of available servers F. If the available resource of the picked fog server can satisfy the resource demands of some tasks in the unscheduled tasks set T, then prefer the most significant task to insert into server f_j. Otherwise, the server f_j is removed by the set of possible servers F is defined by line 10-19.

- We get the last picked fog server f_{g1} and the fog server f_{g2} with the small cost in E. If the available resources in server f_{g1} can satisfy the resource demands of task t_i in the fog server f_{g2}. Hence, we swap the task t_i into server f_{g1} and update the task scheduling variable. When tasks in the fog server f_{g2} are all removed, the fog server f_{g2} will be removed from E, and update the state of the fog server f_{g2}. Then the new server with the small cost is got from E is defined by line 20-27.

D. TIME COMPLEXITY OF CACOTS

We show the time complexity CACOTS framework into different components. (I) Resource Matching: We exploit AHP and TOPSIS methods to match each task to the heterogeneous servers. The time complexity is equal to $O(T \times M)$. T shows the number of tasks arranged for pairwise matching to the M number of resources based on multi-criteria. (II) Task Sequences: We sort all tasks into ascending order of their deadlines, lateness and shortest size with $O(mlogn)$. M is the number of methods which were exploited to sorted the tasks accordingly. And n is the number of sorted tasks during sequences. (III) Task Scheduling: We sort all fog servers in descending order of their μ_j and C_j. Therefore, the time complexity is $O(logM)+N^2$. $O(logM)$ is the time complexity for sorting all fog servers into descending of their pricing in the scheduling process. N^2 shows the time complexity the swapping of tasks between fog server f_{g1} and f_{g2}.

V. PERFORMANCE EVALUATION

To evaluate the performance of the proposed MSCFS and CACOTS framework, we generated the practical results when experiments are conducted on different benchmarks of IoT applications by the system. In this paper, the experimental setup divided into separate parts. (i) MSCFS implementation part, (ii) Components Calibration and Metric parameter. (iii) Comparison of computational offloading framework (iv) Task Scheduling and algorithm comparison part. The simulation parameters described in Table 3. The resource specification of fog servers demonstrated in Table 4. Whereas, Table 5 shows the workload analysis of IoT applications.

A. BASELINE FRAMEWORK AND ALGORITHMIC APPROACHES

The following existing computational offloading framework approaches are taken by experiment into consideration in comparison.

TABLE 3. Simulation parameters.

Simulation Parameters	Values
Windows OS	Windows Docker Engine
Centos 7 Runtime	X86-64-bit AMI
Languages	JAVA, XML, Python
Android Phone	Google Nexus 4, 7, and S
Experiment Repetition	160 times
Simulation Duration	12 hours
Simulation Monitoring	Every 1 hour
Evaluation Method	ANOVA Single and Multi-Factor
Amazon On Demand Service	EC2
Android Operating System	GenyMotion
Application interface	Android interface
Bw_{up}	$100 \sim 1000$Mbps
Bw_{down}	$1000 \sim 100$Mbps

TABLE 4. Fog servers specifications.

Cloud	CORE	MIPS/CORE	VMs	Storage(GB)	Cost-PH
k_1	i3	1000	2	1000	0.5
k_2	i5	3000	4	2000	0.7
k_3	i7	5000	8	4000	0.9
k_4	i9	10000	12	10000	0.05

TABLE 5. Workload analysis of IoT applications.

Workload	$W_{a,i}$(MB)	N	Communication Cost
Healthcare	15.8	825	5G:2.5$
Augmented Reality	24.8	600	4G:1$
E-Transport	38.8	640	3G:0.5$
3D-Game	29.8	750	Cellular:2$

- Baseline 1: We implement the virtual machines based computation offloading framework for the testing. These studies adopted aforementioned framework [12]–[14]. The goal was to offload the entire mobile application to the cloud server.
- Baseline 2: We implement virtual machines dynamic computation offloading framework, which is adopted by these strategies [4]–[6]. The aim is to offload entire applications to heterogeneous servers when there are sufficient resources to fulfil the requirements.

