symmetry

MDPI

Article

Modeling and Analyzing Offloading Strategies of IoT Applications over Edge Computing and Joint Clouds

Jaber Almutairi [1] and Mohammad Aldossary [2,*]

[1] Department of Computer Science, College of Computer Science and Engineering, Taibah University, 12512 Al-Madinah, Saudi Arabia; jalmutairi@taibahu.edu.sa
[2] Department of Computer Science, College of Arts and Science, Prince Sattam Bin Abdulaziz University, 11942 Al-Kharj, Saudi Arabia
* Correspondence: mm.aldossary@psau.edu.sa

Abstract: Internet of Things (IoT) is swiftly evolving into a disruptive technology in recent years. For enhancing customer experience and accelerating job execution, IoT task offloading enables mobile end devices to release heavy computation and storage to the resource-rich nodes in collaborative Edges or Clouds. However, how different service architecture and offloading strategies quantitatively impact the end-to-end performance of IoT applications is still far from known particularly given a dynamic and unpredictable assortment of interconnected virtual and physical devices. This paper exploits potential network performance that manifests within the edge-cloud environment, then investigates and compares the impacts of two types of architectures: Loosely-Coupled (LC) and Orchestrator-Enabled (OE). Further, it introduces three customized offloading strategies in order to handle various requirements for IoT latency-sensitive applications. Through comparative experiments, we observed that the computational requirements exerts more influence on the IoT application's performance compared to the communication requirement. However, when the system scales up to accommodate more IoT devices, communication bandwidth will turn to be the dominant resource and becomes the essential factor that will directly impact the overall performance. Thus, orchestration is a necessary procedure to encompass optimized solutions under different constraints for optimal offloading placement.

Keywords: edge-cloud computing; edge orchestrator; offloading strategy; latency sensitivity; internet of things

Citation: Almutairi, J.; Aldossary, M. Modeling and Analyzing Offloading Strategies of IoT Applications over Edge Computing and Joint Clouds. *Symmetry* **2021**, *13*, 402. https://doi.org/10.3390/sym13030402

Academic Editor: Yin Zhang

Received: 23 January 2021
Accepted: 25 February 2021
Published: 1 March 2021

1. Introduction

Recently there has been substantial growth in the number of connected devices in the digitalized era. The number of connected digital devices will have doubled between 2014 and 2019 [1]. This disruptive time is known as the era of the Internet of Things (IoT), which has attracted the attention of both academia and industry.

One of the biggest issues is that IoT devices are heterogeneous in terms of hardware, network technologies and platforms. Each IoT device has a unique address and can communicate with other devices, which requires transferring, processing and storing the data generated by these devices. Therefore, this immense growth requires platforms to support the increased amount of IoT devices as well as organize and process the produced data, since IoT devices are limited in terms of power and computational capabilities, i.e., Central Processing Unit (CPU) and memory [2].

IoT can be generally defined as "a dynamic global network infrastructure with self-configuring capabilities based on standards and interoperable communication protocols; physical and virtual *things* in an IoT have identities and attributes and are capable of using intelligent interfaces and being integrated as an information network" [3]. IoT interconnects physical objects including sensors, vehicles and buildings into a virtual circumstance, resulting in the increasing integration of cyber-physical objects.

These *things* are mobile in their nature, requiring data from other sources, and thus limited to computational power. In this context, to deal with energy and performance issues, Cloud and Edge computing underpin IoT devices and guarantee IoT service provisions particularly by task offloading. Offloading transfers computations from the resource-limited mobile device to resource-rich Edge and Cloud nodes in order to improve the execution performance of mobile applications and the holistic power efficiency. Offloading has been employed in many numerous sectors such as transportation, health care, homes and factories [4]. Offloading is especially effective for applications such as streaming processing, augmented reality, online gaming and video conferencing [5], which are sensitive to latency and necessitate high quality of services. In this scenario, moving data from and to the cloud goes hand in hand with the offloading processes and varies from one application to another. For instance, some applications require high computation resources with low communication and vice versa. What is worse, the number of devices and users can be explosively increased in some areas due to the mobility feature, which aggravates the network issue [6].

IoT workloads typically involve a great number of data-stream and control flows across different regions that require real-time processing and analytic. Currently, the most effective way to tackle this is to exploit small-scale access points to complement cloud resources in the network edge. Such access points in communication fields can provide a small-scale but resource-rich intermediate like computers or clusters in the vicinity of mobile users. For example, standalone multilayered architecture including Cloudlet [7], CloudAP [8] and other systems [9,10] aim to underpin latency-sensitive applications whilst minimizing the overall service time. However, the quality of service cannot be fully guaranteed due to the lack of integration and effective orchestration between Cloudlets and Cloud. To this end, other works assisted by *Orchestrator* can improve intelligent task offloading over the Edges and Clouds by adopting different orchestration algorithms [11,12]. However, how different offloading architecture and strategies quantitatively influence the end-to-end performance of IoT applications and services is still far from known particularly when the dynamicity of resource patterns and task characteristics manifests.

In this paper, we analyze and uncover how different Edge-Cloud architectures affect the overall IoT service time during the tasks are offloaded and how different application parameters including computation and communication demands impact the holistic efficiency. Specifically, we categorize the basic offloading schemes into two different classes: the Loosely-Coupled (LC) three-tier architecture and Orchestrator-Enabled (OE) three-tier architecture. We further compare how these two schemes impact the IoT service execution through performance-driven modeling, considering the communication latency derived from different network connections between tiers and the allocation status of computation resources. Besides, we present three offloading strategies (static-ratio policy, least-load policy and probabilistic policy) in order to seek optimal orchestration, considering diverse task characteristics.

The main contributions of this work can be summarized as follows:

- Propose a performance-driven method to measure the end-to-end IoT service effectiveness, taking both computational and communication resources into account;
- Conduct an asymptotic performance analysis to deeply dive into system behavior under the loosely-coupled and the orchestrator-enabled offloading schemes;
- Provide a set of models and algorithms in order to handle various IoT applications' requirements and achieve an optimal offloading strategy;
- An evaluation of the IoT task offloading strategies under different constraints, which can be employed in improving the efficiency of task offloading and achieving well-balanced resource management in Edge-Cloud environments.

The rest of this paper is organized as follows: we firstly summarize the related work in Section 2, in order to present the best knowledge of the research problem. Afterward, we outline the main research problem and challenges in Section 3. Section 4 depicts the primary methodology of our analysis and modeling, followed by the simulation-based

evaluation and result analysis in Section 5. Section 6 presents the research limitations. Eventually, we conclude this paper and discuss the future works in Section 7.

2. Related Work

The issue of service time in Edge-Cloud computing environment has received considerable attention among the research community. The existing literature on service time is extensive but the major focus has been on computational delay, communication delay or both. The following studies have investigated the service time with other objectives such as energy and cost. Dinh et al. [13] proposed a framework to minimize the computational latency and power consumption of mobile devices by offloading tasks to several edge nodes. Du et al. [14] proposed an algorithm to guarantee an acceptable computational delay in Fog/Cloud system. Liu et al. [15] designed a task scheduling algorithm to reduce the overall latency taking into consideration the state of queuing and execution. Rodrigues et al. [16] proposed a hybrid method to minimize end-to-end latency by enhancing the process of computational and communication resources, which focuses on virtual machine migration and transmission delay. Wei et al. [17] presented an algorithm that aims to reduce the latency by considering both transmission power and processing frequencies. Yang et al. [18] proposed an approach to support latency-sensitive applications by guaranteeing the maximum allowable time as well as reducing the operational cost. Zeng et al. [19] presented an algorithm to minimize the completion time of tasks based on load balancing and allocating task images. Hence, we presented some of the previous efforts that conducted in the context of minimizing service time. Next, we will represent the related work that considers the application types and requirements within the process to minimize the service time.

