IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 December 2023, accepted 2 January 2024, date of publication 5 January 2024,
date of current version 11 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3350171

RESEARCH ARTICLE

Deadline and Energy-Aware Application Module Placement in Fog-Cloud Systems

ABDULELAH ALWABEL[1], **(Member, IEEE), AND CHINMAYA KUMAR SWAIN**[2]
[1]Department of Computer Sciences, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj 91152, Saudi Arabia
[2]Department of Computer Science and Engineering, SRM University, Amaravathi, Andhra Pradesh 522502, India

Corresponding author: Abdulelah Alwabel (a.alwabel@psau.edu.sa)

The authors extend their appreciation to Prince Sattam bin Abdulaziz University for funding this research work through the project number (2023/01/26767).

ABSTRACT Fog computing has emerged as a promising augmentation of cloud computing, positioned at the network's edge, and it is poised to enhance a wide range of Internet of Things (IoT) driven applications. Although fog computing promises to reduce the response time of applications, its omnipresence is subject to the availability and capabilities of the resources in the fog infrastructure. Hence, there is a need of efficiently harness fog infrastructure to execute different IoT applications while meeting their quality of service (QoS) requirements. However, this objective becomes challenging when the applications are decomposed into multiple modules with diverse latency sensitivities. The scatter placement of application modules over distributed fog nodes further intensifies the problem by increasing the overall energy consumption of the fog environment. Therefore, this study proposes a deadline and energy-aware modular application placement policy for fog computing environments. The proposed policy simultaneously prioritizes the placement of critical applications in the fog infrastructure and consolidates the number of active fog nodes for energy management. The performance of the proposed policy was evaluated using *iFogSim* and compared with several contemporary solutions. The experimental results demonstrate that the proposed policy outperforms others in increasing the percentage of QoS-satisfied applications and reducing energy usage in fog computing.

INDEX TERMS Application module placement, placement policy, latency-aware placement, energy-aware placement, task scheduling, resource management, fog computing, cloud computing.

I. INTRODUCTION

Emerging technologies, such as the Internet of Things (IoT), require computation services that are applicable to real-time application processing [1]. Devices such as sensors and mobile devices in IoT ecosystems can generate a large volume of data that can ideally be processed in cloud systems owing to the cost-efficiency and scalability features of cloud computing [2]. However, cloud system may not be suitable platform for processing requests for some IoT applications, and latency sensitive application instances [3].

For handling latency sensitive applications, fog computing model was proposed by Cisco in 2012 [4] and that aims to bring cloud like services closer to end users and/or end devices. This improves the quality of service (QoS) and/or reduce costs such as computational and communication overheads [5]. As different applications have different deadlines to finish their executions, so the applications should be placed in such a manner that all the applications must satisfy the deadline requirements. If any application misses it's deadline then the service provider must pay the violation cost along with the cost incurred due to the execution of applications and energy consumption cost.

Fog nodes are arranged in a network that is ideally placed just one hop away from the data sources. These nodes are placed in close proximity to the user, allowing for local data processing in the nodes rather than transmitting data to distant cloud servers. This reduces the communication overhead and also the latency for communication. Fog computing is considered to be a middle layer between IoT and cloud computing as shown in Figure 1, which better suited for the latency sensitive application placement. However most of the

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

nodes in the fog computing environment are highly dispersed geographically and limited resource capabilities compared to the cloud environment. So all the application modules can not be allocated to the fog nodes. This create challenges for application placement in fog/ cloud environment.

The application placement problem further gets complicated when we consider the micro-service applications. Microservices applications are composed of interdependent modules that can be executed through different computing nodes [6]. The application module provides instructions to generate the corresponding output. The output may be sent to another module as an input, based on the data dependency between the modules as shown in Figure 2. To process the input within a deadline, each module requires a specific amount of resources, such as CPU, memory, and bandwidth. The application module placement policy for allocating an application module to an appropriate resource in fog-cloud systems to meet the application requirements with less energy consumption is a challenging problem which needs an efficient placement policy.

In this study we present a novel application placement policy for the fog-cloud framework where an application is divided into multiple interdependent modules. The applications are ordered and prioritized based on the deadline i.e., earlier deadline of the application higher it's priority for execution. This approach would reduce the complexity of running an entire application by processing the modules in parallel manner whenever possible and load them to various fog/ cloud servers based on the resource and deadline constraints. The proposed approach reduces the latency by placing the modules locally and/ or by processing parallelly on those servers. As effective module placement policy reduces the use of number of servers and that leads to less energy consumption. The major contributions of this study are as follows:

- Two IoT application placement policies are proposed to improve the QoS and reduce the overall power consumption in fog-cloud framework by placing the modules effectively.
- Proposed approaches consider the dynamic arrival of time sensitive applications and present that in the form of Distributed Data Flow (DDF) model. Based on the characteristics of modules the proposed policy decides the order of execution by placing modules to fog-cloud nodes so that the applications meet their deadline and improve the QoS.
- The proposed approaches are evaluated through *iFogSim* simulated fog environment. The experimental results illustrate that the proposed approaches outperform other approaches on various evaluating parameters.

The rest of the paper is organized as follows. Section II summarizes the research work closely related to our work. Section III describe the system model of this work. The proposed mechanism used to solve the problem under consideration is discussed in Section IV. Section V discusses the evaluation mechanism used in this work. Finally Section VI concludes our work with some future directions.

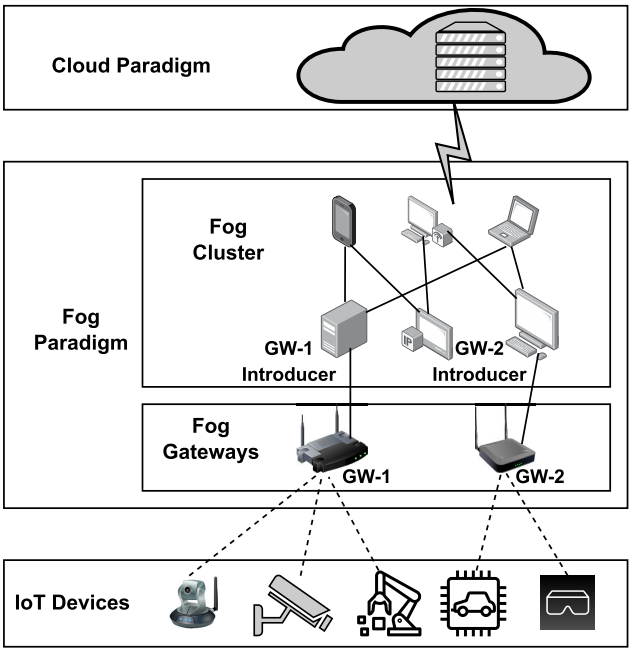

FIGURE 1. IoT with fog-cloud environments.

