Towards Optimal System Deployment for Edge Computing: A Preliminary Study

Conference Paper · October 2020

DOI: 10.1109/ICCCN49398.2020.9209754

5 authors, including:

Dawei Li
Montclair State University, Montclair, NJ, USA
30 PUBLICATIONS **195** CITATIONS
SEE PROFILE

Chigozie Ifunanya Asikaburu
Montclair State University
1 PUBLICATION **1** CITATION
SEE PROFILE

Boxiang Dong
Montclair State University
46 PUBLICATIONS **95** CITATIONS
SEE PROFILE

Sadoon Azizi
University of Kurdistan
23 PUBLICATIONS **38** CITATIONS
SEE PROFILE

Towards Optimal System Deployment for Edge Computing: A Preliminary Study

Dawei Li*, Chigozie Asikaburu*, Boxiang Dong*, Huan Zhou†, and Sadoon Azizi‡
*Department of Computer Science, Montclair State University, Montclair, NJ, 07043, USA
†College of Computer and Information Technology, China Three Gorges University, Yichang, Hubei, 443002, P. R. China.
‡Department of Computer Engineering and IT, University of Kurdistan, Sanandaj, Kurdistan Province, Iran.

dawei.li@montclair.edu, asikaburug1@montclair.edu, dongb@montclair.edu, zhouhuan117@gmail.com, and s.azizi@uok.ac.ir

***Abstract*—In this preliminary study, we consider the server allocation problem for edge computing system deployment. Our goal is to minimize the average turnaround time of application requests/tasks, generated by all mobile devices/users in a geographical region. We consider two approaches for edge cloud deployment: the flat deployment, where all edge clouds co-locate with the base stations, and the hierarchical deployment, where edge clouds can also co-locate with other system components besides the base stations. In the flat deployment, we demonstrate that the allocation of edge cloud servers should be balanced across all the base stations, if the application request arrival rates at the base stations are equal to each other. We also show that the hierarchical deployment approach has great potentials in minimizing the system's average turnaround time. We conduct various simulation studies using the CloudSim Plus platform to verify our theoretical results. The collective findings trough theoretical analysis and simulation results will provide useful guidance in practical edge computing system deployment.**

Index Terms*—Edge computing, edge cloud, flat deployment, hierarchical deployment, average turnaround time, CloudSim Plus.

I. Introduction

Our current era is witnessing a proliferation of services and applications that are provided at the edge of the Internet. The Internet edge includes traditional access networks, such as Wireless Local Area Networks (WLAN) and Wireless Metropolitan Area Networks (WMAN), as well as various emerging systems, such as Internet of Things (IoT) systems [1]. Edge applications can be computation-intensive and/or communication-intensive. However, mobile devices (i.e., mobile phones, tablets, and IoT devices, etc.) are still limited in various aspects, such as storage size, computation capacity, communication bandwidth, and battery life. Cloud computing has been applied to facilitate mobile applications by processing workloads offloaded from mobile devices in resource-rich clouds backed by remote data centers. However, various issues arise due to long latency and limited bandwidth from mobile devices to remote clouds.

Recently, the *edge computing* paradigm [2], [3], which extends cloud computing to the edge of the Internet, has been proposed to support mobile applications. By running applications on relatively small computing systems deployed at the Internet edge, edge computing allows users to exploit computing power outside of the mobile devices without incurring long access delays to remote clouds. Several other terms have been used, such as fog computing, cloudlets, edge-centric computing, mobile edge computing, etc [4]–[8]. In our discussion, we will consistently refer to them as *edge computing*. The small computing systems deployed at the Internet edge will be called *edge clouds*. Some edge computing system models allow mobile devices to function as servers and accept workloads offloaded from other mobile devices [9], [10]; although a promising trend in edge computing, we do not target such systems. In this paper, we assume that mobile devices can only be clients, and they offload computation-intensive application requests/tasks to edge clouds for processing.

A. Motivation and Related Work

In edge computing, the system deployment problem is a fundamental and important one, which has been considered by various existing works [11]–[13]. We notice that most of them have the following limitations.

First, the system model for an edge cloud is largely simplified. In [13], the numbers of servers in all the edge clouds are fixed and equal to each other. In [11] and [14], the numbers of servers in the edge clouds are not the same; however, they are still fixed; the problem is to derive the edge cloud placement given some capacity constraints to minimize the edge cloud access delay, while the queueing time and execution time of application requests are not considered.