In the application side, there is a relatively small body of literature that is concerned with the different types of applications that are supported by Edge-Cloud system. IoT applications vary in terms of their computational and communication requirements [20] as well as their dynamic demands [21]. Fan and Ansari [22] proposed an approach to tackle the computational and communication delay with particular consideration of the application type by allocating the offloading tasks to appropriate resources in the edge-cloud system. Roy et al. [23] proposed a strategy to select the target edge node based on application specification in order to minimize delay and power consumption. Although these studies were conducted to deal with latency-sensitive IoT applications, the effects of different Edge-Cloud deployments have not been closely investigated in the context of minimizing the overall service time.

While some research has been focused on service time delay and some considerable application specifications, there is still very little scientific understanding of how different edge architectures affect the system performance particularly the overall service time. Different architectures of edge computing systems have been presented in [6,10,24]. These architectures have the same concept of pushing the computational resources closer to end-users. Differences mainly occur in Edge nodes and Edge networks, which include the deployment, network technology and feature of Edge nodes such as size, location and how it communicates with the central cloud. Such differences when considered by the service provider and application owner could be a contributing factor to minimize the overall latency.

3. Background and Problem Motivation

In this section, we briefly introduce basic terms and concepts before we state the key research problems and objectives of our work.

3.1. Resource and Workloads in IoT Scenario

3.1.1. Resource Entities in Edge and Cloud Environments

In Edge-Cloud environment, different kinds of resources at the network edge are added into the resource pool in order to reduce the backbone datacenter traffic and the

response latency of IoT services. To clarify the terms, we use *appliance* to describe fundamental entities in the edge-cloud environment. Appliances include *Things*, *Edge nodes* and *Cloud nodes*. Things are defined as networked devices including sensors and devices with built-in sensors, which can monitor and generate huge amounts of data. Cloud servers store the data and provide parallelized capability of computation.

It is worth noting that an Edge node is defined as particular equipment or middleware residing within the midst of things and the remote Cloud. It serves as an agent and in particular both computation, storage and network resources are provisioned much closer to IoT devices. Edge node collects data from a set of sensors, which is then transmitted to a centralized computing system that locally caches data and performs load balancing. Due to these functionalities, compared to nodes in Cloud datacenter, Edge node has relatively smaller computation capacity but is geodistributed. The significance of Edge nodes mainly lies in continuously provisioning networking services with high accuracy and low latency even though the backbone network access would be temporarily unavailable.

3.1.2. IoT Workloads

Complete IoT workloads typically consist of a group of IoT tasks that can be encapsulated in isolated containers. All containers are running on top of physical machines or virtual machines and are interconnected to form a function chain that best serves application's requirements. Formally, an IoT application can be depicted as a Direct Acyclic Graph (DAG) workflow, where each vertex within the workflow represents an IoT task. An example is illustrated in Figure 1, where the e-Health application can be divided into many independent but jointly-working micro-services [25,26]. Specifically, mobile health subsystems are capable of remote monitoring, real-time data analysis, emergency warnings, etc. Data collected from wearable sensors that monitor patient vitals can be continuously sent to data aggregators and, in the event of detection of abnormal behavior, hospital personnel can be immediately notified in order to take appropriate measures [25,26].

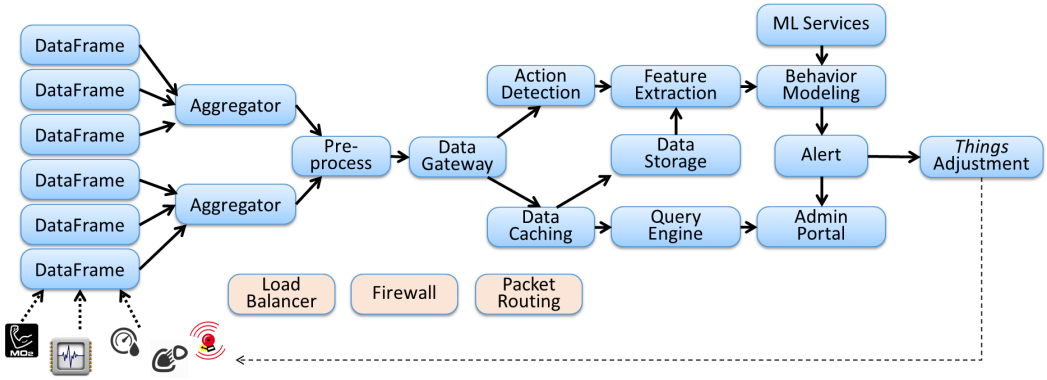

Figure 1. Internet of Things (IoT) application example: e-Health system workflow.

More generally speaking, an appropriate combination of these standalone IoT tasks can be used to facilitate more advanced functionality, allowing for reduced cost and improved user experience. In this work, we particularly target such IoT latency-sensitive applications.

3.1.3. IoT Deployment and Task Offloading

IoT Deployment: the main goal of IoT application deployment is to make each task of IoT tasks be placed on a specific location (e.g., IoT devices, Edge nodes, Cloud nodes) in Edge-Cloud system for their execution. To better capture and model the latency, we assume each task has two main attributes: (1) the amount of data to be moved to/from Edge-Cloud system, and (2) the amount of computational requirement (e.g., Central Processing Unit (CPU), Memory, Graphic Processing Unit (GPU), etc). Namely, resource allocation will ensure all IoT tasks have enough resources to execute their computation logic during which data may be generated and transmitted among different tasks. As aforementioned,

Edge-Cloud combinations will allow for delivering offloading tasks and migrating data in-between [27].

Task Offloading: offloading transfers computations from the resource-limited mobile device to resource-rich edge and cloud nodes in order to improve the execution performance of mobile applications and the holistic power efficiency. User devices are evenly located at the edge of the network. They could offload computation to Edge and Cloud nodes via WLAN network or 4G/5G networks. Broadly, if a single Edge node is insufficient to deal with the surging workloads, other Edge nodes or Cloud nodes are ready for assisting such application.

3.2. Research Problem and Challenges

3.2.1. Problem Statement

Mobile devices are adopted to better function IoT services. They, however, have extremely restricted capabilities in computation resources, storage, and battery, especially, when underpinning resource-intensive applications such as Augmented Reality (AR) and Artificial Intelligence (AI) assisted multimedia applications. In these cases, they urgently require low response latency and broad bandwidth throughput. To achieve this goal, different cross-cloud and Edge-Cloud architectures are proposed to allow for coordinated mechanisms among mobile devices, edge nodes and cloud resources.

The fundamental and commonly accepted means to deal with the coordination is to add resources at the network edge. For instance, computation, bandwidth and storage resources are moved closer to the IoT devices for reduced backbone network traffic and minimized response latency. In contrast, computation-intensive tasks are offloading to Cloud datacenter for acceleration by using parallel processing.

There are a large number of factors that will directly or implicitly influence the effectiveness and efficiency of task offloading. Therefore, this necessitates quantified analysis and modeling of their impacts on performance and make comparisons among different offloading strategies. In addition, IoT applications by nature vary in terms of the computation and communication demands as well as the supplies from current resources. This motivates us to investigate and evaluate how different IoT workloads' behaviors are by adopting different task offloading schemes. Evaluating the relationship between different schemes and various IoT workloads can improve the quality of service by reducing end-to-end service time.

3.2.2. Emerging Challenges

There are a number of challenges that manifest with the growing variety and number of IoT applications, such as:

- **Scale and Complexity:** with the increase of IoT manufacturers developing heterogeneous sensors and smart devices, selecting optimal resources to hold IoT tasks over the joint cloud environment becomes increasingly complicated when considering customized hardware configurations and personalized requirements. For example, some tasks can only operate with specific hardware architectures (e.g., ARM and Intel) or operating systems, while tasks with high-security requirements might require specific hardware and protocols to function. Not only does orchestration cater to such functional requirements, but it must also do so in the face of increasingly larger workflows that change dynamically. The orchestrator must determine whether the assembled systems comprising of Cloud resources, Edge nodes and end-user devices coupled with geographic distributions and constraints are capable of provisioning complex services correctly and efficiently. In particular, the orchestrator must be able to automatically predict, detect and resolve scalability bottlenecks which may arise from an increased application scale.
- **Dynamicity:** this is one of the main characteristics of IoT applications, whose topology or diverse resources may change dynamically. This is a particular problem in the context of software upgrades or frequent join-leave behavior of network objects which

will change its internal properties and performance, potentially altering the overall workload pattern. Specifically, the communication link is likely to be influenced by the fluctuation of bandwidth, connectivity and device mobility. It may lead to unpredictable demands of task offloading and resource allocation over the joint Clouds and Edge nodes.