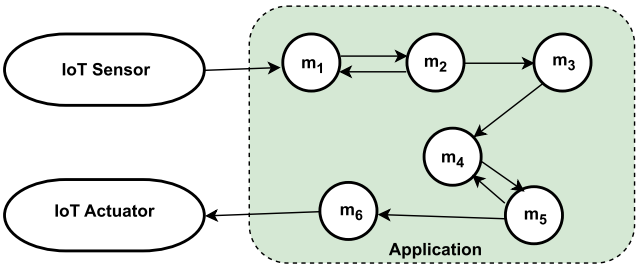

FIGURE 2. Application modules in IoT.

II. RELATED WORK

Task placement and resource management in fog computing are new topics that combine several aspects of cloud computing, mobile computing, and sensor networks [29]. In fog computing, the data generated by sensors, actuators, and other devices must be processed using fog nodes and/or clouds.

The authors of [7] proposed a framework for workload allocation in a fog-cloud system that offers a tradeoff between power consumption and delay issues. The problem of workload allocation can be divided into primary and subproblems. The framework employed a Hungarian algorithm and a Generalized Benders Decomposition (GBD) algorithm to solve the problem, and the results demonstrated that a fog system complements a cloud system. However, the complex nature of the workload and resources was not thoroughly investigated in this work [30].

The study by [8] presented a module-mapping policy for placing IoT applications in fog-cloud environment. Their

TABLE 1. Summary of related work.

Work	Latency	Service Deadline	Energy Management	Resource Utilization	Makespan Time
[7]	✓		✓		
[8]		✓		✓	
[9]			✓	✓	
[10]	✓			✓	
[11]	✓				
[12]	✓		✓	✓	
[13]		✓			
[14]	✓			✓	
[15]	✓	✓		✓	
[16]		✓			
[17]	✓				
[18]	✓	✓		✓	
[19]		✓		✓	
[20]	✓	✓		✓	✓
[21]		✓		✓	
[22]	✓				
[23]	✓	✓		✓	
[24]	✓			✓	
[25]	✓			✓	✓
[26]	✓			✓	✓
[27]	✓			✓	✓
[28]	✓				✓
This Work	✓	✓	✓	✓	✓

work aimed to improve resource utilization. The policy addresses network issues by sorting both network nodes and application modules according to the available capacity and requirements. Then, it maps the modules when the constraint is satisfied. However, this work considered only computing resources, such as the CPU and RAM, to find a proper node. The proposed policy pays little attention to other important factors in IoT ecosystem, such as the latency and availability of specified resources.

In [9], the authors developed an algorithm that determines the optimal solution for resource allocation in fog computing. The allocation problem is presented as a bin-packing penalty-aware problem, where the servers are bins and virtual machines (VMs) are the packs. Each fog device is penalized and rewarded based on the idle energy, maximum frequency, and maximum energy parameters. This method calculates how many VMs could be allocated in t time slots on a server. The VMs were served based on their frequencies. Penalty and reward mechanisms are applied to reduce the energy consumption, which would otherwise increase exponentially.

The work in [10] investigated how user mobility influences the performance of applications by analyzing scheduling problems in fog computing. They studied three different scheduling policies-concurrent, first come first serve (FCFS), and delay priority-to improve the execution time based on the application characteristics. The delay-priority policy outperformed the other policies in terms of delay reduction.

The researchers [11] presented a Fog Service Placement Problem (FSPP) to optimally share resources in fog nodes among IoT services. The FSPP considers latency and deadline requirements during the placement of application modules on fog nodes. In this study, a fog node is characterized by three attributes: CPU, RAM, and storage. The FSPP placement strategy is performed periodically to meet QoS requirements by prioritizing applications according to the response time. The application with the shortest response time was placed on a fog node with sufficient resources.

The authors of [12] proposed a new policy for application module management that aimed at optimizing the number of working nodes. Reducing the number of working nodes leads to reduced power consumption without violating QoS constraints. The proposed management policy was evaluated using iFogsim. This policy outperforms other latency-aware policies by satisfying the latency in service delivery for applications with strict deadlines.

A quality of experience (QoE) application placement policy was developed by [13] to prioritize different application placement requests according to the expectations of users. Furthermore, the policy calculates the capabilities of the fog instances by considering their current status. The policy was evaluated using iFogsim tool. A similar study on QoE was presented in [31]. An optimized placement approach was proposed in [14] for fog computing applications. This approach is based on genetic algorithm (GA). The proposed method minimizes the computational and communication costs of real-time tasks while maintaining efficient resource utilization. Another study that employed (GA) was conducted in [15]. This work presents a cost-aware task scheduling using GA for fog-cloud infrastructure, which improves the cost efficiency of time-sensitive applications.

The authors of [16] presented a real-time task scheduler that considered deadline and frequency constraints to map and process incoming tasks to increase the number of

executed tasks. The proposed approach considers a fog network based on auctions, where tasks rejected by one fog can be accepted at another fog. The proposed work showed that the number of tasks executed increased as a result of using the task-scheduling system.

In [17], the authors proposed throughput-aware application placement in fog computing. The placement approach factored in the computational and communication demands of throughput-oriented applications. The aim of this study is to allocate maximum application modules to one region to reduce communication overhead and thus improve performance. However, the proposed method lacked power-saving capabilities.

A context-aware placement policy for applications in fog environment was presented in [18]. This policy aims to minimize the service delivery time for IoT applications by coordinating IoT device-level contexts with the capacity of the nodes. The context of IoT applications includes the data size and sensing frequency of sensors, which can be utilized to manage network congestion and computational overhead. A novel scheduler was developed in [19] which is a deadline-aware scheduler in a fog-cloud environment that acts as a proxy for processing platforms in the cloud for face recognition applications. The scheduler also focuses on efficiently utilizing the resources.

In [20], the authors proposed an autonomous IoT service placement mechanism based on gray wolf optimization approach to optimize the execution costs in fog environment. This method can improve system performance with regards to service deadline. However, this work pays little attention to energy consumption in fog environment.