Second, most existing works assume that edge clouds should co-locate with Base Stations (BSs) [11], [12], [14]–[16]. This system deployment is usually referred to as the *flat deployment* [17]. The flat deployment has an important drawback: when a local edge cloud does not have enough resources for mobile applications, traditional remote clouds will serve as the direct backup, in which case, long access delays to remote clouds will be incurred. In the real world, an edge system usually features a hierarchical architecture [18]. For example, in a WMAN, BSs first connect to aggregation nodes, which are then connected to core nodes. Other edge systems, such as WiFi networks, WiMax networks, and IoT systems, may also feature similar hierarchical architectures. Edge clouds can thus be deployed along with various system components at various locations. When the lowest level edge clouds cannot provide enough resources, we can resort to edge clouds at a higher level. Such a deployment scheme is usually referred to as the *hierarchical deployment* [17], [19]. Intuitively, a hierarchical deployment allows efficient resource aggregation at a higher level within the system, while long

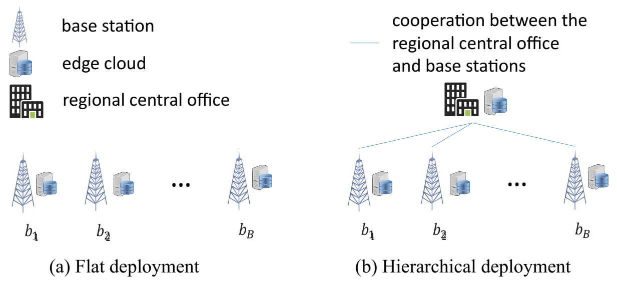

Fig. 1. Flat deployment and hierarchical deployment

access delays to traditional remote clouds can potentially be avoided.

B. Contributions and Paper Organization

Our main contributions in this paper are as follows:

- We consider the server allocation problem in both flat and hierarchical deployment, where each edge cloud is modeled as an M/M/c queueing system. In a flat deployment, we demonstrate that if application request arrival rates at the BSs are equal to each other, the allocation of edge cloud servers across all the BSs should be balanced.
- We demonstrate the potential advantages of the hierarchical system deployment over the flat deployments via theoretical analysis.
- We conduct various simulations using the CloudSim Plus [20] platform to verify our results from theoretical analysis. CloudSim Plus, an independent fork of CloudSim [21], is a discrete-event simulation tool designed for modeling and simulating cloud computing infrastructures and services. It can be easily tweeked and modified to provide high-fidelity simulations for edge computing systems.

This papper is based on our previous work in [22] and [23], which used numerical results from queueing theory to verify our claims. In this paper, we focus on using a practical and high-fidelty cloud and edge computing simulation tool to verify our claims. The rest of the paper is organized as follows. Section II presents the system model. In Section III, we consider the server allocation problem in a flat deployment. In Section IV, we demonstrate the potential advantages of the hierarchical deployment over the flat deployment. Supporting simulations are presented in Section V. Conclusions and future work directions are given in Section VI.

II. System Model

We denote the set of all BSs in the target geographical region as $\mathcal{B} = \{b_1, b_2, \cdots, b_B\}$, where B is the cardinality of the set, i.e., the total number of BSs. We can deploy an edge cloud at each BS to serve the area covered by that BS. For convenience, we will also use b_i to refer to the edge cloud deployed at the BS, the location of the BS, and the user area covered by the BS. Besides the BSs, there exists a regional central office where we can also deploy an edge cloud. If we only deploy edge clouds along with the BSs, we have a *flat deployment* (Fig. 1(a)). If we deploy an edge cloud at the regional central office, we have a *hierarchical deployment* (Fig. 1(b)).

The user area covered by BS b_i has a collective application request/task arrival rate of λ_i following Poisson distribution. Each edge cloud provides processing services to application requests in the form of physical machines or virtual machines [8]. We will consistently refer to them as servers. All the servers in the system have the same execution speed. We assume that all the application requests have an average execution time of $1/\mu$ following exponential distribution. We assumet that when an application requst is being processed, it cannot be preempted. The *turnaround time* for a given application request consists of two components: the *communication delay* (consisting of the time it takes to upload the request to a target edge cloud and the time it takes for the device to download the results from the target edge cloud) and the *service time* (consisting of the queueing time and the execution time at the target edge cloud). Denote d_1 as the average communication delay between a mobile device and its local edge cloud. Denote d_2 as the average communication delay between a mobile device and the edge cloud at the regional central office. We assume that the communication between a mobile device and the regional central office must go through the local BS. Thus, $d_1 < d_2$. In practice, d_1 and d_2 depends on a lot of factors, such as the data size of the requests, the available bandwidth, and the propagation delay, etc. However, we decide not to dive in such details because we will focus on the difference between d_1 and d_2.