- **Scalability:** scalability is another aspect of the challenge. In this context, it specifically refers to the capability of handling an explosive number of IoT applications and dynamic resource changes. Additionally, attributes of IoT tasks dynamically change, thus each task's procedure may have different execution times. In addition, IoT devices are mobile, the number of devices may increase in some areas, thus the workload will be increased for the connected edge node. Hence, the amount of IoT workload will change dynamically over Edge-Cloud system, which could lead to service performance degradation.

To summarize, all IoT tasks will be assigned to physical resources in Edge-Cloud system based on the strategy of the central resource manager. Offloading to an optimal destination according to runtime system status will definitely improve applications' performance. In this paper, we examine how different Edge-Cloud systems and environmental parameters influence the overall effectiveness and efficiency of task offloading, especially taking the above challenges into consideration.

4. Methodology

This section briefly describes the details of the offloading schemes that we are targeting and the relevant performance factors such as computation and communication delays that we need to tackle.

In general, Edge-Cloud systems that underpin the IoT applications consist of three tiers: End device tier, Edge tier and Cloud (Datacenter) tier. The Edge tier includes a number of Edge nodes that geographically distributed and linked to Cloud tier through the core network. Similarly, end-user devices are directly connected with the Edge nodes. It is worth noting that the network bandwidth and communication efficiency in between Edge and Cloud nodes are one of the dominant factors resulting in different performance of IoT devices.

The objective of this work is to measure the impact of task offloading on the IoT service performance within the Edge-Cloud system. Thus, we use the end-to-end service time as a performance metric in the two architectures and three strategies, respectively.

4.1. Mainstream Architectural Schemes

The mainstream of IoT Edge-Cloud systems can be summarized into two categories: *Loosely-Coupled scheme* and *Orchestrator-Enabled scheme*, as illustrated in Figure 2.

Loosely-Coupled (LC) Three-tier Scheme: in this architecture, IoT applications can be deployed over connected Edge-Cloud nodes, and tasks generated are scheduled into confirmed positions if the mobile devices have no enough capability to execute. Its procedure of making decisions is quite fast, same as routing in a router, hence the overhead of offloading is ignorable in almost all situations. This scheme has been adopted by numerous works [28–30].

Obviously, it indicates that tasks can only be offloaded into severely restricted nodes based on a *static policy*, such as fixed proportion and time interval. Thus, the increase in processing is bound. Only in balanced scenarios, in which every node equips performance-closely hardware and handles a similar amount of end devices and so forth, the strategy can work very well with higher stability and availability. In reality, the workload of mobile devices at run time could be offloaded to only one destination for sequential execution or multiple servers for parallel execution.

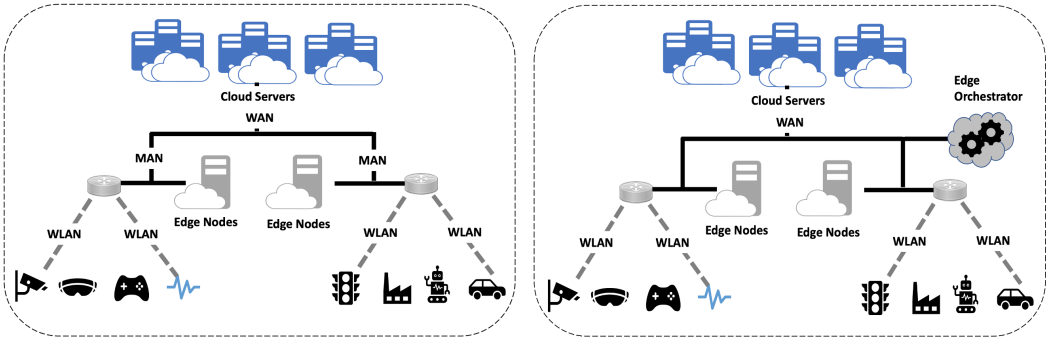

Figure 2. Two representative architectures for cross-cloud IoT services: (left) Loosely-Coupled (LC) three-tier scheme; (right) Orchestrator-Enabled (OE) three-tier scheme.

The major disadvantages of this scheme are: (1) ignorance of the diversity of task resource demand, and (2) lack of collaboration between Edge-Cloud infrastructures. Due to this mechanism adopting a static partition and delivery, once a task was offloaded onto a node, it has no chance to be adjusted according to its special characteristics.

Orchestrator-Enabled (OE) Three-tier Scheme: IoT applications can be deployed over multiple Edge-Cloud nodes controlled by *Edge Orchestrator*, a central scheduler in between Edge and Cloud. This scheme provides a flexible interface in which users can customize optimization strategies for dispatching tasks according to upper-level requirements or application characteristics. This paper introduces two practical strategies in OE: *least-load policy* and *probabilistic policy*, see Section 4.2 for more details.

Digressively, Edge Orchestrator is a critical component that manages resources pertaining to Edge-Cloud nodes and dispatches IoT tasks by binding each task with a specific resource. An effective and efficient placement algorithm, deciding the destination of task offloading, plays a decisive role in the reliability, availability and cost-efficiency of task redistribution. According to mathematical models and optimization methods, the work in [31] classifies offloading algorithms into five categories: 0–1 integer linear programming, K-dimensional bin parking, Markov decision process, convex optimization and Lyapunov optimization problems. Nearly all of them are solving an NP-hard problem, since the time complexity of algorithm coping with multiple dimensional packing problems is exponential. Thus, a balance between algorithm accuracy and time complexity needs to be achieved by considering various workload scales and types. A number of studies utilize this scheme to support their IoT applications such as [32–34].

Comparison: LC just purely links the IoT devices with a nearby Edge-Cloud node to achieve offloading. Once a task is offloaded to a specific node, it cannot be further moved even being in a long queue. In other words, the tasks have no opportunity to utilize ideal resources from other Edge-Cloud nodes, even if its specified collaborative node has no enough capacity holding whole tasks at present. LC mechanism allows tasks to wait to execute until enough available resources are released or to be offloaded into the corresponding Cloud based on a static policy. Apparently, the task offloading is unidirectional and cannot be collaboratively balanced among different Edge-Cloud infrastructures.

By contrast, the procedure of offloading tasks in OE is a little more strategic, in which optimization objectives should be achieved by previously defined rules or policies. In this situation, IoT tasks are firstly generated and sent by mobile devices to utilize Edge-Cloud resources and then routed and received by Edge Orchestrator in order to dispatch them into a near-optimal Edge-Cloud node based on a strategy, such as least-load policy and probabilistic policy. Edge Orchestrator is located among Access Points (APs), Edge and Cloud, which means that it can control not only communication between mobile end devices and Edge nodes but also traffic between Edge and Cloud nodes.

Subsequently, Edge Orchestrator is able to assign a received task to a near-optimal Edge-Cloud node with appropriate resources based on a previously specified offloading algorithm, considering various constraints such as resource availability and expected

delays. These policies in Edge Orchestrator can effectively handle dynamic task offloading at runtime and take real-time utilization and job characteristics into account. It also provides an opportunity to perform task partitioning and accelerate subsequent executions through several Edge-Cloud nodes in parallel. Additionally, individual Edge infrastructures can be coordinated by Orchestrator.

4.2. Offloading Strategies

An effective offloading strategy is one of the critical phases for feasible IoT architectures, since it directly affects the IoT system's availability, reliability, stability and so forth. Thus, we introduced several customized strategies to handle various requirements for IoT latency-sensitive applications. These strategies can be deployed into other existing IoT systems compatibly.