The following existing task scheduling approaches are taken into consideration in comparison.

- Baseline 1: We implement the existing cost-efficient static task scheduling strategies [13], [14] in the experiment part, and test their performance as compared to the proposed scheme in term of application costs.
- Baseline 2: We implement the existing cost-efficient dynamic task scheduling strategies [15], [16] in the experiment part, and test their performance as compared to the proposed scheme in term of application costs.
- Baseline 3: We implement the existing cost-efficient static task scheduling strategies [17], [18] without task scheduling in the experiment part, and test their performance as compared to the proposed scheme in term of application costs.

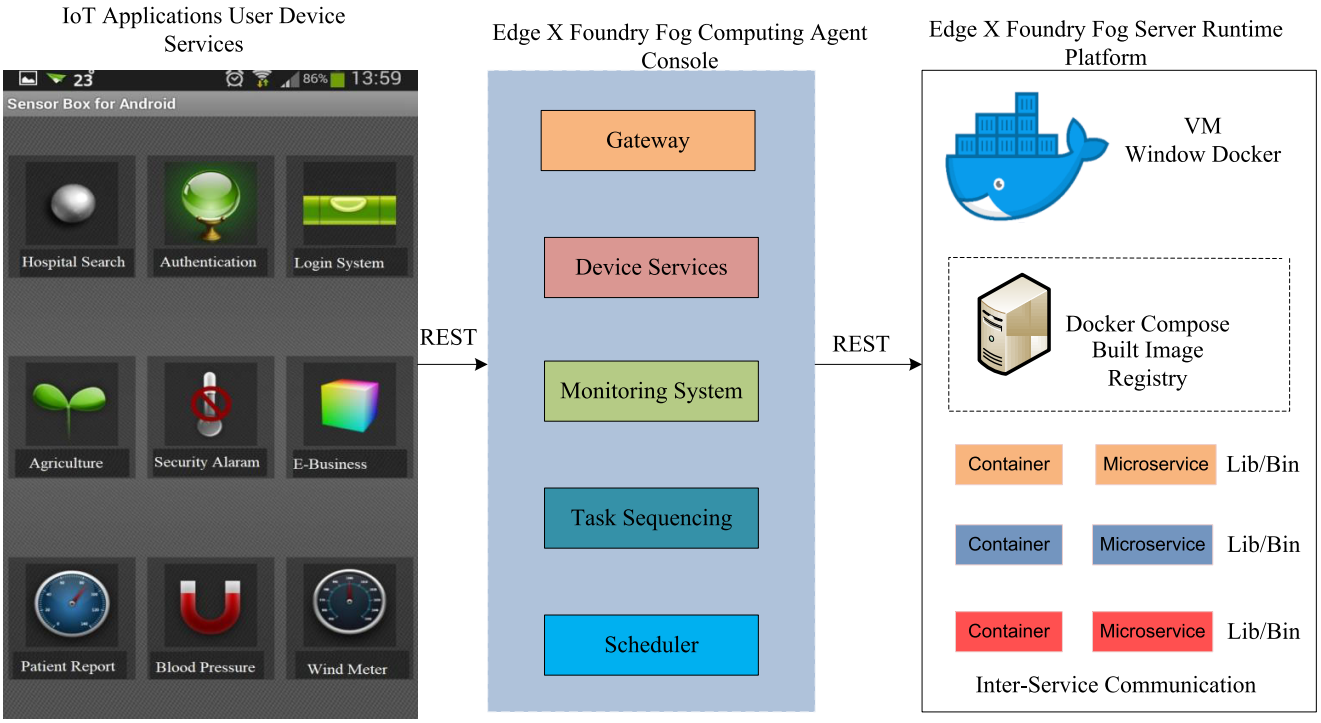


FIGURE 5. MSCFS implementation.