The service provider has a budget to deploy N servers across all the edge clouds. For ease of presentation, we refer to the regional central office as the $(B+1)$th edge cloud. Let n_i be the number of servers deployed at b_i, $\forall i = 1, \cdots, B, B+1$. We have $\sum_{i=1}^{B+1} n_i = N$. The system provider wants to derive a server allocation plan in order to minimize the average turnaround time for all application requests generated within the target geographical region.

III. Server Allocation in Flat Deployment

In this section, we consider the server allocation problem in flat deployment, where we do not deploy an edge cloud at the regional central office, i.e., $n_{B+1} = 0$. Given a valid N, we aim to derive an optimal server allocation plan to minimize the average turnaround time of all application requests.

The edge cloud at a BS can be modeled as an M/M/c queue, i.e., the Erlang C model [24]. Let $\rho_i = \frac{\lambda_i}{n_i \mu}$. The probability that an arriving application request is forced to join a queue at b_i (i.e., all n_i servers are occupied) is given by:

$$C(n_i, \lambda_i/\mu) = \frac{\frac{(n_i\rho_i)^{n_i}}{n_i!}\frac{1}{1-\rho_i}}{\sum_{k=0}^{n_i-1}\frac{(n_i\rho_i)^k}{k!} + \frac{(n_i\rho_i)^{n_i}}{n_i!}\frac{1}{1-\rho_i}} = \frac{1}{1 + (1-\rho_i)\frac{n_i!}{(n_i\rho_i)^{n_i}}\sum_{k=0}^{n_i-1}\frac{(n_i\rho_i)^k}{k!}}, \quad (1)$$

which is the Erlang C formula. The average service time (consisting of the queueing time and the execution time and excluding the communication delay) of requests that are processed at b_i is

$$t_{s,i} = \frac{C(n_i, \lambda_i/\mu)}{n_i\mu - \lambda_i} + \frac{1}{\mu}. \quad (2)$$

Thus, the average turnaround time of all application requests generated within the entire region is:

$$t = \frac{\sum_{i=1}^{B}(\lambda_i(t_{s,i} + d_1))}{\sum_{i=1}^{B}\lambda_i}. \tag{3}$$

The optimization problem can be formulated as follows:

$$\begin{aligned} \underset{n_i, \forall i=1,\cdots,B}{\text{minimize}} \quad & t \\ \text{subject to} \quad & n_i \geq n_i^{min}, \forall i = 1, \ldots, B. \\ & \sum_{i=1}^{B} n_i = N. \\ & \text{Equations (1), (2), and (3).} \end{aligned}$$

The complexity of the above optimization problem lies in the following aspects. First, it is an Integer Programming problem with an exponentially large solution search space; these kinds of problems are generally NP-Hard. Second, the closed-form expression of the objective function is quite complex, resulting from the complexity of the Erlang C formula; the fact that $C(n_i, \lambda_i/\mu)$ involves factorial and exponentiation operations on the solution variables (n_i's) makes various relaxation-based convex optimization techniques inapplicable here.

A. The Case With Equal Application Request Arrival Rates

Assuming that $\lambda_1 = \lambda_2 = \cdots = \lambda_B = \lambda$, Equation (2) reduces to:

$$t_{s,i} = \frac{C(n_i, \lambda/\mu)}{n_i\mu - \lambda} + \frac{1}{\mu}. \tag{4}$$

When λ approaches $n_i\mu$, the denominator of the first term will approach 0, making $t_{s,i}$ extremely large. To minimize the average turnaround time, we should increase n_i so that $n_i\mu - \lambda$ should be much larger than 0 for all edge clouds. However, we only have a limited total number of servers, N. If we allocate too many servers for one edge cloud, the number of servers available for other edge clouds will be less. Intuitively, *the allocation of servers to all the edge clouds should be balanced* so that no specific edge cloud experiences an extremely high average service time.