4.2.1. Static-Ratio Policy

This policy makes a decision on task offloading based on a static ratio, such as task percentage and time interval. If implemented in OE, it has similar behavior with LC, since the adopted policy of LC is of the equivalent logic.

Algorithm 1 describes the fixed task percentage of dispatching threshold. (Line 1–3) illustrates the conditions on dispatching a new task $task_{\text{new}}$ into a particular Edge node $edge_{\text{cur}}$, which is that the available resources of the Edge node $edge_{\text{cur}}.res$ can meet the task request $task_{\text{new}}.req$, and the percentage of tasks offloaded into the currently selected Edge node $edge_{\text{cur}}.ratio$ is not over the threshold $\mathcal{T}_{\text{ratio}}$ defined previously. Other than that, the task needs to be offloaded into Cloud $cloud_{\text{entrance}}$ to execute even if the Edge node is unoccupied at present (Line 6).

Algorithm 1 Static Ratio Policy

Require: $edge_{\text{cur}}$.ratio: the latest ratio of tasks that have been offloaded into the current edge node.
Ensure: $edge_{\text{cur}}$ or $cloud_{\text{entrance}}$

1: **if** $edge_{\text{cur}}.\text{res} \geq task_{\text{new}}.\text{req}$ **and** $edge_{\text{cur}}.\text{ratio} \leq \tau_{\text{ratio}}$ **then**
2: $edge_{\text{cur}}.\text{res} \leftarrow edge_{\text{cur}}.\text{res} - task_{\text{new}}.\text{req}$
3: $task_{\text{new}}.\text{location} \leftarrow edge_{\text{cur}}.\text{address}$
▷dispatch $task_{\text{new}}$ into Edge node $edge_{\text{cur}}$
4: **return** $edge_{\text{cur}}$
5: **else**
6: $task_{\text{new}}.\text{location} \leftarrow cloud_{\text{entrance}}.\text{address}$
▷offload $task_{\text{new}}$ into Cloud
7: **return** $cloud_{\text{entrance}}$
8: **end if**

In the background, numerous IoT applications stress the shortage of resources in mobile devices and Edge nodes. This policy plays an efficient role in coping with resource imbalance issues between Edge-Cloud nodes, in which there are equal amount of tasks and performance like hardware, because it can ensure that each unit of resource handles approximately equivalent tasks to achieve fairness of resource occupation. Simultaneously, its overhead is notably cheap and neglected compared with original systems. Thus it is widely adopted into down-level fundamental infrastructures.

4.2.2. Least-Load Policy

This policy makes decisions on dispatching tasks based on the runtime load information in order to select the least-load machine for offloaded tasks. Most systems fully support this policy at a higher priority due to its simplicity of logic and the feasibility of implementation. It is well suited for the medium situation, in which the number of applications and the execution time of tasks are both moderate.

The principal procedure is described in Algorithm 2. At first, it chooses the least load machine among entire accessible virtual and physical machines in Edge and Cloud at present (Line 2), and if the selected optimal machine m_{opt} has enough available resources to satisfy the task, the decision process ends with dispatching the task $task_{\text{new}}$ into the machine m_{opt} (Line 3–6). Otherwise, to eliminate the above machine m_{opt} from the whole set $\mathcal{M}$ (Line 8) and trigger a new-loop selection (Line 1–10).

Algorithm 2 Least Load Policy

Require: $\mathcal{M}$: total Edge-Cloud machines with load metric
Ensure: m_{opt}: optimal machine with least load

1: **while** $task_{\text{new}}$.location is **null do**
2: $m_{\text{opt}} \leftarrow \{m \mid load_m \leq load_{m_i}, m_i \in \mathcal{M}\}$
3: **if** m_{opt}.res $\geq task_{\text{new}}$.req **then**
4: m_{opt}.res $\leftarrow m_{\text{opt}}$.res $- task_{\text{new}}$.req
5: $task_{\text{new}}$.location $\leftarrow m_{\text{opt}}$.address ▷dispatch $task_{\text{new}}$ into machine m_{opt}
6: **return** m_{opt}
7: **else**
8: $\mathcal{M} \leftarrow \mathcal{M} - m_{\text{opt}}$
9: **end if**
10: **end while**

However, because it heavily relies on a high-performance monitor, which needs to track the load metric of entire virtual or physic machines located at Edge and Cloud in real-time, once the monitor crashed or performed weakly, such as the poor accuracy of monitoring data, this policy can not work desirably. Besides, it is also unacceptable to inject too much overhead while collecting runtime values of various metrics. Thus, this policy has strict requirements on (1) network to ensure the stable transmission of monitoring information and (2) specialized hardware to ensure the metrics' collection in a cheap consumption by calling hardware interfaces.

Anyway, this policy is always the best and first choice worth trying, while facing an optimization problem of scheduling various tasks with diverse characteristics. It usually simplifies a complex packing problem in practice and performs effectively.

4.2.3. Probabilistic Policy

In this policy, the behavior of IoT tasks was treated as Markov Process, one of the stochastic processes, in which a series of events in a probabilistic manner depends only on the state attained in the previous event. Its mathematical formula can be described as: $P(X_{n+1} = x|X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = P(X_{n+1} = x|X_n = x_n)$.

The policy aims to optimize the task's allocation according to their conditional probability between each pair of tasks. However, it is infeasible to define a rigorous equation describing the occurrence models of IoT tasks in a slightly more succinct way because of its complexity and uncertainty. Intuitively, one of the methods coping with modeling $P(X_{n+1} = x|X_n = x_n)$ is based on the Law of large numbers, in which the value of $P(X_{n+1} = x|X_n = x_n)$ can be estimated by the observations. For example, t_1 donates task one, t_2 task two and t_3 task three, and what the procedure is to use the total number of t_1 following t_2 to approximate $P(t_1|t_2)$, denoting the occurrence probability of t_1 if t_2 has occurred and also $P(t_1|t_3), P(t_2|t_1), P(t_2|t_3), P(t_3|t_1), P(t_3|t_2)$.

The probabilistic policy, based on the Markov Process, also aims to make better decisions on selecting the optimal machine, which is described in Algorithm 3. $\mathcal{M}_{\mathcal{T}}^{\mathcal{P}}$ implies a matrix of conditional probabilities, including the occurrence possibility of each pair of t_i and t_j, and its every value is evaluated by counting appearance in the past (Line 1). The next stage is to pick potential tasks following the current $task_{\text{new}}$ (Line 2–5) and order

them together according to multiple levels, such as delay-sensitivity, edge-requirement, cloud-requirement and so forth (Line 6–7). The final phase is to pick out tasks one by one as a previously processed order and find an optimal position for them by considering network transmission, utilization, etc. However, notice, only the true task $task_{\text{new}}$ needs to be dispatched into its related destination and others are only involved in resource allocation but no dispatching (Line 8–21). This method provides us an opportunity to achieve the overall optimization by taking future possible tasks into account, instead of the current state.

