Spring 2018

Joint Computation Offloading and Prioritized Scheduling in Mobile Edge Computing

Lingfang Gao
San Jose State University

Writing Project

Joint Computation Offloading and Prioritized Scheduling in Mobile Edge Computing

Final Report

Author

Lingfang Gao

CS 298

May 2018

Advisor

Dr. Melody Moh

A Writing Project Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment of the Requirements for the Degree: Master of Science

The Designated Committee Approves the Master's Project Titled

JOINT COMPUTATION OFFLOADING AND PRIORITIZED SCHEDULING IN MOBILE EDGE COMPUTING

By

LINGFANG GAO

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

MAY 2018

Dr. Melody Moh Signature:
Department of Computer Science

Dr. Robert Chun Signature:
Department of Computer Science

Dr. Teng Moh Signature:
Department of Computer Science

ABSTRACT

Joint Computation Offloading and Prioritized Scheduling in Mobile Edge Computing

by Lingfang Gao

With the rapid development of smart phones, enormous amounts of data are generated and usually require intensive and real-time computation. Nevertheless, quality of service (QoS) is hardly to be met due to the tension between resource-limited (battery, CPU power) devices and computation-intensive applications. Mobile-edge computing (MEC) emerging as a promising technique can be used to copy with stringent requirements from mobile applications. By offloading computationally intensive workloads to edge server and applying efficient task scheduling, energy cost of mobiles could be significantly reduced and therefore greatly improve QoS, e.g., latency. This paper proposes a joint computation offloading and prioritized task scheduling scheme in a multi-user mobile-edge computing system. We investigate an energy minimizing task offloading strategy in mobile devices and develop an effective priority-based task scheduling algorithm with edge server. The execution time, energy consumption, execution cost, and bonus score against both the task data sizes and latency requirement is adopted as the performance metric. Performance evaluation results show that, the proposed algorithm significantly reduce task completion time, edge server VM usage cost, and improve QoS in terms of bonus score. Moreover, dynamic prioritized task scheduling is also discussed herein, results show dynamic thresholds setting realizes the optimal task scheduling. We believe that this work is significant to the emerging mobile-edge computing paradigm, and can be applied to other Internet of Things (IoT)-Edge applications.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my advisor Dr. Melody Moh for the continuous support of my master study, for her patience, immense knowledge. Her guidance helped me in all the time of research and writing project.

In addition, I would also like to extend my gratitude to my committee members, Dr. Teng Moh and Dr. Robert Chun for their valuable suggestions, support and time. I would also like to thank my husband Zhenbo Xing, my son Jason Xing and my parents for constant love and support throughout my master study.

TABLE OF CONTENTS

List of Acronyms

AHP – Analytical Hierarchy Process

EDF – Earliest Deadline First

IoT – Internet of Things

KKT conditions – Karush – Kuhn – Tucker conditions

MCC – Mobile Cloud Computing

MEC – Mobile Edge Computing

QoE – Quality of Experience

QoS – Quality of Service

SLA - Service Level Agreements

List of Figures

List of Tables

1. INTRODUCTION

Mobile applications are abundant in nowadays, more and more mobile applications are seeking for fast and customized service. These applications are more likely to be resource-demanding applications, such as video chat, online gaming, requires real-time communication and intensive computation. However, due to resource limitation (battery lifetime, storage capacity, CPU power) of mobile devices, users are not satisfying the service compared to desktop [1]. Moreover, intensive computation and real-time transmission also implies heavy CPU processing and wireless transmission, causing significant energy cost of mobile devices [1]. Issues with battery consumption of mobiles, response time, freshness, accuracy, and quick delivery are potentially affected. Many researchers have made great efforts on delivering high quality service to users and saving energy for mobile devices. One popular solution for mobile devices is computation offloading: applications take advantage of resource-rich infrastructures by deploying computation to these infrastructures [2]. Furthermore, researchers have recognized offloading computation to cloud can significant reduce power consumption of mobile devices [3, 4]. While Offloading application to a remote cloud works well for non-time critical applications, such as pictures, videos, and documents, it is not ideal when supporting a real-time mobile solution [5]. Latency and network availability impact cloud based computation offloading.

Mobile edge computing (MEC) is a promising solution to cope with the above challenge. MEC provide cloud-like service within the mobile edge network [6]. Instead of pushing up data to remote clouds, edge computing aims to process part of the mobile's workload on edge nodes, which serves as computing agent closer to users between mobile devices and cloud servers. MEC has several advantages compared to traditional mobile cloud computing (MCC), such as short latency and low energy

consumption [7]. MEC is a feasible solution to satisfy the ever-increasing comprehensive requests demanded by users.

Since part of workload from mobile devices are deployed to edge servers in MEC, efficient task scheduling schemes also needed to be considered. Efficient task scheduling policy would gain high system throughput to improve Service Level Agreements (SLA) [8]. Priority of tasks is of great importance in scheduling because some jobs with stringent latency requirement should be served earlier than other jobs in the system. An appropriate task scheduling algorithm must consider priority of tasks especially in a relatively resource limited edge server.

In this paper, we address issues of computation offloading and task scheduling in mobile-edge computing. A joint solution combining optimal computation offloading and prioritized task scheduling model is proposed for a multi-user MEC system. Briefly, in mobile layer, an optimal computation offloading model with energy consumption constraints is used to decide the offloading fractions of mobile applications. In particular, whether to and how much to offload computation tasks to edge server is determined by mobile energy condition and latency requirement. In the edge layer, tasks coming from mobiles devices are queued and served by a prioritized-based task scheduling policy. Service sequence is determined by subscription priority requirement and latency deadline.

The rest of the paper is organized as follows. In next section, we review related works on computation offloading and task scheduling, especially on those reduce power consumption for mobile devices. Section 3 and 4 presents the architecture of the MEC system and the major algorithms employed in MEC. Evaluation and analysis of algorithms are conducted in Section 5 and Section 6. We conclude the paper in Section 7.

2. RELATED WORK

Computation offloading in mobile devices and task scheduling in edge server are two main challenges in MEC system. However, MEC is a new introduced paradigm; therefore, edge oriented resource management is not yet addressed that much. In this section, we briefly review few computational offloading policies and task scheduling approaches for energy conservation and meeting time constraint in MEC architecture.