Let n_i and n_j be the numbers of servers deployed at b_i and b_j, respectively, according to a server allocation plan, $\mathcal{S}^1$. Assume that $n_i = n$, and $n_j = n + m$, where $m \geq 2$. For the server allocation plan, $\mathcal{S}^1$, the average service time of requests at b_i and b_j can be calculated as $t_{s,i}^1 = \frac{C(n,\lambda/\mu)}{n\mu-\lambda} + \frac{1}{\mu}$ and $t_{s,j}^1 = \frac{C(n+m,\lambda/\mu)}{(n+m)\mu-\lambda} + \frac{1}{\mu}$, respectively. If there exists such an allocation plan, $\mathcal{S}^1$, we can come up with a new allocation plan, $\mathcal{S}^2$, where the numbers of servers deployed at b_i and b_j are $n_i = n+1$ and $n_j = n+m-1$, respectively. For the new allocation plan, the average service time of requests at b_i and b_j are $t_{s,i}^2 = \frac{C(n+1,\lambda/\mu)}{(n+1)\mu-\lambda} + \frac{1}{\mu}$ and $t_{s,j}^2 = \frac{C(n+m-1,\lambda/\mu)}{(n+m-1)\mu-\lambda} + \frac{1}{\mu}$, respectively.

Conjecture 1. *The average service time of applications from areas i and j under allocation plan $\mathcal{S}^1$ is greater than that under allocation plan $\mathcal{S}^2$, i.e.,*

$$t_{s,i}^1 + t_{s,j}^1 > t_{s,i}^2 + t_{s,j}^2. \tag{5}$$

Illustration: Due to the complexity of the closed-form expressions of $t_{s,i}$ and $t_{s,j}$, a formal proof for this conjecture is hard to derive. The conjecture states that if a server allocation plan ($\mathcal{S}^1$) has an unbalanced server allocation, i.e., the difference, m, in the numbers of servers deployed at b_i and b_j is greater than 2, then, we can reduce the average service time for the application requests from areas i and j with a more balanced server allocation plan ($\mathcal{S}^2$), without affecting the rest of the allocation plan. Thus, we can further reduce the overall average turnaround time.

Here, we verify the conjecture using a basic case where $n = 1$ and $m = 2$. For the unbalanced allocation, we have $t_{s,i}^1 = \frac{1}{\mu-\lambda}$ and $t_{s,j}^1 = \frac{C(3,\lambda/\mu)}{3\mu-\lambda} + \frac{1}{\mu} > \frac{1}{\mu}$ For the balanced allocation, we have $t_{s,i}^2 = t_{s,j}^2 = \frac{C(2,\lambda/\mu)}{2\mu-\lambda} + \frac{1}{\mu} = \frac{4\mu}{4\mu^2-\lambda^2}$. Thus, $t_{s,i}^1 + t_{s,j}^1 - (t_{s,i}^2 + t_{s,j}^2) > \frac{1}{\mu-\lambda} + \frac{1}{\mu} - \frac{8\mu}{4\mu^2-\lambda^2} = \frac{2\mu\lambda(2\mu-\lambda)+\lambda^3}{\mu(\mu-\lambda)(4\mu^2-\lambda^2)} > 0.$ ■

With this conjecture, we can draw the conclusion that, the allocation of servers to all the edge clouds should be balanced in order to minimize the average turnaround time for all application requests when application request arrival rates at all the locations are equal to each other.

IV. Potential Advantages of Hierarchical Deployment over Flat Deployment

In this section, we compare the two deployment approaches and demonstrate the potential benefits of the hierarchical deployment. Before any theoretical and numerical analyses, we would like to point out that hierarchical deployment does not rule out flat deployment. Actually, the flat deployment is a special case of the hierarchical deployment where $n_{B+1} = 0$. Thus, hierarchical deployment allows more possibilities for server allocation and thus, can provide the benefits and advantages in general.

In the following, we use a special case to explicitly demonstrate the advantage of hierarchical deployment over flat deployment. We consider a special case where $\lambda_1 = \cdots = \lambda_B = \lambda$ and the total number of servers is equal to the total number of BSs, i.e., $N = B$. In the *flat deployment* approach, each BS will have an edge cloud each with one server. For each edge cloud queueing system to be stable, we should have $\lambda < \mu$. The average service time of application requests can be calculated as

$$t_s^f = \frac{1}{\mu - \lambda}. \tag{6}$$

The average turnaround time of application requests is $t^f = t_s^f + d_1$.