PACK is a novel algorithm that can place tasks based on location-allocation problem [21]. PACK can minimize the distances between data sources and fog nodes while considering the capacity constraints for load balancing. Furthermore, PACK considers location and reliability as parameters for ranking fog nodes to allocate tasks to the most appropriate nodes.

The authors of [22] presented a load-balancing approach between fog and cloud systems for IoT platforms. This study is an extension of their previous work [32]. The aim of this study is to improve the performance of IoT application in fog environment. This approach load balanced tasks among fog nodes which helps to fulfill the requirements of scalability and latency, provided that no fog node was overloaded. Tasks are assigned to the cloud nodes provided that all fog nodes are busy processing other tasks or if the latency of tasks allows them to be sent to the clouds. The proposed work demonstrated that the response time decreased as a result of this approach.

In [23], the authors proposed a new mechanism that combines a mixed-integer linear programming formulation with a static and dynamic heuristic. This mechanism is designed to optimize the fairness of energy and bandwidth usage in a fog environment. However, this mechanism does not consider the makespan time required to execute the tasks.

Fog-Care is a prototype designed by [27] to evaluate latency and throughput in a global-wide healthcare application in fog-cloud environment based on blockchain technology. The prototype demonstrated that throughput has increased and maintained an acceptable level of latency.

A study by [33] proposed a novel approach that manages the allocation of fog services as a service function chain [24]. This approach is based on a dynamic planning model that aims to improve latency, resource utilization, and throughput in a fog computing environment. The proposed approach successfully improved the use of resources by creating an SFC queue network. The authors in [25] proposed an HR-Alloc algorithm for executing the big data applications in hybrid infrastructure. They evaluated the algorithm based on cost and load balancing without losing the performance. In [26] authors proposed BurstFlow, a tool for enhancing communication across data sources located at the edges of the Internet and Big Data Stream Processing applications located in cloud infrastructures. The BurstFlow improves the resource utilization and reduces the execution time for multi-cloud deployments. In [27], authors designed and implemented a prototype of a healthcare software called Fog-Care. This model reduces latency, throughput when adopted in wide dispensed locations. Authors in [28] proposed a method that mitigates non-iid data through a FedAvg-BE algorithm that provides Federated Learning with the border entropy evaluation to select quality data from each device.

Table 1 presents an overview of the related works and highlights the difference between this study and present works.

III. SYSTEM MODEL

A. COMPUTING ENVIRONMENT

Figure 1 depicts a three tier computing environment for the proposed deadline and energy-aware modular application placement policy. At the lower level of the computing environment, IoT devices consistently sense data from the external environment. Fog gateways help IoT devices forward the sensed data to the fog infrastructure for further processing. In this study, the fog infrastructure was organized in the form of fog clusters. In a fog cluster, heterogeneous fog nodes communicate with each other through a mesh network [34].

A fog gateway maintains a persistent connection with a fog-cluster node that acts as a cluster introducer. Once an IoT device is configured with a fog gateway and senses data, it notifies its cluster introducer of the QoS requirements, programming model, and data dependencies among the modules of the corresponding IoT application. Later, the cluster introducer perceives the context, such as the resource availability, computational overhead, and energy profile of other cluster nodes, using the mesh connection. Based on the accumulated information, the cluster introducer of the fog gateway executes the proposed policy of placing the modules of the corresponding applications over the fog cluster.

A fog gateway can remain connected to multiple fog clusters; however, each cluster has only one introducer at a time.

Each fog gateway maintains a list of possible introducers for all accessible fog clusters so that communication with the fog infrastructure can be restored immediately after any introducer node fails. As the cluster nodes share a mesh network, a synchronization operation among introducers of different fog gateways is instantly performed.

B. PROBLEM FORMULATION

At any instance, a fog cluster can receive requests to place a set of η applications $A = \{a_1, a_2, a_3, \ldots.., a_\eta\}$. As shown in Figure 2, each candidate application a_i is a collection of M^i modules, where $M^i = \{m_1^i, m_2^i, m_3^i, \ldots.., m_k^i\}$. The interactions between the modules within an application follow a Distributed Data Flow (DDF) model. According to this model, a module $m_q^i \in M^i$ receives input from its antecedent set of modules $\Psi_q^i \subset M^i$ in the data flow, and after processing the input, the output is forwarded to the subsequent set of modules $\Phi_q^i \subset M^i$ as input. Additionally, when receiving inputs from any module $m_p^i \in \Psi_q^i$, m_q^i imposes an expected data-dependency delay δ_{pq}^i such that its data-driven interactions with the antecedent modules follow the time certainty τ_q^i where

$$\tau_q^i =< \min \delta_{p'q}^i, \max \delta_{p^{\backprime}q}^i >; \forall p', p^{\backprime} \in \Psi_q^i \tag{1}$$

Applications deployed in the fog computing environment are based on the DDF model presented by [35]. According to [12], DDF model is the best suited for the applications with multiple interdependent modules.

In this context applications $a_1, a_2, \ldots.$ can have various requirements, such as computing resource requirements, maximum response time, etc. An application a_i contains several tasks which are represented as application modules $m_1^i, m_2^i, \ldots, m_k^i$, each of them can be executed on a fog node independently from the others while maintaining the same requirements across all modules of the application a_i. In this study, the tasks and modules share the same meaning and are used interchangeably throughout the text. The requirements of the application module m_k^i are as follows:

$$Req(m_k^i) = (CPU_{m_k^i}, RAM_{m_k^i}, Bandwidth_{m_k^i}) \tag{2}$$

In addition to that, all application modules of an application $a_i \in A$ share the maximum delay Δ_{a_i} the application can tolerate. According to [36], the service delay (l_{a_i}) for application a_i is given by:

$$l_{a_i} = \sum_{i=1}^{k} \delta_{pq}^i; \forall m_p^i, m_q^i \in M^i \tag{3}$$

where M^i denotes the set of modules for application a_i. δ_{pq}^i refers to the delay time between application modules m_p^i and m_q^i because of the data dependency between them, which can be tolerated; that is,

$$\sum_{i=1}^{k} \delta_{pq}^i \leq \Delta_{a_i} \tag{4}$$

where Δ_{a_i} is the maximum delay time that application a_i can tolerate, i.e, the maximum delay time that application can allow before a QoS violation occurs. Therefore, the equation in (2) can be extended to

$$Req(m_k^i) = (CPU_{m_k^i}, RAM_{m_k^i}, Bandwidth_{m_k^i}, l_{a_i}) \tag{5}$$

Node $n \in N$ is a computing resource that can either be a fog node or a cloud node:

$$n = \begin{cases} \text{Cloud node, if } n \in N_C \\ \text{Fog node, if } n \in N_F \end{cases} \tag{6}$$

where N_C and N_F refer to the subsets of the cloud and fog nodes, respectively. Let MST denote makespan time, which is the total time required to finish a task/module m_k^i on node n from start to end and can be presented as follows:

$$MST(m_k^i, n) = l_{m_k^i} + w_{m_k^i} + p_{m_k^i} \tag{7}$$

where $w_{m_k^i}$ refers to the waiting time required for module m_k^i to begin the execution on node n. The term $p_{m_k^i}$ refers to the processing time required to execute the application module m_k^i. This is calculated as follows:

$$p_{m_k^i} = \frac{len(m_k^i)}{\mu_n} \tag{8}$$

$len(m_k^i)$ refers to the length of module m_k^i in terms of millions of instructions, and μ_n is the CPU processing rate of the node to which the module will be allocated. The utilization of node n can be computed as follows.

$$u(n) = \frac{\sum_{i=1}^{\eta} CPU(m_k^i)}{CPU_n} * 100 \tag{9}$$

where $CPU(m_k^i)$ denotes the total CPU computing power required for all application modules at node n.

Although latency is an important factor when it comes to choosing a node to place a task, this work argues that the QoS of fog computing environments can be improved by reducing the total time required to submit, place, execute, and finish a task rather than solely focusing on latency.

However, this assumption is not adequate for latency-sensitive applications; therefore, it is vital to consider whether the application allows a fog node to be placed according to its requirements. This study proposes a policy to determine the most suitable fog node to place a task/module m_k^i. This improves the performance of this task without violating the following condition:

$$MST(m_k^i, n) \leq \Delta_{a_i}; \forall m_k^i \in M^i \tag{10}$$

The violation cost of an application should be minimal when different modules of an application are allocated to different fog cloud nodes. The proposed policy first check for the nodes where previous modules of an application are placed. The priority of the node depends on the communication latency among the modules. If two modules are placed on different nodes then their expected data

dependency delay increases. This causes deadline miss for the application and degrades the QoS. So the modules with their inter-dependency may be scheduled on the same node if the resource constraints are satisfied. The violation cost of an application (a_i) is calculated as follows:

$$V_{a_i} = \frac{max(0, F_{a_i} - \Delta_{a_i})}{\Delta_{a_i}} \times 100 \tag{11}$$

where F_{a_i} denotes the completion time for an application a_i. The violation cost is proportional to the delay incurred by the application to complete execution. The violation cost is zero if the application satisfies the deadline requirements.

Algorithm 1 Performance-Aware Placement Mechanism (PEAPM)

```
1: get m, l_max
2:  Queue(Q) <- Sorted by deadline of modules
3: MST_min ← maxMST
4: index = -1
5: while Q is not empty do
6:   foreach n in nodeList do
7:     if Req(m) ≤ Cap(n) && n.delay ≤ l_max then
8:       MST_tmp = MST(m, n)
9:       if MST_tmp ≤ MST_min then
10:        MST_min = MST_tmp
11:        index = nodeList.getIndex(n)
12:      end if
13:    end if
14:  end for
15: end while
16: if index == -1 then
17:   return false
18: else
19:   updateCap(n,m)
20:   return true
21: end if
```

Algorithm 2 Power-Aware Placement Mechanism (POAPM)

```
1: get UtilThres
2: foreach n in nodeList do
3:   if n.util < UtilThres then
4:     sortedNodeList ← sortDecrUtil(nodeList)
5:     i ← 0
6:     foreach m in n.mList do
7:       c_n ← sortedNodeList[i]
8:       if Req(m) ≤ Cap(c_n) && c_n.delay ≤ m.l_max then
9:         updateCap(c_n,m)
10:        remove(n,m)
11:      else
12:        i++
13:      end if
14:    end for
15:  end if
16: end for
```

IV. PROPOSED MECHANISM

This section proposes a novel placement mechanism for fog-cloud systems. This mechanism focuses on performance and power by placing an application module on a fog node to reduce the makespan time (MST) of the application modules. In addition, the mechanism was extended to reduce the number of working fog nodes by stacking more application modules onto fewer fog nodes.

Such a strategy can help improve utilization and therefore reduce power consumption by fog nodes. Here, we also consider that if any application can be executed through cloud nodes without compromising the deadline, then we place that application on the cloud node [37]. This improves performance and reduces the number of fog nodes required in the setting.

A. PERFORMANCE-AWARE PLACEMENT MECHANISM

The performance aware placement mechanism, referred to as PEAPM, and is depicted in Algorithm 1. The algorithm starts with sorting the modules of the application based on their deadlines (Line 2). Then the modules are selected for each application and taken in order of their dependency constraints and start with the module without any dependency constraint belong to that application. The way applications are ordered based on their deadline (earliest deadline is first considered), similarly modules of each application are ordered based on their dependencies. The modules with dependency are analyzed based on their data dependency delay. The nodes are prioritized for a module which incurs minimal data dependency delay. Nodes closer to each other would be selected to reduce the data dependency delay. However the processing speed of the node plays a vital role for module placement. The node with the least MST is selected to host the application module, provided that the node can accommodate the module, i.e, the available CPU and RAM of the node are equal to or larger than the module's CPU and RAM requirements, as shown in line 7. It is important that the placement of the application module on the selected node does not violate the delay requirement of the application. Line 8 invokes a method for calculating the estimated time required to complete the task, as mentioned in Equation 7. After finding and placing the module on the selected node, Line 19 updates the node that accommodates the module. If no fog node can accommodate the module, the algorithm returns false results. The module can then be sent to cloud computing, depending on the resource and delay requirements.

Time Complexity: The time complexity of the Algorithm 1 can be analysed as follows. Step-2 of the algorithm takes $O(nk \times log(nk))$ time, where n represents the number of applications and k represents average number of modules per application. However the the while loop and for loop (line 5 and 6) dominates the time complexity of the entire algorithm. The maximum time complexity of the algorithm can be

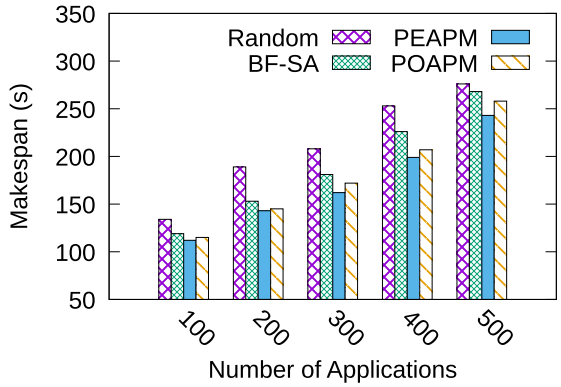

FIGURE 3. Makespan.