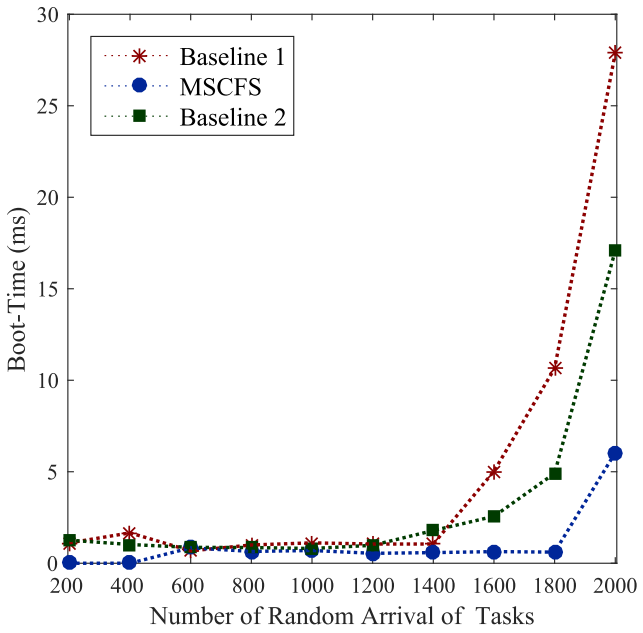


FIGURE 6. Boot-time of microservices.

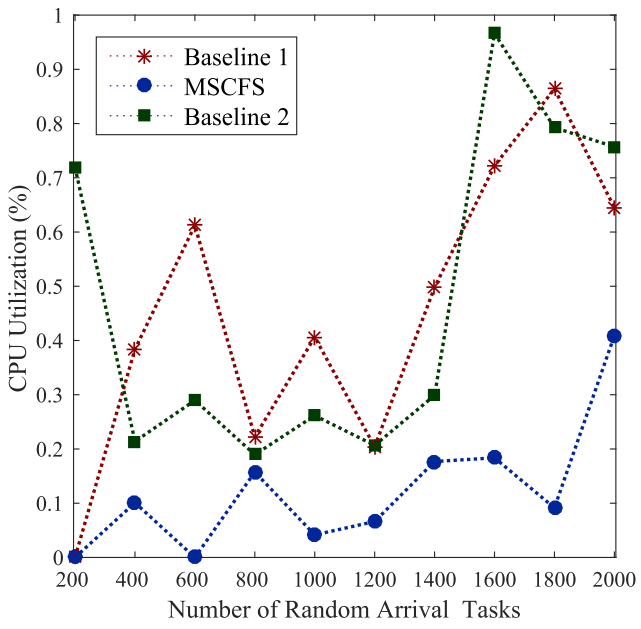


FIGURE 7. CPU utilization of resources.

B. PERFORMANCE METRICS

The component calibration in the paper recommends the experimental method which randomly generates a diverse figure of application tasks which are illustrated in Table 3. In the experiment, there are four different numbers of applications taken into account for testing. For the different forms deadline constraint, the paper sets the deadline for tasks based on the following equation:

$$d_{a,i} = F_{a,i} + \gamma \times F_{a,i} \tag{31}$$

The deadline $d_{a,i}$ of the task is acquired by the earliest finishing duration and a definite proportion of the early finish