Algorithm 3 Probabilistic Policy

Require: $\mathcal{M}_{\mathcal{T}}^{\mathcal{P}}$: a matrix of conditional probabilities.
Ensure: m_{opt}: optimal machine with least cost including utilization, transmission, etc. for $task_{\text{new}}$

1: $\mathcal{M}_{\mathcal{T}}^{\mathcal{P}} \leftarrow \{P(t_i|t_j) = \frac{Count_{t_i|t_j}}{Count_{t_j}} \mid t_i, t_j \in \mathcal{T}\}$
▷outputs a matrix consisting of conditional probabilities of every pair of tasks, depending on counting occurrence of $t_i|t_j$ and t_j
2: $\mathcal{T}' \leftarrow \phi$
3: **for all** $P \in \mathcal{M}_{\mathcal{T}}^{\mathcal{P}}$ **do**
4: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \cup \{t \mid P(t \mid task_{\text{new}}) \geq \tau_{\text{probability}}\}$
5: **end for**
6: $\mathcal{T}'' \leftarrow \mathcal{T}' \cup task_{\text{new}}$
7: multilevel sorting($\mathcal{T}''$)
▷sort $\mathcal{T}''$ as delay-sensitivity, edge-util, cloud-util, input-size...
8: **for all** $t \in \mathcal{T}''$ **do**
9: $\quad$ **let** $m_{\text{opt}} \leftarrow$ **null**
10: $\quad$ **if** t is delay-sensitive **then**
11: $\quad\quad m_{\text{opt}} \leftarrow edge_{\text{least-cost}}$
▷an edge node with the least cost for task t
12: $\quad$ **else**
13: $\quad\quad m_{\text{opt}} \leftarrow cloud_{\text{least-cost}}$
▷a cloud node with the least cost for task t
14: $\quad$ **end if**
15: $\quad$ **if** t is $task_{\text{new}}$ **then**
16: $\quad\quad t$.location $\leftarrow m_{\text{opt}}$.address ▷dispatch t into m_{opt}
17: $\quad\quad$ **return** m_{opt}
18: $\quad$ **else**
19: $\quad\quad$ virtually delete allocated resources
20: $\quad$ **end if**
21: **end for**

4.3. Measurement of Performance

4.3.1. End-to-End Service Time

In this section, we split the holistic end-to-end service time into separate measurable segments. Generally, an Edge-Cloud system consists of IoT devices, Edge nodes and Cloud nodes, as shown in Figure 2. The computation (such as CPU and GPU, etc.) and communication network bandwidth are two assignable types of resources. IoT devices are interconnected and communicate with Edge nodes through the Wide Area Network (WAN). Each spot is typically covered by a Wi-Fi access point and devices are connected to related WLAN (Wireless Local Area Network). Once they enter into the corresponding zone, mobile devices keep sending requests to Edge nodes. If a request is picked and transferred to either Edge nodes or Cloud nodes, WAN or Metropolitan Area Network (MAN) provided

by the Wi-Fi access point is eventually used. Data being transferred (upload/download) or being processed (input/output) can be of different sizes and sometimes unpredictable lengths. For instance, workloads are likely to be diverse, highly depending on the IoT tasks' demands in terms of functional requirements such as CPU number of cores, power supplier, the amount of communication bandwidth and nonfunctional requirements such as security and access control.

4.3.2. Measurement

In our context, presumably, *end-to-end service time* for a single task can be approximately calculated by the sum of computation time and communication time. In particular, computation time can be further divided into task queuing time Q and actual processing time P. Task queuing time consists of the queuing in Edge node, Edge node in-between and Cloud node, i.e., $Q \leftarrow (Q_{edge}, Q_{xedge}, Q_{cloud})$, while the actual processing time is represented as $P \leftarrow (P_{edge}, P_{xedge}, P_{cloud})$. Similarly, communication time is composed of transmission delay and propagation delay for data upload/download. Specifically, the transmission delay is the time required to push data into the link and propagation delay is the time required to send data from sender to receiver. From an upload and download perspective, we define the communication time consumption into upload time $U \leftarrow (U_{WLAN}, U_{MAN}, U_{WAN})$ and download time $D \leftarrow (D_{WLAN}, D_{MAN}, D_{WAN})$ to depict the produced overall delay among three tiers in Edge-Cloud architecture.

Intuitively, Edge nodes have relatively limited computation resources but provision network connections with minimal networking delays. In contrast, Cloud nodes are capable of handling and processing big data by leveraging the Cloud resource pool, resulting in a shortened processing time. However, offloading to Cloud has to experience longer communication times. The communication time is highly dependent upon whether the task will be offloaded and processed in Edge node, collaborative Edge nodes or in Cloud node. In-between, communication to the nearest Edge nodes has the smallest communication time due to WLAN network protocol.

Therefore, to model computation and communication time in Edge-cloud system we need to determine whether the task will be processed in connected Edge, another nearby Edge or the cloud. The computational resources at the Edge nodes have the same functionality, but they differ from the cloud in their capacity because the Edge nodes are a small computational resource close to end devices. The following presents the latency of offloading tasks with the edge-cloud system:

- **Latency to local Edge:** the total service time for offloading tasks to connected Edge node in Edge-Cloud system is the sum of: time to send task data to the Edge server, queuing to process the task in the Edge server, the processing time of the task and the time to send the output to the IoT devices. We consider the connection to the local Edge will be within WLAN, as presented in Equation (1).

$$L_{edge} = \sum U_{WLAN} + Q_{edge} + P_{edge} + D_{WLAN} \tag{1}$$

- **Latency to collaborative Edge:** the total service time to process an offloading task in a collaborative Edge can be calculated by the summation of time of uploading task data to local Edge via WLAN, uploading data to the collaborative Edge via MAN, queuing time, processing time and time for downloading the output to IoT device via MAN and WLAN, as presented in Equation (2).

$$L_{xedge} = \sum U_{WLAN} + U_{MAN} + Q_{xedge} + P_{xedge} + D_{MAN} + D_{WLAN} \tag{2}$$

- **Latency to central Cloud:** in order to calculate service time for an offloading task in the cloud, we should consider the network delay through WLAN, MAN, WAN as well as the queuing time and processing time, as presented in Equation (3).

$$L_{cloud} = \sum U_{WLAN} + U_{MAN} + U_{WAN} + Q_{cloud} + P_{cloud} + D_{WAN} + D_{MAN} + D_{WLAN} \quad (3)$$

5. Evaluation

5.1. Simulation Set-Up

In practice, making experiments on separate Edge-Cloud architectures is not an easily-feasible procedure due to the variety of frameworks, the diversity of mobile end devices and applications, the complexity of computing services, the compatibility of communication protocols, etc. Since Edge nodes and IoT devices have not been conveyed at this point, EdgeCloudSim [35] is a simulation environment, as a practically feasible experiment platform, which supports to mimic diverse IoT scenarios and Edge-Cloud architectures. Additionally, the EdgeCloudSim was implemented by adjusting the CloudSim [36].

In the Edge-Cloud environment, there are a number of IoT/ mobile devices that have a number of applications. These applications consist of different tasks that require to be processed in the Edge-Cloud resources. Edge nodes are distributed closer to end devices, and we assume each edge node covers a specific area. IoT devices connect to the nearest edge node through WLAN and then can send the offloaded tasks. We assume that each node has a node manager and all edge nodes are managed by the *Edge Orchestrator*.

EdgeCloudSim provides sufficient models to represent some specific situations. For example, the queuing model is introduced to represent the several kinds of delay taking place in the WLAN, MAN and WAN, mobile devices and even CPUs of VMs. Thus, experiments of this work are practically finished within this simulation to investigate and evaluate the performance of the LC three-tier architecture and OE three-tier architecture for several IoT workloads as well as different strategies.

Table 1 shows the key parameters of our simulation experiments. Each architecture scheme has one centralized Cloud which is connected to three distributed Edge nodes. Each Edge node handles a certain number of IoT end devices located in the area of the corresponding Edge node. For each experiment, the number of mobile devices is increased by 200 from 100 to 700 devices in order to explore the relationship between the capacity of Edge node and end-to-end service time of applications in different architectures and strategies. IoT devices generate a number of tasks (also called IoT workload in our case) as a Poisson distribution, thus if the number of IoT devices increases, the number of IoT workloads will also increase.

5.2. Methodology

IoT workloads usually vary the degree of resource reliance between light and heavy. Namely, it ranges from low computation and communication demands such as health-care applications to high computation and communication demand such as video online gaming. The numerical configuration is an important step and the following steps are inspired and motivated by the uncertainty associated with the unpredictable workload discussed in [6].

To deal with applications that might be encountered in practice, we define the configuration of tasks varying communication bandwidth demand from 0.25 MB to 1 MB as an increased step of 0.25 MB, and doubling computation demand starting from 500 MIPS to 4000 MIPS, as shown in Table 2. This configuration results in 16 distinct combinations, labeled as diverse applications/workloads. The symbol *App(x,y)* denotes an application requiring about x MB network bandwidth and occupying around y Million Instruction Per Second (MIPS) CPU computation to execute.