2.1 Computation Offloading

Many researches are focusing on computation offloading in MEC for energy saving and performance enhancement. Huang, Wang, and Niyato [9] proposed an adaptive offloading algorithm. With dynamic data rate adjusting techniques, mobile execution energy consumption with time constraint was minimized. Xie and Dan [10] studied a dynamic size-controlled algorithm for computation offloading in a collaborative MCC system. A joint allocation of tasks and resources for MEC system was proposed by Sardellitti, Scutari, and Barbarossa [11], a tradeoff between energy consumption and tardiness was discussed. Yousefpour et al., [12] proposed an QoS-aware based offloading method to discuss tradeoff between energy and latency. More recently, the optimization of energy-delay of MEC system with varied applications have been carried out by Lyapunov optimization algorithm, which investigating offloading scheme, task allocation, CPU cycle requirement and network [13]. Furthermore, tradeoff between mobile power and processing delay for multi-user MEC systems was investigated via implementing a dynamic network and computational resource allocation [14]. Although energy conservation is attractive for MEC system, performance guaranteed is important for real-time mobile applications. However, there has been very little research report on the performances guaranteed, e.g., under the constraints of computing capability, transmission bandwidth, and task latency requirement while minimizing energy consumption.

2.2 Priority Task Scheduling in Edge Computing

QoS requirements are especially critical for mobile applications, such as priority of user's request, speed of delivery and service cost. Prioritized task scheduling in edge computing plays an important role in edge computing, as it significantly reduces service time and improves SLA. A priority based service scheduling algorithm was proposed by Dakshayini and Guruprasad [15]. The model gained high throughput of the cloud and significantly reduced service time by making an efficient provision of cloud resources. We adopted this priority based algorithm for task scheduling in edge layer, as described in Section 3.3. Ignole and Chana [16] introduced a multilevel priority-based task scheduling in cloud computing environment. The proposed scheduling policy prioritized tasks based on dynamic threshold values, and considerably reduced makespan. Besides, Ghanbari and Othman [17] recently reported a priority based job scheduling algorithm. The proposed algorithm is according to multiple criteria decision-making model based on the theory of Analytical Hierarchy Process (AHP). Choudhari and Moh [18] applied a proposed prioritized task scheduling in the fog layer of a client-fog-cloud computing system, their study reveals that the proposed algorithm significantly reduced the response time and the cost of the system.

3. MOBILE-EDGE SYSTEM ARCHITECTURE

The general structure of mobile-edge computing system can be represented in Fig. 1. We consider multiple mobile devices in one mobile-edge computing system. The edge server is regarded as a mini data center installed at a wireless access station. Each mobile user is subscribed to this closer edge server. Tasks from mobile users are incoming through wireless channel. A similar MEC architecture was reported by Tao and Ota [19]. Task offloading can help mobile users to improve computation performance and reduce energy consumption of mobile devices.

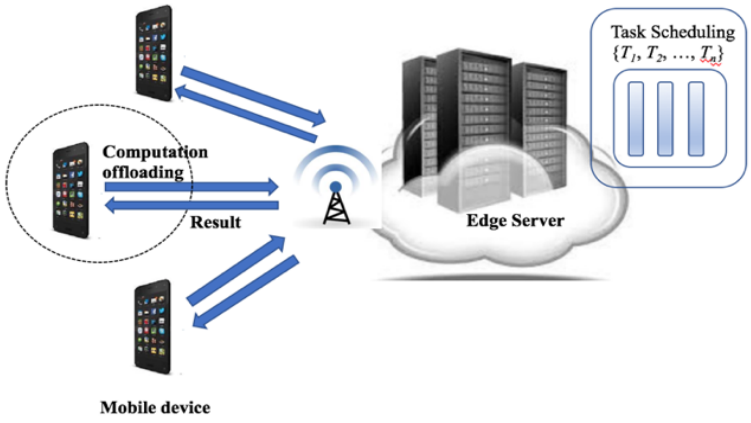

Fig. 1 A multi-user mobile edge computing system

This model consists of two layers: mobile client and edge server layer. In this general structure, there is much room for various task allocation schemes, specifically where to handle and how to handle tasks. This is where the various allocation methods and scheduling strategies come into play. Here, we design three task allocation models in our MEC architecture, including (i) all local process model, (ii) all offload process model, and (iii) partial offload process model. The succeeding sections present and discuss different allocations and scheduling ways of managing tasks in the context of MEC.

3.1. Task Model

In this study, we consider an independent task $T = \{t_1, t_2, \ldots, t_j\}$ for each mobile user. A task ***t*** *submitted* by a mobile user n can be modeled by a collection of parameters, i.e., $t_n = \{c_n, d_n, T_n, P_n\}$, where c_n, d_n, T_n, and P_n denotes required CPU cycle per bit of tasks, task data size, deadline requirement, and subscribed priority value of task t_n , respectively. We let l_n denotes the offloading data size of mobile n, α_n denotes the fraction of task offloading for each mobile user n, where $l_n = d_n \alpha_n$. Further, we define a desired power consumption $E_{b,n}$ for each mobile device, from which we can calculate the energy requirement baseline for each mobile.

3.2. Optimization Offloading Decision Model

All local and all offloading model are pretty straightforward. However, in partial offload process model, how to decide the optimal offloading fraction of tasks for each mobile user is the main issues to be addressed herein. In this section, we formulate offloading decision problem as an energy efficiency optimizing problem under latency requirement constraint for MEC system, which is adopted from Tao's research [19].

We first introduce the all local process model. Tasks are computed locally, no task transmission needed. Hence, energy consumption of all local process just relates to task data size and CPU requirement. Here we define E_l as the all local computation energy consumption as shown in (1), $t_{n,l}$ as the all local computation completion time shown in (2),

$$E_l = f_n c_n \tag{1}$$

$$t_{n,l} = \frac{c_n}{h_n} \tag{2}$$

where f_n denotes power consumption per CPU cycle for mobile n, h_n denotes computing capability of mobile n.

In all offload process model, there is no mobile energy consumption for execution in mobile devices, only energy consumption for transmission considered in the system shown in Eq. (3). Completion time in offload process model contains transmission time and execution time shown in (4). Energy consumption and completion time of all offload process model can be defined as E_o and $t_{n,o}$,

$$E_o = \frac{d_n p_n}{r_n} = p_n t_n \tag{3}$$

$$t_{n,o} = \frac{d_n}{r_n} + \frac{c_n}{h_n^e} \tag{4}$$

where p_n denote the transmission power for mobile n, r_n is the transmission rate of mobile n, which can be defined as (5),

$$r_n = B\ log_2\ (1 + \frac{p_n g_n^2}{N_o\ B}) \tag{5}$$

Let *B* denote the bandwidth of the wireless channel, g_n is the channel gain of the edge server.