We then consider the *hierarchical deployment* approach where no edge cloud is co-located with a BS. There is only one edge cloud deployed at the regional central office. It has B servers and can be modeled as an M/M/c queue. Due to the additive property of the Poisson arrival process, the effective request arrival rate at the edge cloud is $B\lambda$. Let $\rho = B\lambda/(B\mu) = \lambda/\mu$. The probability that an arriving application request is forced to join the queue is given by:

$$C(B, B\lambda/\mu) = \frac{\frac{(B\rho)^B}{B!}\frac{1}{1-\rho}}{\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{(B\rho)^B}{B!}\frac{1}{1-\rho}} \tag{7}$$

The average service time of application requests is:

$$t_s^h = \frac{C(B, B\lambda/\mu)}{B\mu - B\lambda} + \frac{1}{\mu} \tag{8}$$

The average turnaround time is $t^h = t_s^h + d_2$.

Intuitively, the probability that a newly arrived request finds all the B servers occupied in the hierarchical deployment is less than the probability that a newly arrived request finds the only one local server occupied in the flat deployment. We have the following theorem.

Theorem 1. *In the above-mentioned scenario, the average service time of a request in the hierarchical deployment is always less than the average service time of a request in the flat deployment.*

Proof: Calculate the difference between t_s^h and t_s^f:

$$\begin{aligned}
t_s^h - t_s^f &= \frac{C(B, B\lambda/\mu)}{B\mu - B\lambda} + \frac{1}{\mu} - \frac{1}{\mu - \lambda} \\
&= \frac{1}{B(\mu-\lambda)} \frac{\frac{(B\rho)^B}{B!}\frac{1}{1-\rho}}{\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{(B\rho)^B}{B!}\frac{1}{1-\rho}} - \frac{\lambda}{\mu(\mu-\lambda)} \\
&= \frac{1}{\mu-\lambda}\left(\frac{\frac{B^{B-1}\rho^B}{B!}\frac{1}{1-\rho}}{\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{B^B\rho^B}{B!}\frac{1}{1-\rho}} - \rho\right) \\
&= \frac{1}{\mu-\lambda}\left(\frac{\frac{B^{B-1}\rho^B}{B!}}{(1-\rho)\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{B^B\rho^B}{B!}} - \rho\right) \\
&= \frac{\frac{B^{B-1}\rho^B}{B!} - \rho\left((1-\rho)\sum_{k=0}^{B-1}\frac{B^k\rho^k}{k!} + \frac{B^B\rho^B}{B!}\right)}{(\mu-\lambda)\left((1-\rho)\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{B^B\rho^B}{B!}\right)}.
\end{aligned}$$

Consider that

$$\sum_{k=0}^{B-1}\frac{B^k\rho^k}{k!} > \sum_{k=0}^{B-1}\frac{B^k\rho^k}{B!} > \frac{B^{B-1}\rho^{B-1}}{B!}. \tag{9}$$

We have

$$\begin{aligned}
t_s^h - t_s^f &< \frac{\frac{B^{B-1}\rho^B}{B!} - \rho\left((1-\rho)\frac{B^{B-1}\rho^{B-1}}{B!} + \frac{B^B\rho^B}{B!}\right)}{(\mu-\lambda)\left((1-\rho)\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{B^B\rho^B}{B!}\right)} \\
&= \frac{\frac{\rho^{B+1}B^{B-1}(1-B)}{B!}}{(\mu-\lambda)\left((1-\rho)\sum_{k=0}^{B-1}\frac{(B\rho)^k}{k!} + \frac{B^B\rho^B}{B!}\right)}.
\end{aligned} \tag{10}$$

All the factors are positive except $(1-B)$. Thus, $t_s^h < t_s^f$. ■

A. Illustration With Simple Cases

We illustrate and compare the average turnaround time of the flat and hierarchical deployment with two simple cases. When $B = 2$, we have

$$C(2, 2\lambda/\mu) = \frac{1}{1 + (1-\rho)\frac{2!}{(2\rho)^2}\sum_{k=0}^{1}\frac{(2\rho)^k}{k!}} = \frac{2\rho^2}{1+\rho}. \tag{11}$$

Thus, $t_s^h = \frac{C(2,2\lambda/\mu)}{2\mu-2\lambda} + \frac{1}{\mu} = \frac{\frac{\lambda^2/\mu^2}{1+\lambda/\mu}}{\mu-\lambda} + \frac{1}{\mu} = \frac{\mu}{\mu^2-\lambda^2}$, and $t^h = d_2 + t_s^h$.