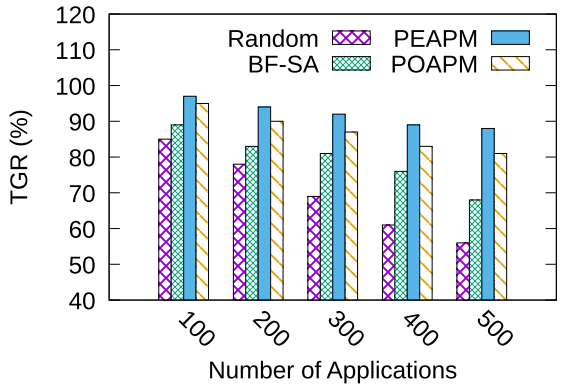

FIGURE 4. Task guaranteed ratio.

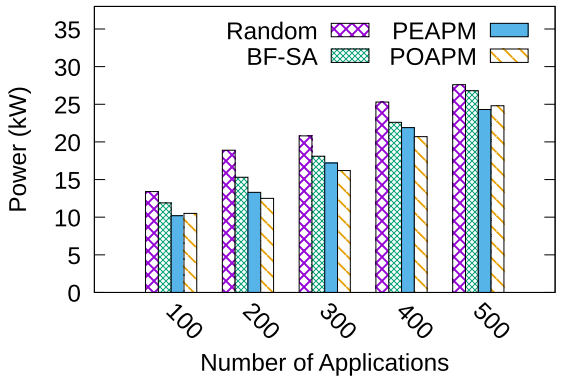

FIGURE 5. Power consumption.

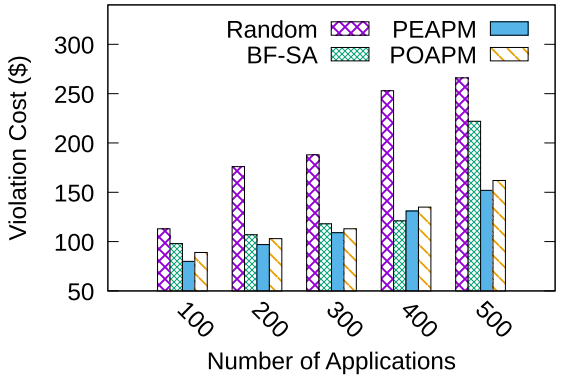

FIGURE 6. Violation cost.

represented by $O(nkm)$, where m represents the number of fog and cloud nodes in total.

B. POWER-AWARE PLACEMENT POLICY

Power consumption plays a crucial role in the cloud and fog computing environments by reducing the running and maintenance costs for both service providers and consumers. The utilization of computing resources has a positive impact on power savings because it reduces the number of computing nodes running in fog or cloud computing. A recent study showed that the application of Dynamic voltage and frequency scaling (DVFS) to CPUs can yield an almost linear power-to-frequency relationship for computing nodes [38]. Therefore, this section proposes a novel mechanism that can reduce the power consumption in a fog environment by reducing the total number of running fog nodes. The idle nodes can be switched to the power-safe mode.

The utilization of a fog node is given by 9. This mechanism aims to reduce the number of fog devices hosting the application modules. Here, we propose a power-aware placement mechanism (POAPM) described in Algorithm 2. The algorithm began by surveying the utility level of each fog node. If the utility level is less than a particular threshold, it triggers the migration process of all the application modules currently running on the node.

The utility threshold is provided in the algorithm as an input to determine the optimal threshold that yields the optimum power-saving results. Setting the threshold to a small value such as 10% would result in only a small proportion of application modules being migrated to another node, thereby providing a limited improvement in power savings. Similarly, setting the threshold to a large value can lead to an unlimited number of modules that must be moved. The choice of this threshold will be left to the trial and error approach; i.e., this work will change the threshold values during the experimental phase to find the most appropriate value.

In line 4, the algorithm sorts the nodes in the fog layer in ascending order according to their current utilization levels. The node with the highest utilization level is selected first. The algorithm attempts to stack maximum application modules on the same node to improve utilization, thus increasing power savings. Methods *updateCap* and *remove* are used to migrate an application module m from the underutilized node n to the candidate node n_c by placing the module in the candidate node and removing it from the underutilized node. To comply with the performance of fog environment line 8 in the algorithm, the latency of the candidate node does not violate the latency requirement; i.e., the latency will increase if the module moves to the new node. If so, this module will not be placed in this node, and another candidate will be considered. The algorithm places

TABLE 2. Simulation configuration.

Parameter	Value
Simulation duration	120-240 sec
Monitoring period	10 sec
Latency:	
IoT device to fog nodes	10-15 ms
Fog node to Cloud	140-160 ms
Application service delivery deadline	350-750 ms
Fog devices inter-communication delay	3-5 ms
Application data receiving frequency	3/sec-7/sec

modules to nodes such that each node will be as compact as possible. The compactness will result in requirement of less number of fog nodes and that leads to less energy consumption.

Time Complexity: The time complexity of the Algorithm 2 can be analysed as follows. Step-2 of the algorithm takes $O(m)$ times and step-4 takes $O(mlogm)$ times for sorting the nodes based on their utilization. Combining step-2 and step-4 take $O(m^2logm)$ times. So the maximum time complexity of the algorithm can be represented by $O(nm^2logm)$, where m represents the number of fog and cloud nodes in total.

V. EVALUATION

To test the proposed mechanisms described in Section IV, the iFogsim simulation tool [39] was used to run and test the mechanisms under various workloads. In this experiment, synthetic data were used as the input workload. The simulation parameters within a specific range were determined using a pseudo-random number generator. The simulation configurations considered for this experiment is reported in Table 2. The module configurations used in the experiments are listed in Table 3.

The proposed algorithms were compared with two other algorithms for evaluation purposes.

- **Random:** The modules of each application are randomly allocated to the cloud or fog node considering the resource constraints.
- **BF-SA:** This is the heuristic model to allocate the modules. The proposed approach considers the slack-aware best fit approach to allocate the modules to nodes. Here, we consider the resource and deadline constraints for the module allocation [40].