Based on the above model, we can formulate energy consumption of partial offload process model. We define α_n as task offloading fraction to edge server. Hence, the energy consumption of each mobile *n* contains local execution consumption and partial offloading tasks transmission consumption as shown in (6),

$$E_n = E_o\alpha_n + E_l\ (1 - \alpha_n) = p_n t_n \alpha_n + f_n c_n\ (1 - \alpha_n) \tag{6}$$

The completion time of each mobile *n* also includes local execution time and partial offloading transmission time, which can be calculated as (7),

$$t_n = t_{n,l}(1 - \alpha_n) + t_{n,o}\alpha_n \tag{7}$$

The goal of the optimization problem is to minimize energy consumption of mobiles. Therefore, our model aims to calculate the optimal task offloading fraction for each mobile n under the constraints of edge server computing capability, transmission bandwidth, and task latency requirement as well. The problem is a convex optimization problem [20]. We use Lagrange method to derive a task allocation scheme. Energy efficient offloading optimizing problem can be formulated as:

$$\min_{\{\partial_n, t_n\}} \sum_{i=1}^{n} [\frac{d_n p_n}{r_n}\alpha_n + f_n c_n\ (1 - \alpha_n)] \tag{8}$$

$$s.t. \quad \frac{c_n}{h_n}(1 - \alpha_n) + \left(\frac{d_n}{r_n} + \frac{c_n}{h_n^e}\right)\alpha_n - T_n \leq 0 \quad \forall n, \tag{9}$$

$$\frac{d_n p_n}{r_n}\alpha_n + f_n c_n(1 - \alpha_n) - E_{b,n} \leq 0 \quad \forall n \tag{10}$$

$$\sum_{i=1}^{n} c_n \alpha_n \leq \mathsf{C} \tag{11}$$

$$\sum_{i=1}^{n} r_n \leq B \tag{12}$$

where C denotes edge server CPU computing capability.

The optimal problem (8) is a convex optimization problem. We define an increasing and convex function $h(x) = N_0 B(2^{\frac{x}{B}} - 1)$ (while $x > 0$), adopted from

Tao's research [19]. Therefore, the transmission power p_n can be calculated as (13), and Eq. (6) can be rewritten as Eq. (14)

$$p_n = \frac{1}{g_n^2} h\left(\frac{l_n}{t_n}\right) \tag{13}$$

$$E_n = E_o \alpha_n + E_l\,(1 - \alpha_n) = p_n t_n \alpha_n + f_n c_n\,(1 - \alpha_n)$$

$$= \frac{t_n}{g_n^2} h\left(\frac{l_n}{t_n}\right) \alpha_n + f_n c_n\,(1 - \alpha_n) \tag{14}$$

Since function $h(x)$ is convex, and its multiplier function $\frac{t_n}{g_n^2} h\left(\frac{l_n}{t_n}\right)$ is also convex. Therefore, the sum of convex equations, remains convex. To solve this convex problem, we define a partial Lagrangian function $\mathcal{L}(\alpha, t, \lambda, \mu)$ shown in (15),

$$\mathcal{L}(\partial, t, \lambda, \mu) = \frac{t_n}{g_n^2} h\left(\frac{l_n}{t_n}\right) \alpha_n + f_n c_n\,(1 - \alpha_n) + \lambda\left[\frac{c_n}{h_n}(1 - \alpha_n) + \left(\frac{d_n}{r_n} + \frac{c_n}{h_n^e}\right)\alpha_n - T_n\right]$$
$$+ \mu\left[\frac{t_n}{g_n^2} h\left(\frac{l_n}{t_n}\right) \alpha_n + f_n c_n\,(1 - \alpha_n) - E_{b,n}\right] \tag{15}$$

where $\lambda \geq 0\ and\ \mu \geq 0$ are the dual Lagrange multiplier linked to constraints of completion time and energy consumption. Let α_n^* denotes optimal solution which always exist. Then we apply KKT condition and transform Eq. (15) to following equations:

$$\frac{\partial_{\mathcal{L}}}{\partial^*_{\alpha_n}} = (1 + \mu)\frac{t_n}{g_n^2} h\left(\frac{l_n}{t_n}\right) - (1 + \mu) f_n c_n + \lambda\left[\frac{c_n}{h_n^e} - \frac{c_n}{h_n}\right] \tag{16}$$

$$\frac{\partial_{\mathcal{L}}}{\partial^*_{t_n}} = \left[\frac{\alpha_n^*}{g_n^2} + \mu\alpha_n^*\right]\left[h\left(\frac{l_n^*}{t_n^*}\right) - \frac{l_n^*}{t_n^*} h'\left(\frac{l_n^*}{t_n^*}\right)\right] + \lambda\alpha_n^* \tag{17}$$

$$\frac{\partial_{\mathcal{L}}}{\partial^*_{\lambda}} = \frac{c_n}{h_n}(1 - \alpha_n) + \left(\frac{d_n}{r_n} + \frac{c_n}{h_n^e}\right)\alpha_n - T_n \tag{18}$$

$$\frac{\partial_{\mathcal{L}}}{\partial^*_{\mu}} = \frac{t_n}{g_n^2} h\left(\frac{l_n}{t_n}\right) + f_n c_n\,(1 - \alpha_n) - E_{b,n} \tag{19}$$

$\alpha_n\ and\ t_n$ can be derived from above equation. Based on this, the optimization problem is computed based on the dual function $g(\lambda, \mu) = min\mathcal{L}(\alpha, t, \lambda, \mu)$ and Lambert function, adapted and adjusted from Tao's research [19]. Finally, we conduct the result of α_n^* as shown in (20),

$$\alpha_n^* = \frac{T_n h_n h_n^e - c_n h_n^e}{\frac{h_n^e h_n d_n}{r_n} + c_n h_n - c_n h_n^e} \quad (20)$$

From the above derive we can conclude that optimal computation offloading fraction is tightly related to task's latency requirement and task data size.