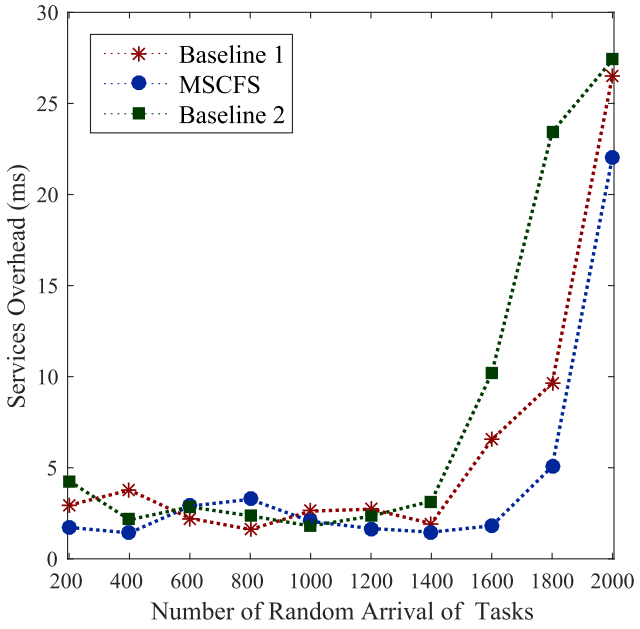


FIGURE 8. Overhead of microservices.

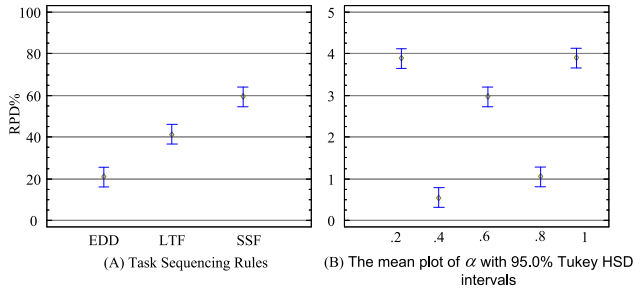


FIGURE 9. The mean plot of α with 95.0% Tukey HSD intervals and The mean plot of random arrival tasks of applications rules with 95.0% Tukey HSD intervals.

time. We used γ to prove the parameter to handle the tightness of task deadline, and their values range exists between .2 and 1, i.e., $\gamma \in \{.2, .4, .6, .8, 1\}$, thus, each task can obtain five unlike size of deadlines, i.e., $D1, D2, D3, D4, D5$. Furthermore, to verify the performance of algorithms in the dissimilar deadline of each task, the paper calculates according to following formula (31). In order to compute the recital of the VFCN, MTOP and CTOS, the paper exploits RPD (Relative Percentage Deviation) statistical analysis. It evaluates the power consumption consumed by dissimilar parameters and framework plus algorithm permutation throughout the parameter of component calibration. The RPD estimation can be as demonstrated in the following Equation (32):

$$RPD(\%) = \frac{F_a^* - F_a}{F_a^*} \times 100\% \tag{32}$$

F_a shows the objective function of the paper.

C. MSCFS IMPLEMENTATION

From the user perspective, we develop the Internet of Things (IoT) applications in Android Studio and integrate GenyMotion emulator for testing [27] as shown in Figure 5. From the resource panorama, we integrated MSCFS based Edge X Foundry open-source platform, i.e., https://docs.edgexfoundry.org/Ch-QuickStart.html. The MSCFS consist of three main layers, such as the user device layer, Fog Control Agent Layer, and resource layer. IoT applications are offloading their tasks randomly to the Fog Computing Agent Console (FCA) via Representational State Transfer (REST) API. Gateway is a standard interface in the FCA, which interprets any request in JavaScript Notation (JSON) format in order console interface read it immediately. Device services tell FCA what kind of service a task requires for execution. Monitoring System manages a look-up table of tasks and resources to ensure the system is stable or not. Task Sequence and Scheduler are methods in the FCA, which sort the tasks into some orders and schedule them onto the heterogeneous fog server for processing. The runtime is an environment where the system will operate. For instance, Java Runtime Virtual (JVM) can run a dot class file of java program efficiently. It is similar to the VM window