Corollary 1. *When $B = 2$, the average turnaround time of the hierarchical deployment, t^h, is less than that of the flat deployment, t^f, if $\mu < \sqrt{\frac{\lambda}{d_2-d_1} + \lambda^2}$; $t^h = t^f$ if $\mu = \sqrt{\frac{\lambda}{d_2-d_1} + \lambda^2}$; $t^h > t^f$ if $\mu > \sqrt{\frac{\lambda}{d_2-d_1} + \lambda^2}$.*

Proof: Calculate the difference between t^f and t^h:

$$\begin{aligned}
t^h - t^f &= d_2 + \frac{\mu}{\mu^2-\lambda^2} - (d_1 + \frac{1}{\mu-\lambda}) \\
&= d_2 - d_1 - \frac{\lambda}{\mu^2-\lambda^2}
\end{aligned} \tag{12}$$

We can see that $t^h < t^f$ if and only if $d_2 - d_1 - \frac{\lambda}{\mu^2-\lambda^2} < 0$, which is equivalent to $\mu < \sqrt{\frac{\lambda}{d_2-d_1} + \lambda^2}$. ■

V. Simulations

We use the CloudSim Plus [20], [25], [26] platform to conduct various simulations. CloudSim Plus, an independent fork of CloudSim [21], is a discrete-event simulation tool designed for modeling and simulating cloud computing infrastructures and services. The entire simulation platform is written in the Java programming language. It defines many classes/interfaces for entities in the cloud computing environments; these interfaces/classes include *Datacenter*, *Host*, *Vm*, and *Cloudlet* (the class that corresponds to an application request/task in CloudSim Plus). It also defines the interfaces/classes for common concepts/techniques used in cloud computing, such as *CloudInformationServices* (serving as a registry of all resources in a cloud), *DatacenterBroker* (acting as a cloud customer that accepts application tasks/requests and submits them to the cloud), *VmAllocationPolicy* (used by the data center to allocate hosts for VMs), *VmSchedulingPolicy* (defining how a host schedules the VMs assigned to it), *CloudletScheduler* (defining how a VM schedules the cloudlets, i.e., application tasks/requests, assigned to the VM), etc. For most Java interfaces, the simulation tool provides some basic implemtations, and makes implementations of customized classes quite convenient. The simulation tool can also be easily modified to provide high-fidelity simulations for edge computing systems.

A. Verification for Balanced Server Allocation

In this subsection, we use a simple scenario to verify that when the request arrival rates across different user areas are the same, the server allocation to edge clouds should be balanced to achieve the minimum average turnaround time. Our simulation scenario consists of two user areas, and thus, two BSs. We have a total of 10 hosts to be allocated to the two BSs. Each host has one processing element, and runs a VM that uses all the resources on the host. The execution speed of the processing element is 1000 Million Instructions Per Second (MIPS). In each area, there is a same set of 60 application requests randomly arriving between 0 second to 600 seconds. The average length, i.e., of all the application requests is 50,000 Million Instructions (MI), while the length of each application request can range from 30,000 MI to 70,000 MI with uniform distribution. Thus, the average execution time of the application requests is 50 seconds, and the execution time for each application request can range from 30 seconds to 70

TABLE I. AVERAGE TURNAROUND TIME FOR DIFFERENT SERVER ALLOCATIONS

VM Mapper	Area	Subscenario No.				
		1	2	3	4	5
Round-Robin	Area 1	1267.92	510.6	263.95	146.62	85.62
	Area 2	54.27	55.74	59.30	65.58	85.62
	Overall	661.11	283.16	161.63	106.10	85.62
Best VM	Area 1	1267.95	509.87	263.35	144.1	82.67
	Area 2	53.77	54.67	53.07	61.58	82.67
	Overall	660.86	282.27	158.21	102.84	82.67

seconds. We choose these settings so that when there are 5 hosts at either BS, the load of the system at each area is close to 100% ($\frac{50 \text{ (seconds)} * 60 \text{ (tasks)}}{600 \text{ (seconds)} * 5 \text{ (VMs)}} = 1$).

We consider 5 subscenarios and caculate the average turnaround time for all the application requests in both user areas. In the ith subscenario, there is/are i host(s) at the first BS, and there are $10 - i$ hosts at the second BS. For example, in the first subscenario, there is 1 host at the first BS, and there are 9 hosts at the second BS. In the second subscenario, there are 2 hosts at the first BS, and there are 8 hosts at the second BS. In the 5th subscenario, there are 5 hosts at the first BS, and there are 5 hosts at the second BS.