3.3. Prioritized Task Scheduling Model

Offloading tasks coming from mobile users may have various latency requirements that needed to be satisfied. Here, we enhance an existing priority based scheduling algorithm in edge server layer based on Dasshayini and Guruprasad's research [15]. Parameters used for prioritized scheduling model can be found following,

- Three priority queues Q_H, Q_M, and Q_L, corresponding to three subscription catalogues (SB CAT) of task: 3 = High, 2 = Medium, 1 = Low.
- Two thresholds T_1 and T_2 for latency requirement at levels.
- The maximum tolerable waiting time of each task *i* can be calculated as (21),

$$wait_i = (LR_i - C_i) \quad (21)$$

  where LR_i denote latency requirement of task i, C_i is current time

- $T_{est,i}$ is estimated service time of task *i*.

Task will be placed in one of three queues based on subscription catalogues and latency requirement.

In the above parameters, thresholds T_1 and T_2 are set and adjust based on experiments. Thresholds are used to reorder the tasks based on their latency requirements and subscribed priority levels. As T_1 and T_2 have signify different ($T_1 < T_2$), the maximum waiting time, $wait_i$ is used to check against estimated service time $T_{est,i}$ and T_1, T_2. Therefore, all the tasks have stringent latency requirement will be inserted into high priority queue. Tasks which have low latency requirement will be

added into medium or low priority queues, so that tasks have higher priority are processed first.

4. PROPOSED ALGORITHM

In this section, the proposed prioritized offloading in mobile-edge computing is described herein. The proposed algorithm called Joint Computation Offloading and Prioritized Scheduling Algorithm, which extends Tao's research [19] by implementing optimization offloading decision algorithm in mobile layer and an enhanced version of prioritized scheduling algorithm [15] in edge layer. The algorithm consists of two parts. Part A in mobile layer, each mobile randomly generates a task, and within the mobile device, offloading fraction is calculated by the optimal offloading decision model described in Section 3.2. Part B in edge layer, edge server process all the offloading tasks coming from mobile users and order them in a priority queue based on the latency requirement and subscription catalogues, or terms of priority levels presented in Section 3.3. Below, the high-level description of the algorithm is presented.

A. Energy-Efficient Task Offloading Algorithm in Mobile Layer

1: ***for*** *each mobile user n*
2: ***for*** *each task* d_i *in mobile n*
3: *Calculate the optimal offloading fraction* α_n *by (20)*
4: *Offload* $d_i \times \alpha_n$ *tasks to edge server;*
5: *Execute* $d_i \times (1 - \alpha_n)$ *tasks at local mobile*
5: ***end for***
6: ***end for***

B. Prioritized Task Scheduling Algorithm in Edge Layer

1: ***for*** *each task in edge server queue*
2: *Task manager in edge server check maximum waiting time* $wait_i$ *by (21);*
3: ***if*** $wait_i \leq$ *estimated service time* $T_{est,i}$, ***then***
4: *Place* $task_i$ *in* Q_H;
5: ***else if*** $T_{est,i} + T_1 \leq wait_i \leq T_{est,i} + T_2$., ***then***
6: ***if*** SB_{CAT} *==1,* ***then***
7: *Place* $task_i$ *in* Q_H;

8: ***if*** SB_{CAT} *==2,* ***then***
9: *Place* $task_i$ *in* Q_M*;*
10: ***if*** SB_{CAT} *==3,* ***then***
11: *Place the* $task_i$ *in* Q_L*;*
12: ***else if*** $wait_i > T_{est,i} + T_2$*, then*
13: ***if*** SB_{CAT} *==1 and is* Q_H *not full,* ***then***
14: *Place* $T_{est,i}$ *in* Q_H*;*
15: ***else***
16: *Place the* $T_{est,i}$ *in* Q_M*;*
17: ***else if*** SB_{CAT} *==2 and is* Q_M *not full,* ***then***
18: *Place the* $T_{est,i}$ *in* Q_M*;*
19: ***else***
20: *Place the* $T_{est,i}$ *in* Q_L*;*
21: ***else if*** SB_{CAT} *==3,* ***then***
22: *Place the* $T_{est,i}$ *in* Q_L*;*
23: ***end if***
24: ***end for***

The above algorithm can be described as, mobile devices calculate the optimal computation offloading fractions for each batch of task in Step A. Within this step, fraction of α_n^* tasks are offloaded to edge server, fraction of $(1 - \alpha_n^*)$ tasks are executed in local mobile. For those tasks offloading to edge server, execute algorithm B.

Algorithm B processes all offloading requests from 2000 mobile devices, in this step, maximum waiting time $wait_i$ is calculated and compared with estimated execution time $T_{est,i}$. If $wait_i$ is less or equal to $T_{est,i}$, the task is placed in to highest priority queue Q_H regardless of subscription catalogues. Next, $wait_i$ is compared against $T_{est,i}$ plus threshold value T_1 and $T_{est,i}$ plus threshold value T_2, if $wait_i$ greater is between them, place task into the queue based on its subscription catalogues. Last, check $wait_i$ with $T_{est,i}$ plus threshold value T_2. Place task in its original subscription catalogue if the corresponding queue has space, otherwise downgrade one queue level.

We try to guarantee that high priority tasks will be executed first by applying this algorithm.

5. SIMULATION SETTING

To evaluate the different task allocation designs and algorithms in a mobile-edge computing scenario, we use CloudSim as simulator [21]. Algorithms shown in Table 1were implemented and compared the simulation results through CloudSim. In order to evaluate how the prioritized scheduling algorithm performed in this mobile-edge computing system, it is reasonable to compare it to a known heuristic algorithm scheme. Classic algorithm for task scheduling in edge server used for comparison with our proposed algorithm is Earliest Deadline First (EDF). EDF involves ordering tasks based on the deadline requirement, the task with highest deadline requirement will be executed firstly. The settings for MEC system parameters in simulation are summarized in Table 2.

Table 1 Algorithms

Acronym	Algorithm in mobile devices	Algorithm in edge server
All local	FCFS	*N.A.*
All offload	*N.A*	EDF
All offload + Priority	FCFS	Prioritized Scheduling
Tao's Partial Offloading	Optimal Offloading Decision	EDF
Tao's Partial Offloading + Priority	Optimal Offloading Decision	Prioritized Scheduling

Table 2 Simulation Parameters and Values

Parameter	Value
#. Mobile Devices (same as #. tasks)	2000
#. Edge Host	1
#. VMs on Edge Host	5
Mobile Devices Computing Capability	Randomly from 0.5 ~ 1.0 GHz
Edge VMs Computing Capability	10 GHz
Task Data Size	20 MB ~ 200 MB
Network Bandwidth	5 MHz
Density of Noise Power of Network Channel	10^{-12} W
Latency Requirement	200 ms ~ 2000 ms
Latency Requirement for Bonus Score Exp.	1400 ms ~ 2600 ms

Task Priority	Randomly set to be 1 or 2 or 3
Scheduling Threshold T_1	300 ms
Scheduling Threshold T_2	800 ms

Various metrics are considered herein, including task completion time, mobile energy consumption, cost of edge serve VM usage, and Bonus score.