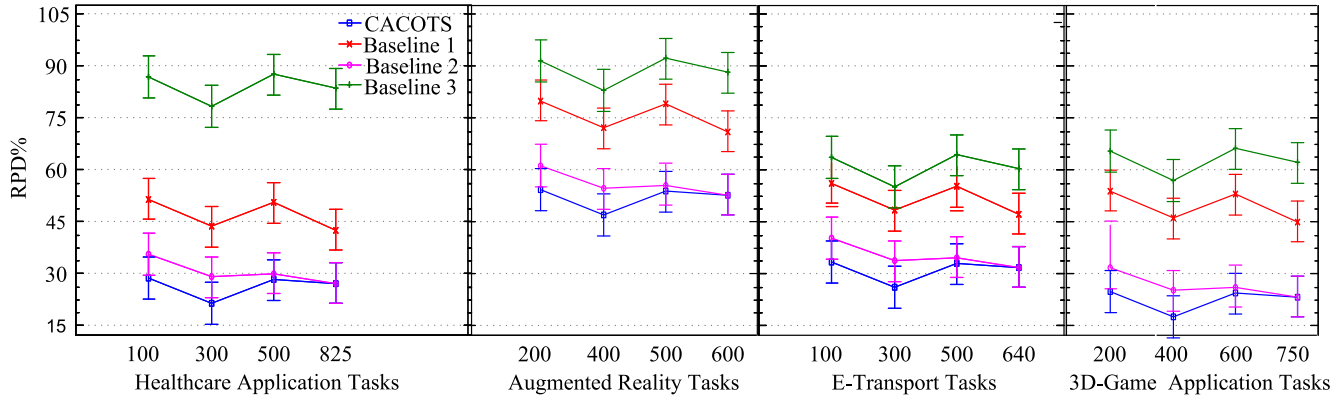


FIGURE 10. Total cost of all tasks.

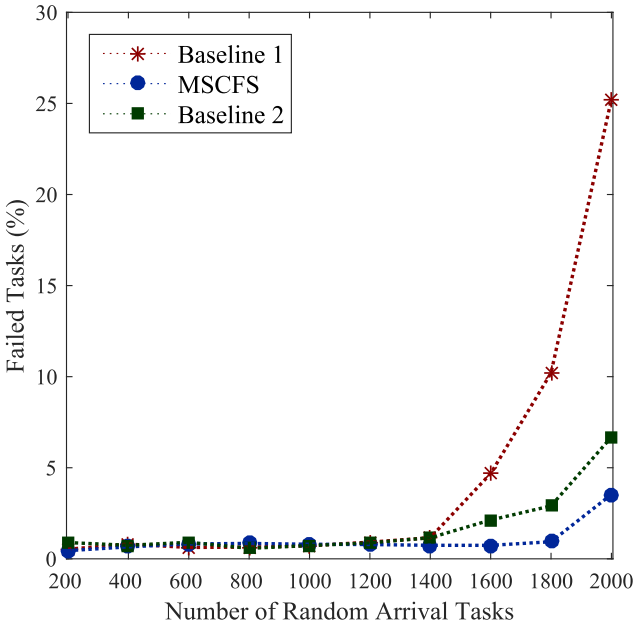


FIGURE 11. Failed tasks during scheduling.

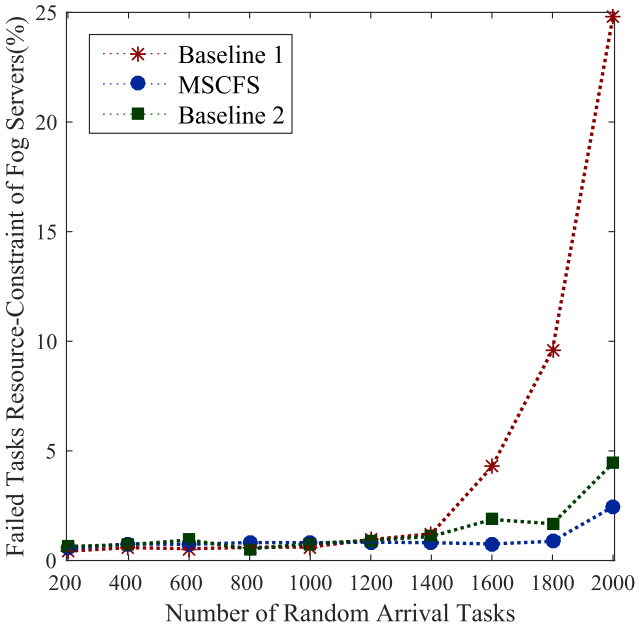


FIGURE 12. Failed tasks due to resource-constrained fog servers.

docker, where container images have created for autonomous microservices. Each container must be registered via registry services to consume it easily. REST API offers inter-service communication among microservices with lower overhead.