The edge clouds at the two BSs will use the same strategy to map the application requests to the hosts/VMs in the local edge cloud. CloudSim Plus provides a simple datacenter broker (*DatacenterBrokerSimple*) that maps application requests to the VMs in a Round-Robin fashion. We also define our custormerized datacenter broker *DatacenterBrokerSimpleBestVm* so that when we consider mapping an application request to a VM in the local edge cloud, we choose the VM that can finish the application request the earliest. This is done by overriding the *defaultVmMapper()* method in the *DatacenterBroker* interface. We denote our new VM Mapper method as "Best VM," although it is just an intuitive heuristic.

The simulation results for the 5 subscenarios with different VM Mappers are shown in Table II. We can see that, the minimal turnaround time is achieved when the allocation of hosts to the BSs is balanced, i.e., when the numbers of hosts at two BSs are equal.

B. Verifications for Potential Advantages of Hierarchical Deployment Over Flat Deployment

We design simulation scenarios so that we can compare the turnaround time of a flat deployment and that of a hierarchical deployment. We still consider a scenario with 2 areas, i.e., 2 BSs. We have a total of 10 hosts. Each host has one processing element, and runs a VM that uses all the resources on the host. We randomly generate a set of *N_REQUESTS* application requests for each user area. Each application request has the same characteristics as that in our previous simulation. That is, each application request randomly arrive between 0 seconds to 600 seconds, with an average execution time of around 50 seconds.

In the flat deployment scenario, we allocate the servers to the base stations in the best way. That is, the two BS will have the same number of servers, i.e., the flat deployment scenario has 5 hosts at each BS. The first set of *N_REQUESTS* application requests are processed at the 5 hosts at the first BS. The second set of *N_REQUESTS* application requests are processed at the 5 hosts at the second BS. As we have shown through theoretical analysis, the hierarchical deployment can have potential advantages over the flat deployment because it enables resource sharing for all application requests. In our simulation, we choose a extreme design to place all the 10 hosts at the central office as the hierarchical deployment scenario, while there are no servers co-located with the BSs. In this design, all of the 10 hosts can be shared by all the application requests. The application requests can be flexibly assigned to any server/VM for execution. We assume that the communication delay bewteen the BS and the central office is negligible compared to the application requests' execution time.

We also consider two strategies for mapping application requests to the servers/VMs: the Round-Robin strategy used by *DatacenterBrokerSimple*, and our customized strategy (Best VM) that maps the application request to the VM that can finish the request the earliest. We vary the number of application requests, *N_REQUESTS*, from 30 to 120 with a stepsize of 10, and design 10 groups of comparisions. The simulation results with different VM mappers are shown in Table II.

Using either Round-Robin or Best VM as the VM mapper, the row labeled with "Area 1" includes the average turnaround time for application requests at area 1 in the flat deployment; the row labeld with "Area 2" include the average turnaround time for application requests at area 2 in the flat deployment; the row labeled with "Average of Flat" includes the average turnaround time for all the application requests in the flat deployment. The row labeled with "Average of Hierarchical" includes the average turnaround time for all application requests in the hierarchical deployment. We can see from the table that if we use the "Round-Robin" as the VM mapper, for majority of the cases, the average turnaround time in the hierarchical deployment is less than that in the flat deployment; however, there are some exceptions. For example, when the number of application requests is 110, the average turnaround time of the hierarchical deployment (308.35 seconds) is greater than that of the flat deployment (307.88). This result is not a counter example however, because Round-Robin is really not a method that can efficiently share the available resources; it just blindly assign the requests to the VMs without taking into consideration the current assignment/execution of the VM. In contrast, Best VM is a much better method that tries to achieve efficient resource sharing. From the table, we can see that if we use the "Best VM" as the VM mapper, the average turnaround time in the hierarchical deployment is always less than that in the flat deployment. This argument is also backed by all other simulations that have been conducted but whose results are not included in the paper due to page limit. The simulation results also tell us that although the hierarchical deployment has great potential to achieve better performances, we need to use efficient scheduling/mapping methods to achieve the good performances.

VI. CONCLUSION AND FUTURE WORK

In this paper, we consider the server allocation problem with two general edge cloud deployment approaches: flat and hierarchical deployment. For the flat deployment strategy, we demonstrate that when the request arrival rates are the same across all the edge clouds, the allocation of servers to the