6. Performance Evaluation

In this section, we discuss the performance of varied task allocation strategies, including all local mobile computing, all offload to edge server with and without prioritized scheduling computing, Tao's Partial Offloading to edge server with optimization computation offloading, without prioritized scheduling computing method. Results of task completion time, mobile energy consumption, cost of edge server VM usage, and bonus score are presented below.

6.1. Task Completion Time

We discuss the completion time vs. task size for different task allocation strategies as shown in Fig. 2. The completion time of partial offloading tasks includes local mobile execution time, transmission time in wireless network as well as execution time in edge server.

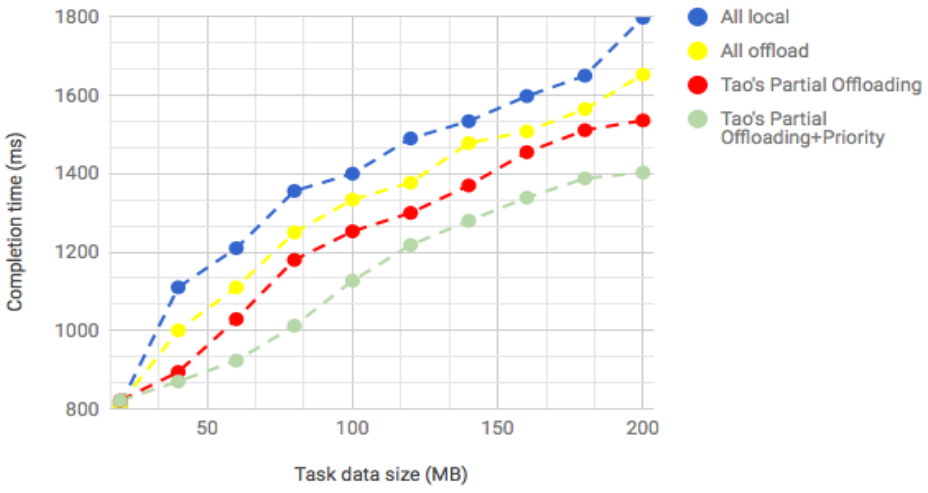

Fig. 2 Completion time vs. task data sizes

From the result of Fig. 2, it can be observed that, firstly, when task sizes are small, different task offloading methods has similar completion times. However, the completion time has significant different trend when task sizes becomes larger. Secondly, partial loading with or without prioritized scheduling has less completion time compared to all local execution and all offload to edge server computation. Lastly, our proposed joint computation offloading and prioritized scheduling method (Tao's Partial Offloading + Priority) performs better in terms of completion time compared to Tao's partial offloading without prioritized scheduling method. The improvement is because, implementing priority scheduling in edge server, tasks are placed in priority orders according to their tolerance delay. Therefore, reduce the total completion time compared to partial offloading without prioritized scheduling method. We can conclude that Tao's Partial Offloading with prioritized scheduling in edge server reduce the total completion time of tasks due to efficient task scheduling strategy based on the task deadline and their priority.

Fig. 3 reveals the completion time variance with latency deadline with task size of 140 MB. Fig. 3 shows that (i) our proposed prioritized partial offloading strategy has lower completion time compared to other offloading methods.

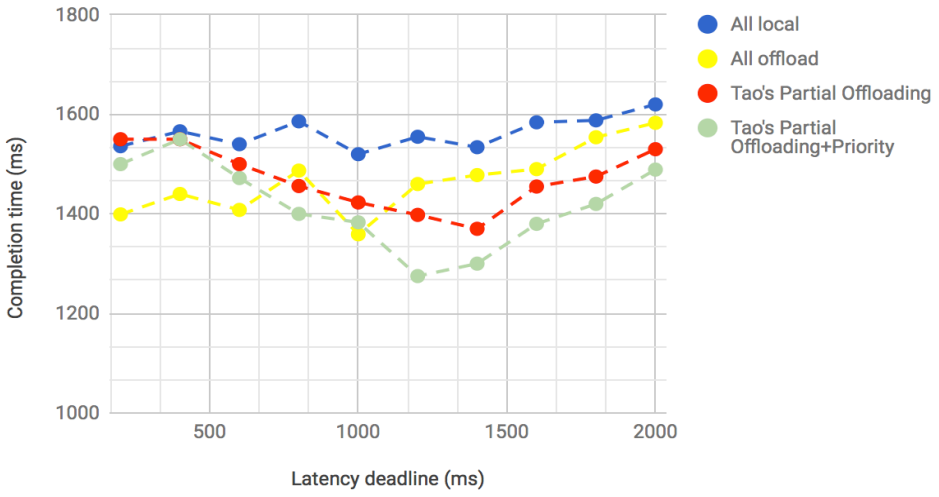

Fig. 3 Completion time vs. latency requirements

This is because latency requirement affects offloading percentage between mobile devices and edge server. When latency requirements are low, most of the tasks are offloaded to edge server, therefore, the completion time of tasks reduces due to the powerful computing capability of edge server. (ii) Our proposed offloading strategy performs better in completion time because of the efficient task execution order scheduling. (iii) Completion time increases for all offloading strategies as the latency requirement becomes low. (iv) And there is no significant completion time variance when latency requirement increases for all local mobile execution.

6.2. Mobile Energy Consumption

Mobile energy consumption is the main metric of evaluating task allocation strategy in mobile-edge architecture. Fig. 4 depicts the energy consumption difference with the task size and latency requirement.