A. EVALUATION CRITERIA

In this section, we evaluate the four approaches based on four evaluation criteria. Here, we aim to minimize the makespan, power consumption, and violation cost and maximize the task guaranteed ratio (TGR). The TGR is calculated as follows [41]:

$$TGR = \frac{\sum_{i=1}^{n} U_i}{n} \times 100 \tag{12}$$

where $U_i = 1$ if the application completes its execution before the deadline; otherwise, $U_i = 0$.

B. IMPACT OF NUMBER OF APPLICATIONS

In this experiment, we compared our proposed approach with other state-of-the-art approaches and evaluated the performance of our approach by varying the number of tasks. Figure 3, Figure 4, Figure 5, and Figure 6 illustrate the performance in terms of different matrices, which were discussed in the previous section. Figure 3 reports the makespan time for the set of applications in our simulation, and it was found out that PEAPM and POAPM took less time to complete the set of applications.

Our proposed approach, PEAPM, attempts to allocate the modules in a more distributed and compact manner to the available fog-cloud nodes based on resource and deadline constraints. Our proposed approaches (PEAPM and POAPM) had better TGR (Figure 4) as compared to other two approaches (Random and BF-SA); therefore, the violation cost is minimal as compared to other approaches (as shown in Figure 6). Because fewer servers are required for application execution, our approach consumes less power than the other approaches, as shown in Figure 5.

C. IMPACT OF NUMBER OF FOG SERVERS

In this experiment, we fixed the number of cloud nodes (10) and reported the performance of the different approaches by varying the number of fog nodes (20-60). As the fog nodes are closer to the application request points, the performance in terms of TGR and violation cost is better than those in other experimental settings. Figure 7 illustrates the variation in makespan time with the number of fog nodes. The results demonstrate that as the number of fog nodes increases, the makespan time decreases. This can be attributed to the fact that when more computational resources are located near the application generation point, the TGR increases, while the violation costs decrease (as indicated in Figure 8 and Figure 10).

Our proposed approach allocates more tasks to fog nodes, which reduces deadline violations compared to other approaches. POAPM is a power-aware approach that considers the compact allocation of modules to servers to use a minimum number of servers. Therefore, POAPM performs better than the other three approaches in terms of power consumption (Figure 9).

D. IMPACT OF NUMBER OF CLOUD SERVERS

The results of the experiment are shown in Figure 11 to Figure 14. We considered a fixed number of fog nodes (10) and varied the number of cloud nodes from 20 to 60. As the number of cloud nodes increases, the makespan (Figure 11) and TGR (Figure 12) for all approaches improve. However, the power consumption and violation cost for this experiment did not improve, because the modules sent to the cloud servers required more time to complete their execution, and each cloud server consumed more power than the fog server.

TABLE 3. Module configuration.

Type	CPU Length (in MIPS)	Network Length (in Bytes)	Maximum Delay (in milliseconds)
Sensors	100-500	2,000	1 - 10 ms
Client module	500-2,000	20,000	20 - 100 ms
Analysis module	3000-10,000	20,000-40,000	20 - 100 ms
Event handler module	2,000-5,000	10,000-30,000	20 - 100 ms
Actuators	500-2,000	20,000	5 - 20 ms

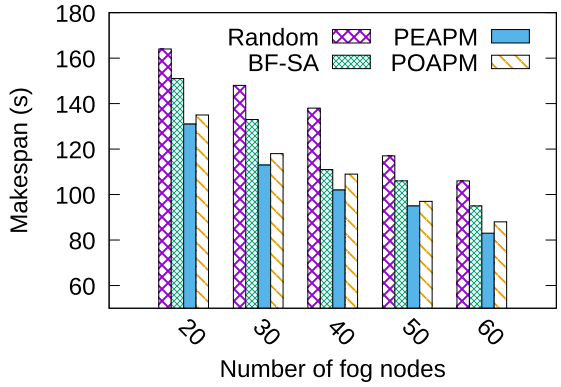

FIGURE 7. Makespan.

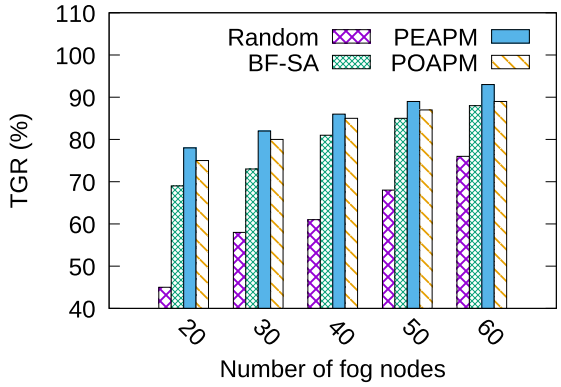

FIGURE 8. Task guaranteed ratio.

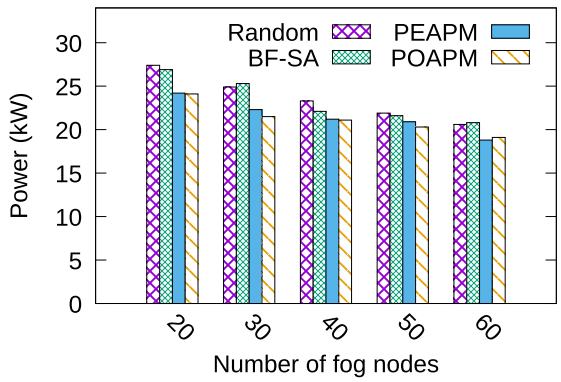

FIGURE 9. Power consumption.

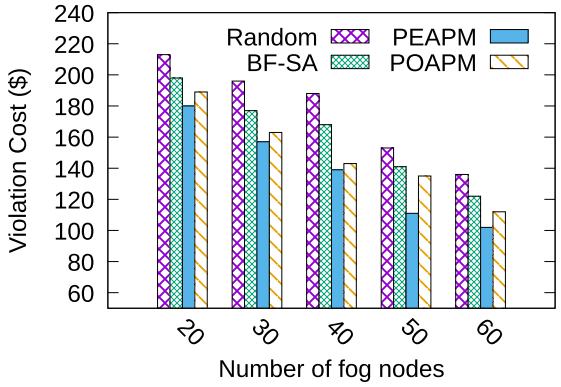

FIGURE 10. Violation cost.