Table 1. Simulation key parameters.

Parameter	Value
Poisson Interarrival Time of Tasks (second)	3
Simulation time (hours)	2
Warm up period (s)	3
Number of repetitions	5
Number of Edge Nodes	3
Number of hosts per edge nodes	2
Number of VMs per Edge server/cloud	4/ not limited
VM speed (MIPS) per Edge server/cloud	10,000
Minimum number of end devices	100
Maximum number of end devices	1000
Active/idle for end devices period (s)	45/15
Probability of Offloading to Cloud (%)	10
Average Data Size for Upload/Download (KB)	1500/15
Average Task Size (MI)	1500
WAN/WLAN Bandwidth (Mbps)	20/300
WAN Propagation Delay (ms)	100

Table 2. Workload examples and corresponding configurations of Central Processing Unit (CPU) speed and bandwidth.

	500 MIPS	**1000 MIPS**	**2000 MIPS**	**4000 MIPS**
0.25 MB	App1	App2	App3	App4
0.5 MB	App5	App6	App7	App8
0.75 MB	App9	App10	App11	App12
1 MB	App13	App14	App15	App16

Section 5.3 investigates how the size of the demanded resources (CPU and network) of IoT tasks influence the overall service time under diverse mobile end devices. Furthermore, Section 5.4 validates (1) **the scalability** of two architecture schemes by tuning the number of mobile devices from 1 to 1000 by 200, and (2) **the availability and performance** of three offloading strategies (*static ratio policy, least load policy and probabilistic policy*) in the edge-cloud environment for various applications with diverse degrees of reliance on CPU and network resources. Finally, the aggregated results of all submitted applications are presented to reveal the holistic effects of different parameters on performance.

5.3. Service Time vs. Resource Requirements

As depicted in Figure 3, no matter how many mobile end devices, the average service time of IoT applications shows a corresponding increase along with the increment of its CPU requirements, and the fewer end devices, the more obvious fluctuation. For example, the end-to-end service time of *App(1,4k)* is about four times *App(1,500)* when the number of mobile end devices equals to 100 but only nearly two times when the number is 700.

Intuitively, the reason is that computation resources are severely limited, and when the demand for CPU increases, the time of waiting and processing in CPU will also rise correspondingly. However, once the number of tasks increases to a certain value, the conflict of CPU resource (means Clock Cycles) will increase slowly as it is near to the maximum of CPU capacity, and the critical threshold of service time will also be transferred to another factor.

Besides, the increment of service time in OE architecture is slightly more serious than that in LC as the devices increase, because the Edge orchestrator needs to consume more time on allocation and dispatching when more IoT tasks are waiting to be coordinated between diverse resource entities. When the whole of CPU cores are pinned

and utilized by colocated IoT tasks, the difference between LC and OE architectures will tremendously enlarge.

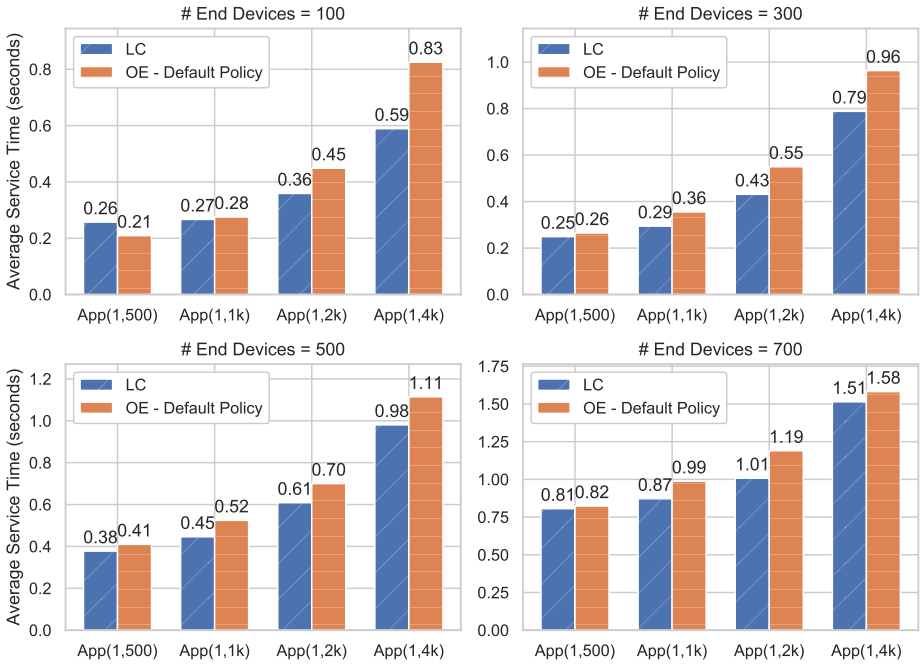

Figure 3. Impact on varying CPU demands.

In contrast, as shown in Figure 4, when the bandwidth demand of a task varies, the service time only slightly increases due that the network bandwidth is not a critical limit for IoT tasks in current experiments. In other words, network resource or performance is notably sufficient to handle nearly all IoT tasks. Notice, when the number of end devices is close to 700, the increment becomes obvious and an efficient assignment of network resources will play a meaningful role in service time and quality.

Figure 4. Impact on varying bandwidth demands.

In effect, LC just purely finds a single node to realize the offloading. Once the task is offloaded to a particular node, it cannot be further moved. In addition, if no node is capable of holding IoT tasks, the LC system has to wait until more available resources are released and make rooms for the pending tasks. This procedure, instead, takes a longer time compared with the OE scheme which can easily deal with dynamic task offloading at runtime and perform task partitioning and parallel offloading to multiple destinations.

To summarize, the computational requirements exerts more influence on the IoT application's performance compared to the communication requirement. However, when the system scales up to accommodate more IoT devices, communication bandwidth will turn to be the dominant resource and becomes the essential factor that will directly impact the overall performance. On the other hand, for small tasks and a small-scale offloading scenario, loosely-coupled offloading approaches can deal with task redistribution requirements smoothly. Nevertheless, when IoT tasks become bigger in terms of resource requests or when the offloading frequently occurs, orchestration is a necessary procedure to encompass optimized solutions under different constraints for optimal offloading placement.

5.4. Scalability and Availability

This section will validate how service time changes along with the number of IoT devices under two architectures and three offloading strategies. As observed from Figure 5, when resource requests from IoT tasks are relatively small (e.g., App(0.25,500)), the overall service time is insensitive to the growing number of end devices even under different architectures and strategies. The reason is that no matter where the final destination of offloading is, the current resources in Edge-Cloud nodes are sufficient to handle these light IoT tasks offloaded, thus the offloading procedure does not make significant contributions.

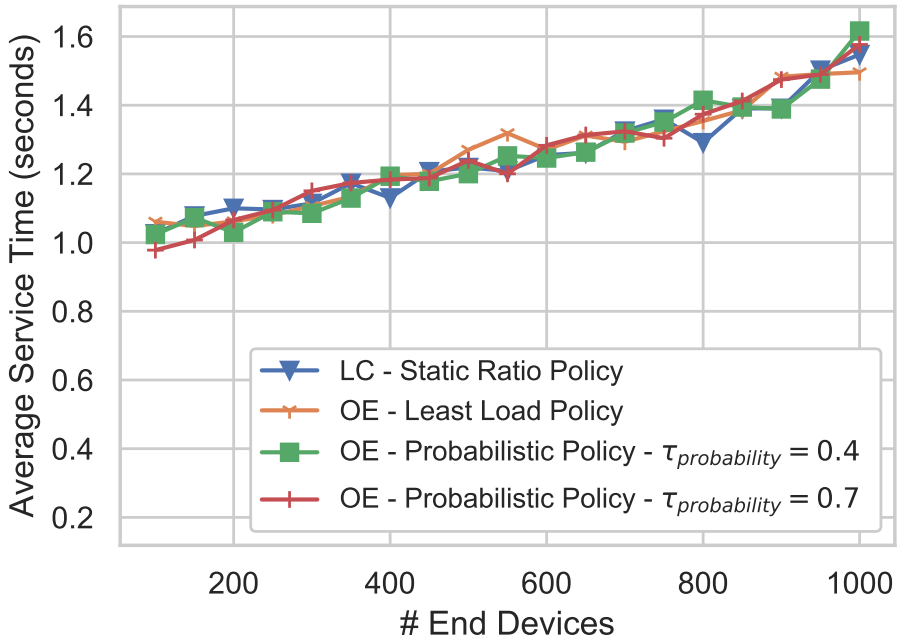

Figure 5. Service time of various application characteristics under different offloading strategies.