D. COMPARISON OF COMPUTATIONAL OFFLOADING FRAMEWORKS

The proposed microservices container fog system based computation offloading has incurred with lower bootup time as compared to existing heavyweight virtual machine-based framework. Another hand, we improved the resource utilization of services in the proposed system. We prove these aspects via experimental and simulation environment. As Figure 6, 7 and 8 show that, our proposed computation offloading improved boot uptime of services, overhead and utilization of the resources. The main reason behind that containers are lightweight as compared to the heavyweight virtual machines when to run the IoT applications during scheduling. Therefore, our computational offloading framework is an efficient platform to execute the delay-sensitive applications.

E. TASK SEQUENCING

The proposed task sequencing rules(i.e., EDD, SPF, and SSTF) are the components calibrated for organizing tasks for the scheduling. Figure 9 (b) demonstrates the mean plot α of proposed task sequence rules with 95.0% Tukey HSD spaces, the RPD significance of the EDD is noticeably lesser than the SPF and SSTF as illustrated in Figure 9 (b). It is accomplished that the task is mapped or scheduled by the task sequence, which is created by the EDD rule incurs to the lower delay fog clouds in a heterogeneous environment. Therefore, we have chosen EDD for the task sequencing component in the Mob-Cloud. In our paper, we exploit the EDD sequence method to compute the task priority. The higher priority is to set to those tasks which have the smallest deadline among others.

F. TASK SCHEDULING

Cost-efficient task scheduling is the key to IoT applications. The applications costs (e.g., communication cost and computation cost) and tasks deadlines are the key issues are taken into consideration during scheduling. We consider the different fog servers for the task scheduling problem. We aim to minimize applications costs and execute them under their deadlines. Figure 10 and 13 illustrate that the proposed scheme CACOTS incur lower applications costs for all applications and run them under their deadlines. We consider the different deadlines, as mentioned in the equation (31). The main reason is that the proposed scheme iteratively improve the solutions from neighbour space until the final optimal solution to be reached. We improve the ratio of task failure in our task scheduling scheme as compared to the existing studies. As an existing baseline approaches only consider the initial solution. Figure 11 and 12 records that after the experiment, the failure ratio of tasks reduces in the proposed scheme as compared to the baseline approach. Hence, the proposed dynamic CACOTS method is efficient in a dynamic situation and improve the costs and execute them under their deadlines.