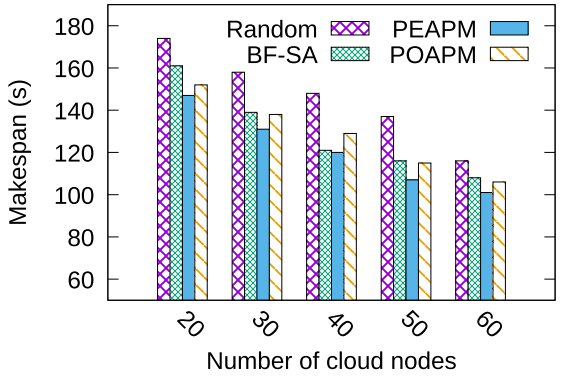

FIGURE 11. Makespan.

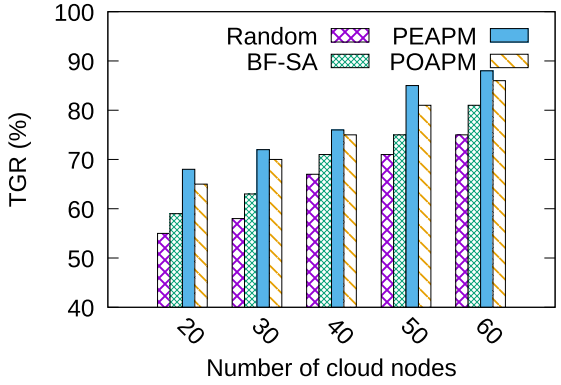

FIGURE 12. Task guaranteed ratio.

PEAPM and POAPM performed better than the Random and BF-SA approaches under these settings. The performance improvement in our approach is owing to the segregation of the latency-sensitive and latency-tolerant modules, with each type of module handled separately to achieve better performance. The violation cost for random cases was extremely poor, whereas PEAPM was the best among the four approaches.

TABLE 4. ANOVA test results for PEAPM, POAPM, BF-SA, and Random for synthetic datasets.

Source of variation	df	Sum of square	Mean square	F value	p value	F critical
Between Groups	3	58365.05	19455.01	3.8734	0.0101	2.6507
Within Groups	196	984461.98	5022.76			
Total	199	1042827.03				

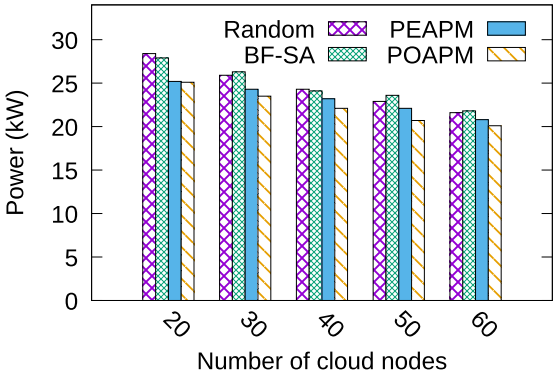

FIGURE 13. Power consumption.

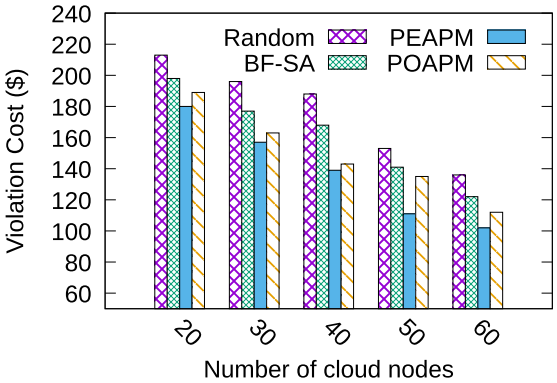

FIGURE 14. Violation cost.

E. ANALYSIS OF VARIANCE (ANOVA) RESULTS

Analysis of Variance (ANOVA) [42] is used to test the difference in means in two or more cases. This statistical method is being performed using SPSS software. The null hypothesis assumed to be the means of four populations are equal. Mathematically we can represent that as H_0: $\mu_1 = \mu_2 = \mu_3$. In the alternate hypothesis, we assume that at least one of the means is different from other means. We conducted ANOVA for synthetic data sets with the $\alpha = 0.05$. The results of the ANOVA are reported in Table 4. For the synthetic dataset, the F value > F critical (Table 4), that means we reject the null hypothesis. This implies the population means of four different datasets are not equal and it is due to p value is much less than 0.5. Hence, we assert that the performance gain achieved by PEAPM against POAPM, BF-SA, and Random is not by chance.

VI. CONCLUSION AND FUTURE DIRECTION

This study presented a novel application-module placement policy for a fog-cloud framework that focuses on placing application modules on nodes to reduce processing time while maintaining an acceptable level of delay time. We developed two heuristics, PEAPM and POAPM, and evaluated their performances using simulations at different settings. Based on the experimental results, it was determined that our proposed algorithms outperformed others in terms of TGR, makespan, power consumption, and violation cost. The performance of the proposed approaches varied based on different server settings in the fog-cloud environment.

There is a trade-off between the number of servers deployed in the fog or cloud environment, and the proportional number of server deployments at each layer can improve the overall performance. However, it is necessary to investigate the optimal number of servers that should be deployed in each layer (fog and cloud) to enhance the system's performance. This aspect should be investigated in future studies. Furthermore, the use of various meta-heuristic optimization frameworks can be explored to enhance system performance, providing a promising avenue for further investigation in this area.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st, Ed., MCC Workshop Mobile Cloud Comput. (MCC)*, vol. 131. New York, NY, USA: ACM Press, Aug. 2012, p. 13.

[2] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3909–3914.

[3] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet Things J.*, vol. 3, no. 2, pp. 161–169, Apr. 2016.

[4] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: From architecture to edge computing and big data processing," *J. Supercomput.*, vol. 75, no. 4, pp. 2070–2105, Apr. 2019.

[5] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the Internet of Things," *Pervasive Mobile Comput.*, vol. 52, pp. 71–99, Jan. 2019.

[6] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 85–97, Feb. 2019.

[7] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[8] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 1222–1228.

[9] Z. Pooranian, M. Shojafar, P. G. V. Naranjo, L. Chiaraviglio, and M. Conti, "A novel distributed fog-based networked architecture to preserve energy in fog data centers," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2017, pp. 604–609.

[10] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar. 2017.

[11] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-aware fog service placement," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, May 2017, pp. 89–96.