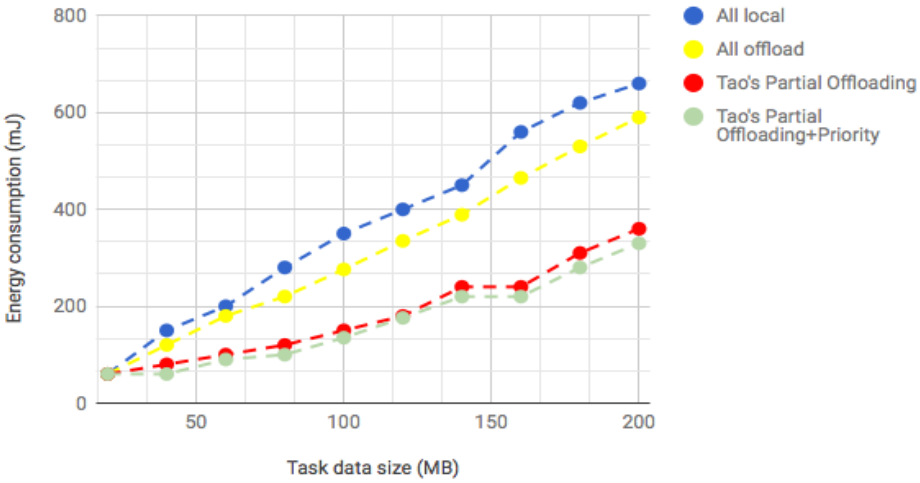

Fig. 4 Energy consumption vs. task data sizes

Results show task offloading to edge server reduce energy consumption of mobile devices significantly compared to all local execution as shown in Fig. 4. Specifically, Tao's Partial Offloading consumes less mobile battery energy than all offload and the impact is greater when the task data sizes become larger. This indicates that Tao's Partial Offloading deploys tasks to edge server make a tradeoff between all

local computing energy consumption and data transmission via wireless network energy consumption for all offloading.

We also discuss the energy consumption variance with latency requirements change shown in Fig. 5. Similar to completion time vs. latency requirement, when task latency requirements are low, large portions of tasks are offloaded to edge server, hence, reduce the energy consumption. Energy consumption of all local execution keeps constant as the latency requirements changes as energy consumption is just affected by task size and CPU requirement. And no significant mobile energy consumption observed between Tao's Partial Offloading with or without prioritized scheduling, because task scheduling in edge server has no impact on mobile energy consumption.

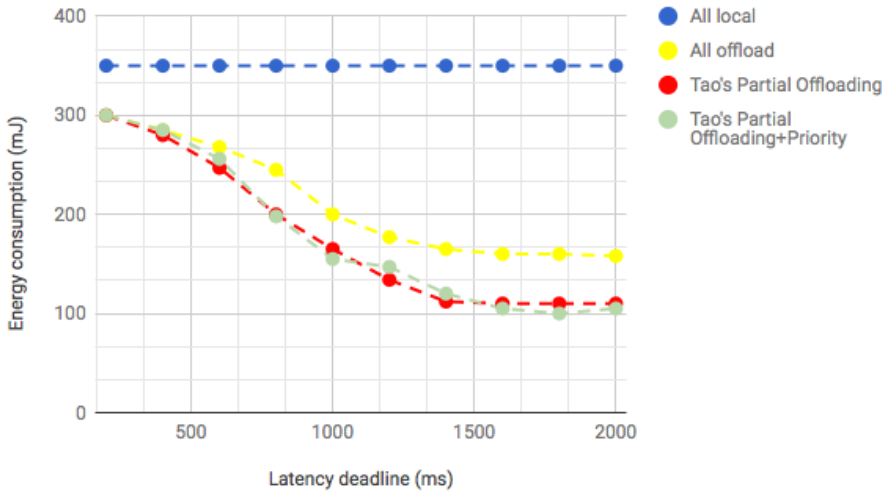

Fig. 5 Energy consumption vs. latency requirements

6.3. Cost of Edge Server VM Usage

As task completion time is closely linked to edge serve usage, the server VM usage cost under different task data sizes and latency requirement are determined. We investigate edge server VM usage cost in mobile-edge computing for different offloading strategies. Edge server VM usage cost is calculated based on occupation time of VM in edge server. The cost of VM usage is set to be 0.1 $/Hr.

Fig. 6 compares the VM cost among four different offloading methods, which is all offload, all offload + prioritized scheduling, Tao's Partial Offloading as well as Tao's Partial Offloading + prioritized scheduling strategies. As shown in Fig. 6, cost of edge server VM usage increase as task sizes increase from 20 MB to 200 MB, and partial offload has lower (30 % ~ 50%) cost compared to all offload computation method. Obviously, large task data size and all offload takes longer VM CPU time to process.

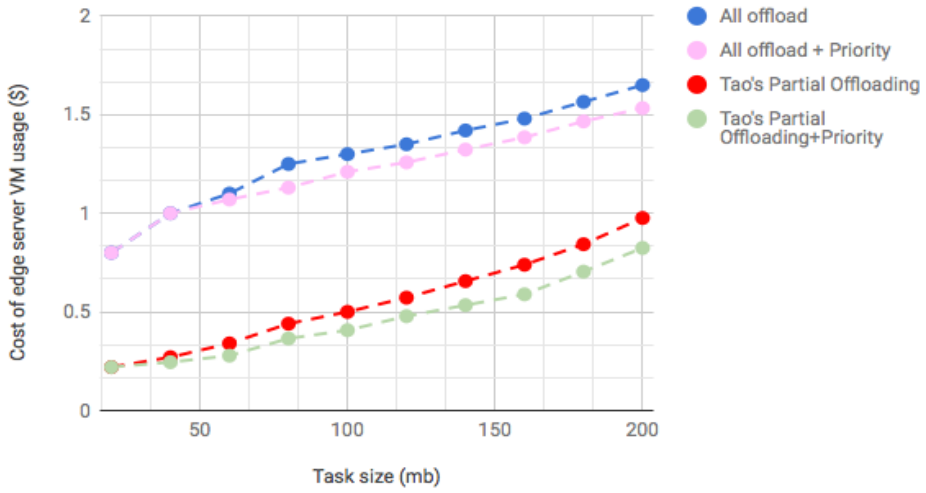

Fig. 6 Cost of edge server VM usage vs. task data sizes

Moreover, our result also shows that VM cost lowered (10% ~ 15%) by implementing prioritized scheduling strategy compared to all offload methods without prioritized strategy in edge server. Specifically, Tao's Partial Offloading + prioritized scheduling method has the lower VM usage cost compared to Tao's Partial Offloading methods without prioritized scheduling, this is consistent with the completion time analyze.