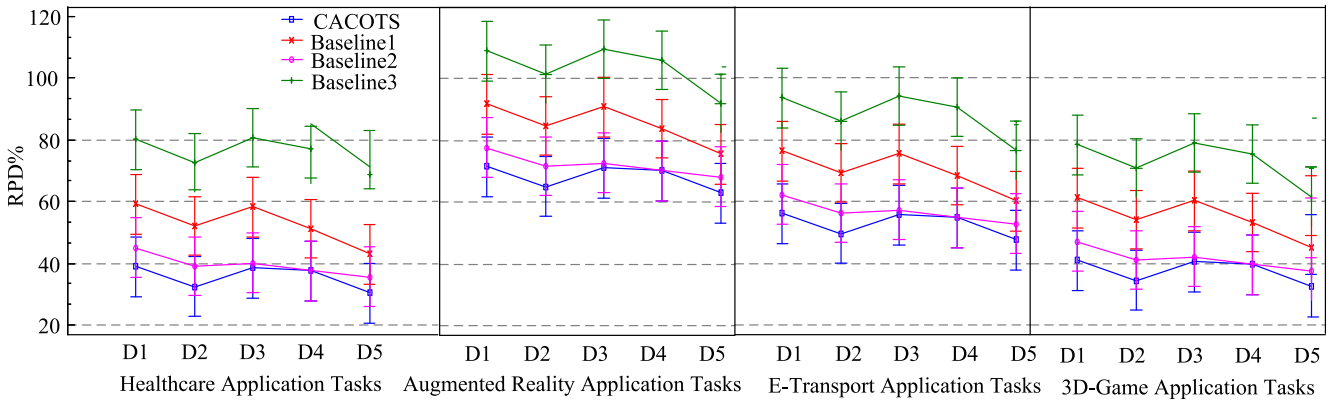


FIGURE 13. Different deadlines of tasks of different applications.

VI. CONCLUSION AND FUTURE WORK

In this paper, we consider the cost-efficient task scheduling problem in the heterogeneous fog servers. We propose a new Microservice container fog system (MSCFS) based framework to run the mobility and delay-sensitive applications with minimum cost. Furthermore, we introduce the Cost Aware Computational Offloading and task scheduling (CACOTS)framework, which solves the task scheduling into multiple steps. Such as task sequencing step, resource matching step and scheduling step. The experimental results prove that the proposed MSCFS and CACOTS schemes can enhance server utilization. As decrease the services latency and average services bootup time more effectively, and minimize costs.

In the future, we will focus on services composition to run the IoT application onto the hybrid service platforms such as Amazon, Azure and Google together. Security and transient failure are taken into account during task scheduling and offloading in the system.

REFERENCES

[1] A. Capponi, C. Fiandrino, C. Franck, U. Sorger, D. Kliazovich, and P. Bouvry, "Assessing performance of Internet of Things-based mobile crowdsensing systems for sensing as a service applications in smart cities," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2016, pp. 456–459. [Online]. Available: https://academic.microsoft.com/paper/2548364277

[2] A. Lakhan and X. Li, "Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks," *Computing*, vol. 102, no. 1, pp. 105–139, Jan. 2020.

[3] N. Fernando, S. W. Loke, and J. W. Rahayu, "Mobile cloud computing," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013. [Online]. Available: https://academic.microsoft.com/paper/2144554277

[4] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya, "An application placement technique for concurrent iot applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, early access, Jan. 15, 2020, doi: 10.1109/TMC.2020.2967041.

[5] L. Vasiliu. (2012). *Clonecloud: Elastic Execution Between Mobile Device and Cloud*. [Online]. Available: https://academic.microsoft.com/paper/2339162612

[6] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 49–62. [Online]. Available: https://academic.microsoft.com/paper/2101788345

[7] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953. [Online]. Available: https://academic.microsoft.com/paper/2088692353

[8] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.

[9] I. Attiya, M. Abd Elaziz, and S. Xiong, "Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm," *Comput. Intell. Neurosci.*, vol. 2020, Mar. 2020, Art. no. 3504642.

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009. [Online]. Available: https://academic.microsoft.com/paper/2135099885

[11] S. Wu, C. Niu, J. Rao, H. Jin, and X. Dai, "Container-based cloud platform for mobile computation offloading," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2017, pp. 123–132. [Online]. Available: https://academic.microsoft.com/paper/2727307896

[12] A. Lakhan, D. K. Sajnani, M. Tahir, M. Aamir, and R. Lodhi, "Delay sensitive application partitioning and task scheduling in mobile edge cloud prototyping," in *Proc. Int. Conf. 5G Ubiquitous Connectivity*. Springer, 2018, pp. 59–80, doi: 10.1007/978-3-030-22316-8_6.

[13] A. Lakhan and L. Xiaoping, "Energy aware dynamic workflow application partitioning and task scheduling in heterogeneous mobile cloud network," in *Proc. Int. Conf. Cloud Comput., Big Data Blockchain (ICCBB)*, Nov. 2018, pp. 1–8.