[12] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–21, Feb. 2019.
[13] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (QoE)-aware placement of applications in fog computing environments," *J. Parallel Distrib. Comput.*, vol. 132, pp. 190–203, Oct. 2019.
[14] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "Meet genetic algorithms in Monte Carlo: Optimised placement of multi-service applications in the fog," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2019, pp. 13–17.
[15] T. S. Nikoui, A. Balador, A. M. Rahmani, and Z. Bakhshi, "Cost-aware task scheduling in fog-cloud environment," in *Proc. CSI/CPSSI Int. Symp. Real-Time Embedded Syst. Technol. (RTEST)*, Jun. 2020, pp. 1–8.
[16] M. Louail, M. Esseghir, and L. Merghem-Boulahia, "Dynamic task scheduling for fog nodes based on deadline constraints and task frequency for smart factories," in *Proc. 11th Int. Conf. Netw. Future (NoF)*, Oct. 2020, pp. 16–22.
[17] F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti, "Throughput-aware partitioning and placement of applications in fog computing," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2436–2450, Dec. 2020.
[18] R. Mahmud, A. N. Toosi, K. Ramamohanarao, and R. Buyya, "Context-aware placement of Industry 4.0 applications in fog computing environments," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7004–7013, Nov. 2020.
[19] A.-V. Postoaca, C. Negru, and F. Pop, "Deadline-aware scheduling in cloud-fog-edge systems," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput. (CCGRID)*, May 2020, pp. 691–698.
[20] M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, "Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment," *Softw., Pract. Exp.*, vol. 51, no. 8, pp. 1745–1772, Aug. 2021.
[21] T. Lähderanta, T. Leppänen, L. Ruha, L. Lovén, E. Harjula, M. Ylianttila, J. Riekki, and M. J. Sillanpää, "Edge computing server placement with capacitated location allocation," *J. Parallel Distrib. Comput.*, vol. 153, pp. 130–149, Jul. 2021.
[22] E. Batista, G. Figueiredo, and C. Prazeres, "Load balancing between fog and cloud in fog of things based platforms through software-defined networking," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7111–7125, Oct. 2022.
[23] N. Godinho, H. Silva, M. Curado, and L. Paquete, "A reconfigurable resource management framework for fog environments," *Future Gener. Comput. Syst.*, vol. 133, pp. 124–140, Aug. 2022.
[24] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 1243–1259, Jun. 2022.
[25] J. C. S. D. Anjos, K. J. Matteussi, P. R. R. De Souza, G. J. A. Grabher, G. A. Borges, J. L. V. Barbosa, G. V. González, V. R. Q. Leithardt, and C. F. R. Geyer, "Data processing model to perform big data analytics in hybrid infrastructures," *IEEE Access*, vol. 8, pp. 170281–170294, 2020.
[26] P. R. R. De Souza, K. J. Matteussi, A. D. S. Veith, B. F. Zanchetta, V. R. Q. Leithardt, Á. L. Murciego, E. P. De Freitas, J. C. S. D. Anjos, and C. F. R. Geyer, "Boosting big data streaming applications in clouds with BurstFlow," *IEEE Access*, vol. 8, pp. 219124–219136, 2020.
[27] H. J. D. M. Costa, C. A. D. Costa, R. D. R. Righi, R. S. Antunes, J. F. D. P. Santana, and V. R. Q. Leithardt, "A fog and blockchain software architecture for a global scale vaccination strategy," *IEEE Access*, vol. 10, pp. 44290–44304, 2022.
[28] F. C. Orlandi, J. C. S. D. Anjos, V. R. Q. Leithardt, J. F. D. P. Santana, and C. F. R. Geyer, "Entropy to mitigate non-IID data problem on federated learning for the edge intelligence environment," *IEEE Access*, vol. 11, pp. 78845–78857, 2023.
[29] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–37, Sep. 2020.
[30] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.
[31] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.
[32] E. Batista, G. Figueiredo, M. Peixoto, M. Serrano, and C. Prazeres, "Load balancing in the fog of things platforms through software-defined networking," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1785–1791.
[33] Y. Zhang, F. Zhang, S. Tong, and A. Rezaeipanah, "A dynamic planning model for deploying service functions chain in fog-cloud computing," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 10, pp. 7948–7960, Nov. 2022.
[34] M. Veeramanikandan and S. Sankaranarayanan, "Publish/subscribe based multi-tier edge computational model in Internet of Things for latency reduction," *J. Parallel Distrib. Comput.*, vol. 127, pp. 18–27, May 2019.
[35] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, "Developing IoT applications in the fog: A distributed dataflow approach," in *Proc. 5th Int. Conf. Internet Things (IoT)*, Oct. 2015, pp. 155–162.
[36] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–43, Jul. 2021.
[37] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A survey," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–41, 2019.
[38] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *Proc. Int. Conf. Autonomic Comput.*, Jun. 2008, pp. 3–12.
[39] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "IFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.
[40] C. K. Swain, P. Routray, S. K. Mishra, and A. Alwabel, "Predictive VM consolidation for latency sensitive tasks in heterogeneous cloud," in *Proc. 4th Int. Conf. Adv. Distrib. Comput. Mach. Learn. (ICADCML)*. Singapore: Springer, 2023, pp. 135–150.
[41] C. K. Swain and A. Sahu, "Interference aware workload scheduling for latency sensitive tasks in cloud environment," *Computing*, vol. 104, no. 4, pp. 925–950, Apr. 2022.
[42] K. E. Müller and B. A. Fetterman, *Regression and ANOVA: An Integrated Approach Using SAS Software*. Hoboken, NJ, USA: Wiley, 2003.

ABDULELAH ALWABEL (Member, IEEE) received the B.Sc. degree in computer science from King Saud University, Saudi Arabia, in 2006, the M.Sc. degree in advanced computing—internet technologies with security from the University of Bristol, U.K., in 2010, and the Ph.D. degree in computer science from the University of Southampton, U.K., in 2015. He is currently an Assistant Professor with Prince Sattam bin Abdulaziz University. His research interests include fault–tolerance mechanisms and resource management in cloud computing, fog computing, and the IoT. He has published several articles investigating the impact of failures in cloud computing.

CHINMAYA KUMAR SWAIN received the M.Tech. degree in computer science and engineering from the Indian Institute of Technology Bombay, Mumbai, and the Ph.D. degree from the Indian Institute of Technology Guwahati, India. He is currently an Assistant Professor with SRM University, Andhra Pradesh, India. His research interests include real time scheduling, resource management in cloud systems, and distributed computing systems.