Similar results also found in VM usage cost vs. latency requirement. As shown in Fig. 7, offloading methods with prioritized scheduling strategy has better performance than the methods without prioritized. And the cost increases at some levels as the latency requirement getting lower in edge server.

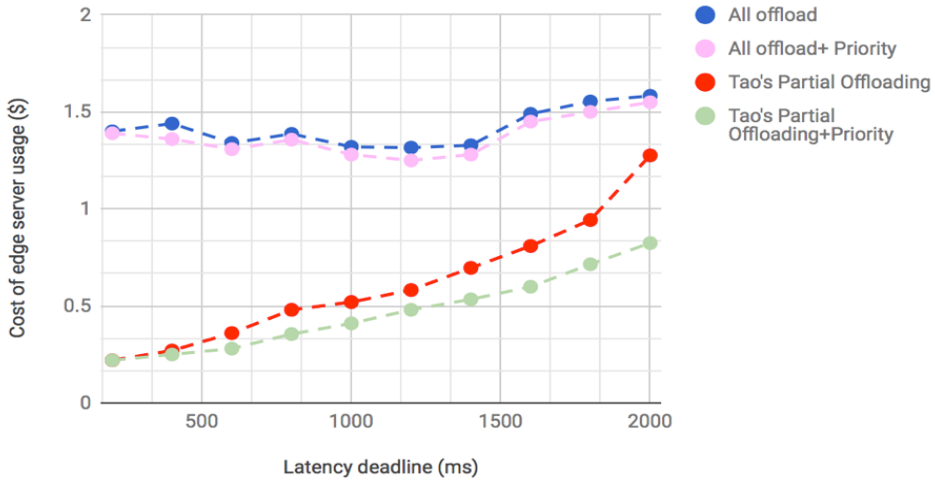

Fig. 7 Cost of edge server VM usage vs. latency requirements

6.4. Bonus Score

Bonus score is a metric used to evaluate QoE in cloud service. We use this score to represent the "benefit" to process a task based on its completion time and priority value. The Bonus score can be calculated as:

$$Bonus\ Score = \frac{(Latency\ Requirement - Completion\ Time) \times Priority\ Value}{Task\ size}$$

We calculate bonus score for all offload, all offload + prioritized scheduling, Tao's Partial Offloading as well as Tao's Partial Offloading + prioritized scheduling methods to evaluate the best performance offloading strategy. The bonus score vs. task data sizes and latency deadline can be found in Fig. 8 and Fig. 9.

Fig. 8 shows that, in general, when task sizes are small (20 MB ~ 60 MB), four offloading methods have similar bonus score achievement. When task sizes become larger, (i) Tao's Partial Offloading (with or without prioritized scheduling) have greater bonus score than all offload method. This is due to most the offloading task in partial offload methods has less completion time compared to all offload method, (ii) Tao's Partial Offloading with prioritized scheduling performs better in terms of bonus than

Tao's partial offload. This indicates that task scheduling according to priority values improves the throughput of successfully completed, which reduce the completion time of tasks with higher priority value and in turn increase the bonus score.

Similar trend found in bonus score vs. latency deadline as shown in Fig. 9. Greater bonus score achieved from four offloading methods when latency requirement ranges from 1600 ms to 2000 ms. However, the bonus score keeps constant or even getting lower when latency requirement set to be 2200 ms to 2600 ms. This is because nearly 80% of the tasks are offloaded from mobile devices to edge server when latency requirements increase to 1800 ms. Therefore, it can be observed that Tao's Partial Offloading and all offload has similar bonus score at 2000 ms latency requirement. The observed lower bonus scores at 2200 ms for Tao's Partial Offloading + prioritized may due to differential between latency requirement and completion time becomes less, which is consistent to the longer completion time shown in Fig. 3 when latency requirement get lower.