Edge Orchestrator can effectively seek placement for the incoming tasks based on a defined strategy by taking specific characteristics of IoT tasks into account. In comparison, to deal with App(1,4k), the tasks with the largest resource requests, both LC and OE experience a soaring increase of service time along with the incremental number of end devices. Obviously, Edge Orchestrator in the OE scheme needs to consume a slightly longer time for coordinating resources among Edge-Cloud nodes and make placement decisions subsequently. We can observe a similar phenomenon when comparing App(1,500) with App(1,4k) shown in Figure 5.

Herein, the above analysis indicates that different applications have diverse sensitivity to the device scale. IoT service providers need to coordinate and specify the proper resource according to the changing configuration of Edge-Cloud infrastructure for various IoT applications.

Offloading strategies, as a core phase, are making a vital impact on the appearance of offloading. As shown in Figure 6, the service time of adopting Probabilistic Policy fluctuates obviously but approximately hovers around the Least Load Policy, or sometimes in between Static Ratio Policy and Least Load Policy. This trend means that the stability of Probabilistic Policy is yet weaker than that of Least Load Policy and still needs further improvement on task dispatching.

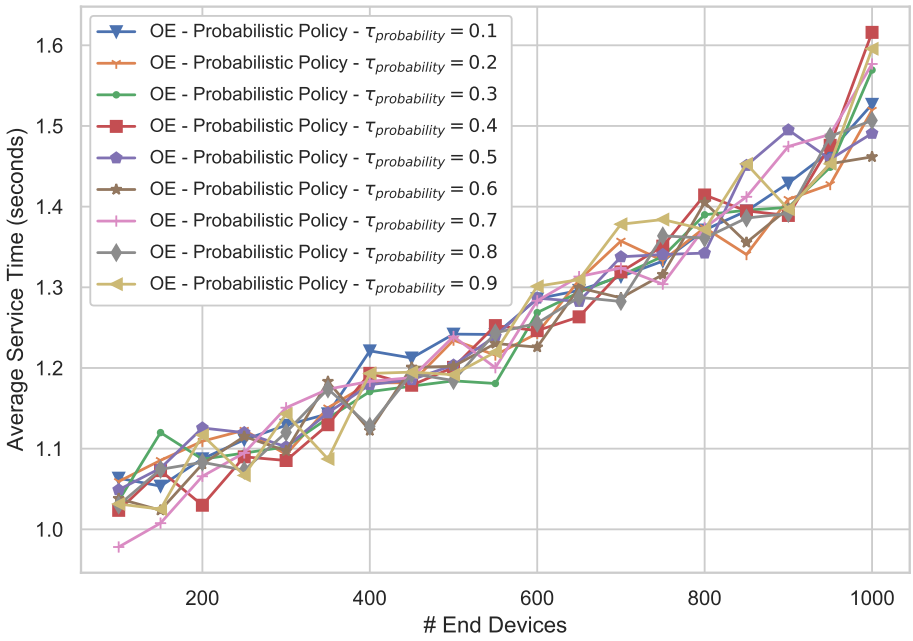

Figure 6. Hyperparameter tuning.

The Least Load Policy performs stably and effectively because it wholly relies on runtime load information. The CPU load is one of the most critical metrics for task executions, hence why the policy presenting large profits for an end-to-end service time is understandable.

In contrast, the Probabilistic Policy needs to evaluate the occurrence probability of the following tasks according to the past event rate and introduce the manner of resource reservation for might-coming workloads. Therefore, this approach results in the determined placement for current tasks not being optimal or near-optimal while dispatching tasks. Unfortunately, if the forecast on the next following task is incorrect, the reservation becomes a kind of resource waste. Introducing overbooking technology [37] can cope with the problem and improve the tolerance of offloading architecture for inaccurate estimation.

Manifestly, as the number of end devices increments and the resource requirements of IoT tasks rises, the OE architecture becomes more and more effective and efficient, and the offloading strategies start to make more contributions on task dispatching. A meaningful conclusion is that LC architecture (also Static Ratio Policy) is much more suitable for a simple IoT scenario, in which the number of IoT devices is relatively less. However, when the resource request comes larger (e.g., App(1,4k) or more), the LC scheme, otherwise, lengthens the time consumption, probably due to the static procedure of offloading to a single node, which ignores the present status of Edge-Cloud nodes and individual requirements of IoT tasks. Only an Edge Orchestration considering complex constraints can help resolve this problem.

6. Research Limitations

There are several promising research directions that are related to this work and need to be addressed as highlighted below.

- This work handles the offloading strategies of independent tasks; however, task dependency plays an essential factor to affect the decision of offloading tasks. Thus, this work can be extended to consider task dependency in the process of scheduling offloading tasks. Task dependency and the intercommunication between tasks can be represented as a DAG, which can be modeled within the proposed approach to enhance the overall service time of latency-sensitive applications.
- Another complement work that will enhance the work presented is to predict the behaviour of latency-sensitive applications [25,26]. The prediction can be in several areas such as predicting the volume of incoming tasks, predicting the users' mobility which could help to determine their locations. Therefore, it would help the resource manager to prepare the required resources in advance and avoided any performance degradation. This extension would be useful when scheduling offloading tasks in order to minimise the overall service time.
- In addition, the proposed approach considers three customized offloading strategies in order to handle various requirements for IoT latency-sensitive applications based on CPU speed and bandwidth. Thus, the approach could be extended to consider computational resources such as different GPUs and FPGAs since there are many applications for AR/VR and video gaming that require intensive computations in order to process their tasks in the Edge-Cloud environment.

7. Conclusions and Future Work

Cloud computing is evolving rapidly toward the fundamental infrastructure and facilitating the future of IoT to potentially connect billions of edge devices. Efficient coordination mechanisms and task offloading algorithms are leveraged to enable mobile devices and Edge-Cloud to cooperatively work. In this paper, we have analyzed the impact on the performance of IoT applications under different offloading architecture schemes and strategies as well as examine their effectiveness, while varying the number of IoT devices and requested resource changes. Additionally, other main conclusions from our research are summarized as follows:

The presence of scalable resource management is not solely confined to Cloud datacenters and more challenging in IoT service provisioning. They can manifest prominently in any large-scale computing system including Edge computing models that support IoT applications. These systems are particularly susceptible to emergent surges of applications and sudden changes in resources.

Exploiting the latency-based model and performance profiling is meant to reveal the inherent nature of application behavior and root causes of performance interference. In the IoT era, Edge computing is extremely suitable for optimizing and guaranteeing the performance of real-time applications. The owner of IoT applications can benefit from this work to be aware of how applications perform with different IoT deployments schemes and choose the proper architecture given a specific application scenario, which could help to enhance the application performance in the future.

The proposed analysis methodology can be easily extended to underpin IoT applications beyond the single Cloud datacenter. If the system boundaries of interconnected components are constantly changing, it is intuitive to add up new tiers to depict such networked interactions. For specific applications that may leverage and cooperate among *Joint Clouds*, an additional tier to depict the federation of clouds can be also added accordingly.

A future research direction will be to investigate how to autonomously determine the optimal deployment mechanism for a given system context. We are also implementing different offloading algorithms and would like to integrate them into an offloading library, which can be easily adopted by other real-world systems.