TABLE II. Average Turnaround Time for Different Numbers of Application Requests

VM Mapper	Scenario	Number of Application Requests in Each Area									
		30	40	50	60	70	80	90	100	110	120
Round-Robin	Area 1	49.23	49.40	58.14	83.78	142.46	173.18	179.67	209.32	321.72	373.38
	Area 2	50.56	58.45	62.16	66.62	130.37	166.98	191.87	252.60	294.03	352.70
	Average of Flat	49.90	53.93	60.14	75.20	136.42	170.08	185.77	230.96	307.88	362.85
	Average of Hierarchical	48.95	51.78	54.77	69.69	133.71	167.48	175.71	227.52	308.35	360.33
Best VM	Area 1	48.90	48.60	55.88	78.08	141.67	168.16	178.98	208.58	321.05	372.29
	Area 2	50.56	56.85	59.78	64.30	129.30	163.08	190.22	252.10	293.12	351.13
	Average of Flat	49.73	52.67	57.83	71.19	135.49	165.62	184.60	230.34	307.09	361.71
	Average of Hierarchical	48.78	50.91	52.25	64.72	132.40	160.94	171.44	226.33	307.02	359.16

edge clouds should be balanced to achieve the minimum average turnaround time. We demonstrate that the hierarchical deployment approach has great potentials in terms of reducing the average turnaround time, compared to the flat deployment; besides, the strategy we use to schedule/map application requests to VMs may play an important role.

Future work can be conducted in, but not limited to, the following directions. First, general and practical hierarchical architectures with more than two hierarchical levels should be considered. Second, more practical application request models and VM models should be considered; for example, each application request may require multiple processing elements, and we may have different types of VMs to choose from at real time. Third, the impact of various scheduling/allocation policies should be investigated, and explorations of optimal system deployment considering these scheduling/allocation policies are needed.

References

[1] X. Sun and N. Ansari, "Mobile edge computing empowers internet of things," *ArXiv e-prints*, vol. arXiv:1709.00462, September 2017.

[2] Z. Pang, L. Sun, Z. Wang, E. Tian, and S. Yang, "A survey of cloudlet based mobile computing," in *Proceedings of International Conference on Cloud Computing and Big Data (CCBD)*, November 2015, pp. 268–275.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Third quarter 2017.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC Workshop on Mobile Cloud Computing*, August 2012, pp. 13–16.

[5] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the Workshop on Mobile Big Data (Mobidata)*, June 2015, pp. 37–42.

[6] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, October 2015.

[7] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, and Y. Zhang, "Software defined cooperative offloading for mobile cloudlets," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1746–1760, June 2017.

[8] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, October - December 2009.

[9] C. Zhou and C. K. Tham, "Deadline-aware peer-to-peer task offloading in stochastic mobile cloud computing systems," in *Proceedings of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2018, pp. 1–9.

[10] S. Bohez, T. Verbelen, P. Simoens, and B. Dhoedt, "Allocation algorithms for autonomous management of collaborative cloudlets," in *Proceeding of the 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, April 2014, pp. 1–9.

[11] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 27, no. 10, pp. 2866–2880, October 2016.

[12] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, April 2016, pp. 1–9.

[13] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, October 2017.

[14] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," in *Proceedings of the 40th IEEE Conference on Local Computer Networks (LCN)*, October 2015, pp. 570–578.

[15] X. Sun, N. Ansari, and Q. Fan, "Green energy aware avatar migration strategy in green cloudlet networks," *ArXiv e-prints*, vol. arXiv:1509.03603, September 2015.

[16] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.

[17] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, April 2016, pp. 1–9.

[18] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet network design optimization," in *Proceedings of IFIP Networking Conference*, May 2015, pp. 1–9.

[19] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData and SocialInformatics (BD&SI)*, October 2015, pp. 28:1–28:6.

[20] M. C. S. Filho, R. L. Oliveira, C. C. Monteiro, P. R. M. Inácio, and M. M. Freire, "Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 400–406.

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, January 2011.

[22] D. Li, B. Dong, E. Wang, and M. Zhu, "A study on flat and hierarchical system deployment for edge computing," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0163–0169.

[23] E. Wang, D. Li, B. Dong, H. Zhou, and M. Zhu, "Flat and hierarchical system deployment for edge computing systems," *Future Generation Computer Systems*, vol. 105, pp. 308 – 317, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X19304042

[24] L. Kleinrock, *Queueing Systems. Volume 1: Theory*. Wiley-Interscience, January 1975.

[25] "A modern, full-featured, highly extensible and easier-to-use java 8+ framework for modeling and simulation of cloud computing infrastructures and services," http://cloudsimplus.org/, accessed: 2020-03-22.

[26] "A modern, full-featured, highly extensible and easier-to-use java 8+ framework for cloud computing simulation," https://github.com/manoelcampos/cloudsim-plus, accessed: 2020-03-22.