[14] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Service (IWQoS)*, Jun. 2017, pp. 1–10. [Online]. Available: https://academic.microsoft.com/paper/2733253862

[15] C.-T. Huynh, T.-D. Nguyen, H.-Q. Nguyen, and E.-N. Huh, "Cost efficient real-time applications scheduling in mobile cloud computing," in *Proc. 5th Symp. Inf. Commun. Technol. (SoICT)*, 2014, pp. 248–255. [Online]. Available: https://academic.microsoft.com/paper/2051459264

[16] X. Ma, S. Zhang, P. Yang, N. Zhang, C. Lin, and X. Shen, "Cost-efficient resource provisioning in cloud assisted mobile edge computing," in *Proc. GLOBECOM - IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6. [Online]. Available: https://academic.microsoft.com/paper/2784170357

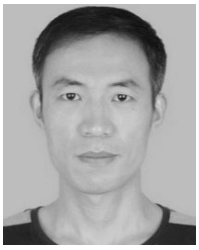
[17] Q.-U.-A. Mastoi, T. Ying Wah, R. Gopal Raj, and A. Lakhan, "A novel cost-efficient framework for critical heartbeat task scheduling using the Internet of medical things in a fog cloud system," *Sensors*, vol. 20, no. 2, p. 441, 2020.

[18] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Trans. Services Comput.*, 2018.

[19] A. R. Mahesar, A. Lakhan, D. K. Sajnani, and I. A. Jamali, "Hybrid delay optimization and workload assignment in mobile edge cloud networks," *Open Access Library J.*, vol. 5, no. 9, pp. 1–12, 2018.

[20] I. A. Jamali, A. Lakhan, A. R. Mahesar, and D. K. Sajnani, "Energy aware task assignment with cost optimization in mobile cloud computing," *Int. J. Commun., Netw. Syst. Sci.*, vol. 11, no. 8, pp. 175–185, 2018.

[21] R. Api. (2015). *Redmine—Defect 10761 Rest API Json Response Can not be Parsed by Jquery*. [Online]. Available: https://academic.microsoft.com/paper/2186518486
[22] A. Lakhan and X. Li, "Mobility and fault aware adaptive task offloading in heterogeneous mobile cloud environments," *ICST Trans. Mobile Commun. Appl.*, vol. 5, no. 16, 2019, Art. no. 159947.
[23] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 677–692, Jul. 2019. [Online]. Available: https://academic.microsoft.com/paper/2613345295
[24] T. L. Saaty, "Decision making with the analytic hierarchy process," *Int. J. Services Sci.*, vol. 1, no. 1, pp. 83–98, 2008. [Online]. Available: https://academic.microsoft.com/paper/2034960640
[25] D. J. Pearce and P. H. J. Kelly, "A dynamic topological sort algorithm for directed acyclic graphs," *J. Experim. Algorithmics*, vol. 11, pp. 1–7, Feb. 2007. [Online]. Available: https://academic.microsoft.com/paper/2084763084
[26] Q. Yong-Hui, "A study for the multi-attribute decision-making method based on topsis," *Technol. Develop. Enterprise*, pp. 89–91, 2006. [Online]. Available: https://academic.microsoft.com/paper/2378721052
[27] Abdullah and L. Xiaoping, "Dynamic partitioning and task scheduling for complex workflow healthcare application in mobile edge cloud architecture," in *Proc. IEEE 4th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2018, pp. 2532–2536.



XUEHUA ZHAO received the Ph.D. degree from the College of Computer Science and Technology, Jilin University, in 2014. He is currently a Lecturer with the School of Digital Media, Shenzhen Institute of Information Technology. His main research interests are related to machine learning and data mining.

CHANGCHENG HUANG is currently an Associate Professor with the School of Computer Science and Artificial Intelligence, Wenzhou University, China, and the Director of the Department of Big Data Technology. He is responsible for major research projects, such as intelligent robot control systems. His current research interests mainly focus on database systems, big data technology, and their applications in industrial, medical, and educational scenes.