Author Contributions: Conceptualization, Data curation, Formal analysis, Methodology, Resources, Software, Visualization and Writing—original draft: J.A. and M.A.; Supervision, Validation, Writing—review and editing M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge the Deanship of Scientific Research, Taibah University, Al-Madinah, Saudi Arabia, for providing research resources and equipment. In addition, the authors would like to thank the Deanship of Scientific Research, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia, for supporting this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. IoT Number Trend. Available online: https://www.businessinsider.com/internet-of-everything-2015-bi-2014-12 (accessed on 13 August 2020).
2. Yousefpour, A.; Fung, C.; Nguyen, T.; Kadiyala, K.; Jalali, F.; Niakanlahiji, A.; Kong, J.; Jue, J.P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. Syst. Archit.* **2019**, *98*, 289–330. [CrossRef]
3. Vermesan, O.; Friess, P.; Guillemin, P.; Gusmeroli, S.; Sundmaeker, H.; Bassi, A.; Jubert, I.S.; Mazura, M.; Harrison, M.; Eisenhauer, M.; et al. Internet of things strategic research roadmap. *Internet Things Glob. Technol. Soc. Trends* **2011**, *1*, 9–52.
4. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
5. Yi, S.; Li, C.; Li, Q. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*; ACM: New York, NY, USA, 2015; pp. 37–42.
6. Shekhar, S.; Gokhale, A. Dynamic resource management across cloud-edge resources for performance-sensitive applications. In Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Madrid, Spain, 14–17 May 2017; pp. 707–710.
7. Satyanarayanan, M.; Bahl, V.; Caceres, R.; Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervas. Comput.* **2009**, *8*, 14–23. [CrossRef]
8. Zhang, Y.; Yang, R.; Wo, T.; Hu, C.; Kang, J.; Cui, L. Cloudap: Improving the qos of mobile applications with efficient vm migration. In Proceedings of the 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, Zhangjiajie, China, 13–15 November 2013; pp. 1374–1381.
9. Toczé, K.; Nadjm-Tehrani, S. A taxonomy for management and optimization of multiple resources in edge computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 7476201. [CrossRef]
10. Premsankar, G.; Di Francesco, M.; Taleb, T. Edge computing for the Internet of Things: A case study. *IEEE Internet Things J.* **2018**, *5*, 1275–1284. [CrossRef]
11. Li, W.; Santos, I.; Delicato, F.C.; Pires, P.F.; Pirmez, L.; Wei, W.; Song, H.; Zomaya, A.; Khan, S. System modelling and performance evaluation of a three-tier Cloud of Things. *Future Gener. Comput. Syst.* **2017**, *70*, 104–125. [CrossRef]
12. Hegyi, A.; Flinck, H.; Ketyko, I.; Kuure, P.; Nemes, C.; Pinter, L. Application orchestration in mobile edge cloud: placing of iot applications to the edge. In Proceedings of the 2016 IEEE 1st International Workshops on Foundations and Applications of Self Systems, Augsburg, Germany, 12–16 September 2016; pp. 230–235.
13. Dinh, T.Q.; Tang, J.; La, Q.D.; Quek, T.Q. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Trans. Commun.* **2017**, *65*, 3571–3584.
14. Du, J.; Zhao, L.; Feng, J.; Chu, X. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Trans. Commun.* **2018**, *66*, 1594–1608. [CrossRef]
15. Liu, J.; Mao, Y.; Zhang, J.; Letaief, K.B. Delay-optimal computation task scheduling for mobile-edge computing systems. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 1451–1455.
16. Rodrigues, T.G.; Suto, K.; Nishiyama, H.; Kato, N. Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control. *IEEE Trans. Comput.* **2016**, *66*, 810–819. [CrossRef]
17. Wei, X.; Tang, C.; Fan, J.; Subramaniam, S. Joint Optimization of Energy Consumption and Delay in Cloud-to-Thing Continuum. *IEEE Internet Things J.* **2019**, *6*, 2325–2337. [CrossRef]
18. Yang, B.; Chai, W.K.; Pavlou, G.; Katsaros, K.V. Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud. In Proceedings of the 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, 3–6 October 2016; pp. 136–141.

19. Zeng, D.; Gu, L.; Guo, S.; Cheng, Z.; Yu, S. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Trans. Comput.* **2016**, *65*, 3702–3712. [CrossRef]
20. Wang, S.; Zafer, M.; Leung, K.K. Online placement of multi-component applications in edge computing environments. *IEEE Access* **2017**, *5*, 2514–2533. [CrossRef]
21. Tärneberg, W.; Mehta, A.; Wadbro, E.; Tordsson, J.; Eker, J.; Kihl, M.; Elmroth, E. Dynamic application placement in the mobile cloud network. *Future Gener. Comput. Syst.* **2017**, *70*, 163–177. [CrossRef]
22. Fan, Q.; Ansari, N. Application aware workload allocation for edge computing-based IoT. *IEEE Internet Things J.* **2018**, *5*, 2146–2153. [CrossRef]
23. Roy, D.G.; De, D.; Mukherjee, A.; Buyya, R. Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. *J. Supercomput.* **2017**, *73*, 1672–1690. [CrossRef]
24. Mahmud, R.; Ramamohanarao, K.; Buyya, R. Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions. *ACM Comput. Surv.* **2020**, *53*, 1–43. [CrossRef]
25. Ijaz, M.F.; Attique, M.; Son, Y. Data-Driven Cervical Cancer Prediction Model with Outlier Detection and Over-Sampling Methods. *Sensors* **2020**, *20*, 2809. [CrossRef]
26. Ali, F.; El-Sappagh, S.; Islam, S.R.; Kwak, D.; Ali, A.; Imran, M.; Kwak, K.S. A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion. *Inf. Fusion* **2020**, *63*, 208–222. [CrossRef]
27. Wen, Z.; Yang, R.; Garraghan, P.; Lin, T.; Xu, J.; Rovatsos, M. Fog orchestration for internet of things services. *IEEE Internet Comput.* **2017**, *21*, 16–24. [CrossRef]
28. Sharma, S.K.; Wang, X. Live data analytics with collaborative edge and cloud processing in wireless IoT networks. *IEEE Access* **2017**, *5*, 4621–4635. [CrossRef]
29. Yousefpour, A.; Ishigaki, G.; Jue, J.P. Fog computing: Towards minimizing delay in the internet of things. In Proceedings of the 2017 IEEE international conference on edge computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 17–24.
30. Wang, N.; Varghese, B.; Matthaiou, M.; Nikolopoulos, D.S. ENORM: A framework for edge node resource management. *IEEE Trans. Serv. Comput* **2017**, *13*, 1086–1099. [CrossRef]
31. Wang, J.; Pan, J.; Esposito, F.; Calyam, P.; Yang, Z.; Mohapatra, P. Edge cloud offloading algorithms: Issues, methods, and perspectives. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 2. [CrossRef]
32. Farris, I.; Taleb, T.; Flinck, H.; Iera, A. Providing ultra-short latency to user-centric 5G applications at the mobile network edge. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3169. [CrossRef]
33. Li, C.; Tang, J.; Tang, H.; Luo, Y. Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment. *Future Gener. Comput. Syst.* **2019**, *95*, 249–264. [CrossRef]
34. Mahmud, R.; Koch, F.L.; Buyya, R. Cloud-fog interoperability in IoT-enabled healthcare solutions. In Proceedings of the 19th International Conference on Distributed Computing and networking, Varanasi, India, 4–7 January 2018; p. 32.
35. Sonmez, C.; Ozgovde, A.; Ersoy, C. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3493. [CrossRef]
36. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software Pract. Exper* **2011**, *41*, 23–50. [CrossRef]
37. Sun, X.; Hu, C.; Yang, R.; Garraghan, P.; Wo, T.; Xu, J.; Zhu, J.; Li, C. ROSE: Cluster resource scheduling via speculative over-subscription. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–5 July 2018; pp. 949–960.