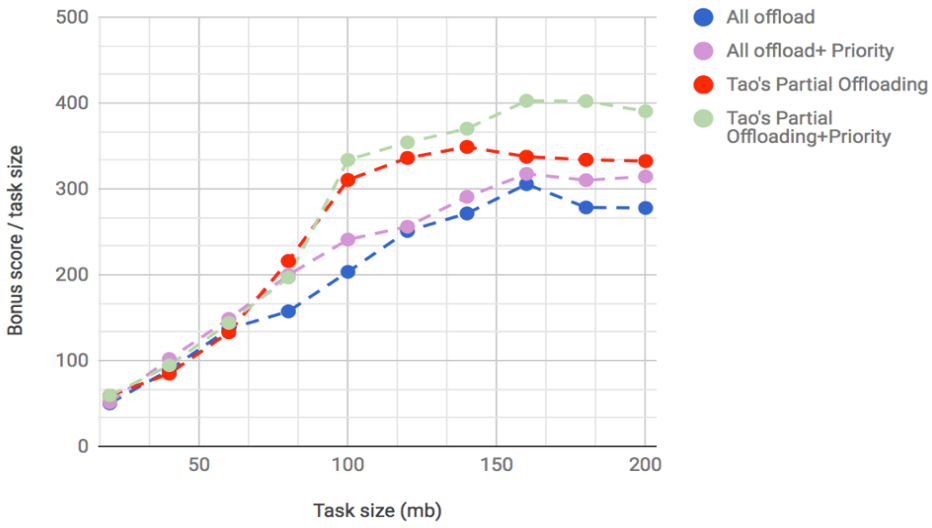

Fig. 8 Bonus score vs. task data sizes

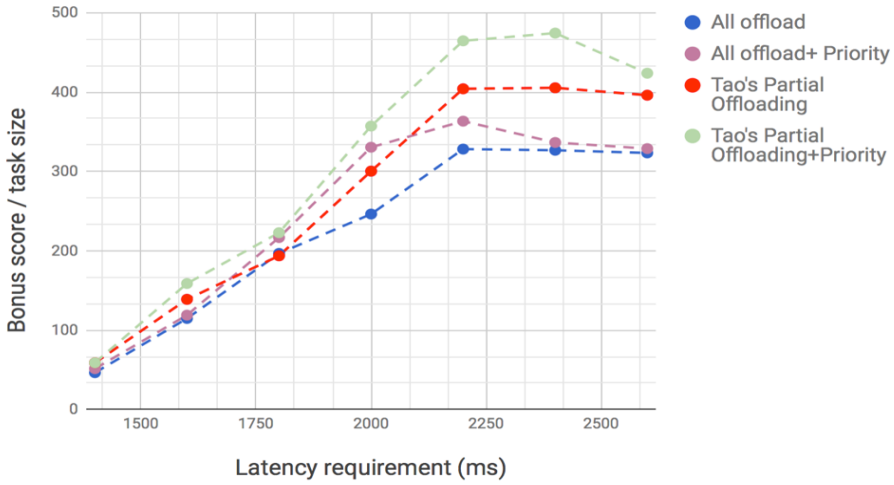

Fig. 9 Bonus score vs. latency requirements

6.5. Dynamic change thresholds in prioritized scheduling

In the previous sections, two thresholds value T_1 and T_2 in priority-based task scheduling in edge layer are fixed. However, the fixed value might not fit when system scale up and down. System scale up and down have various dimensions, here we discuss scale up and down with latency requirements changes. The dynamic change of thresholds value T_1 and T_2 is set to be T_1 = 25% average latency requirement, T_2 = 75% average latency requirement. Those two parameters are come up from the previous section, latency requirements are range from 200 ms ~ 2000ms. Therefore, T_1 = 300 ms equals to 25% of average latency requirements, T_2 = 800 ms equals to 75% of average latency requirements. In the following, we will discuss the performance for fix settings and dynamic setting in scale up and scale down systems.

6.5.1 Current latency requirements

In order to compare different fix settings with dynamic setting, we set up two fix settings, (i) current fix setting T_1 = 300 ms, T_2 = 800 ms, (ii) new setting T_1 = 1000 ms, T_2 = 3000 ms, and one dynamic threshold setting (iii) dynamic setting T_1 = 25% average latency requirements, T_2 = 75% average latency requirements. The current

latency requirement ranges from 200 ms ~ 2000 ms, task size is 140 MB. Performance results can be found in Fig. 10 to Fig. 12.

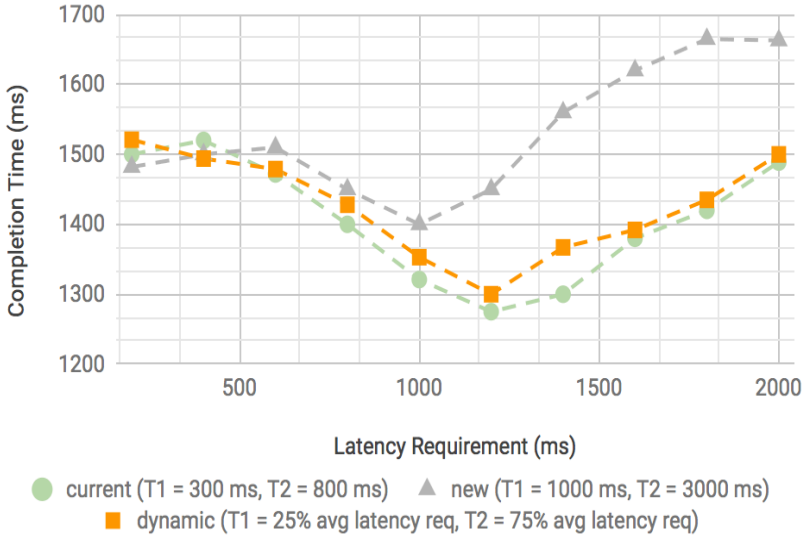

Fig. 10 Completion cost vs. latency requirements at current system

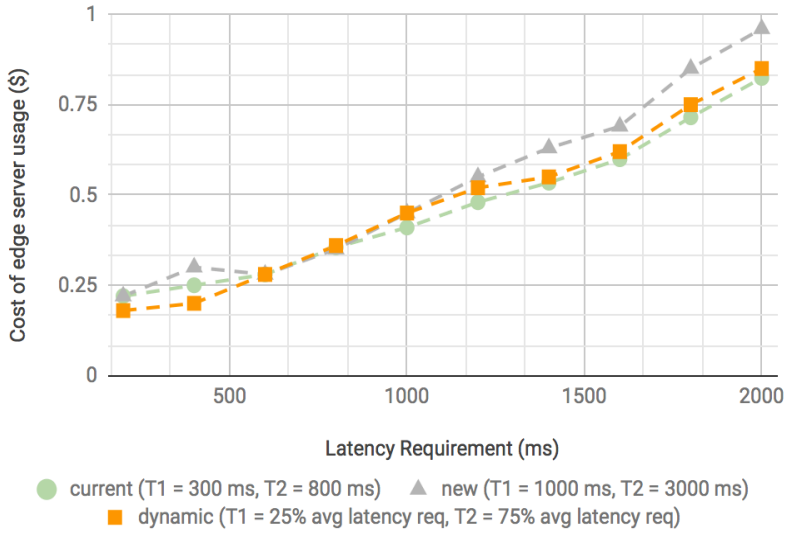

Fig. 11 Cost of edge server usage vs. latency requirements at current system

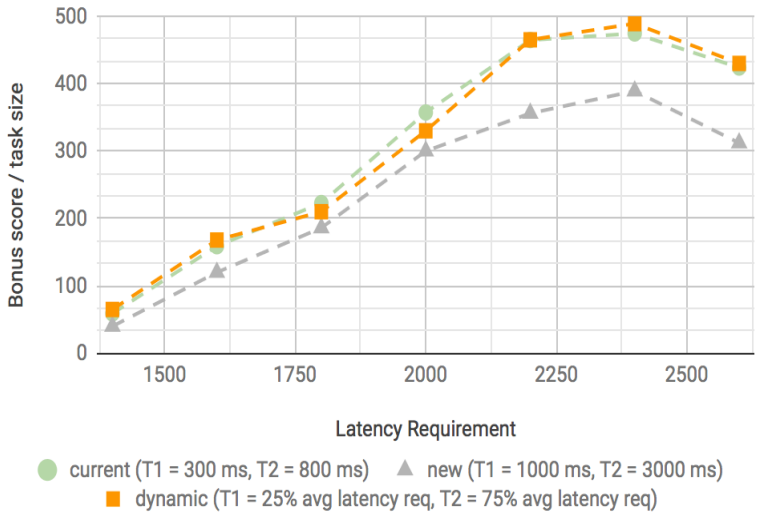

Fig. 12 Bonus score vs. latency requirements at current system

Above results show that current setting and dynamic setting has less task completion time, less edge server usage cost and better bonus score compared to new setting. This is because in new setting, T_2 setting is too large that most of the tasks will be fall into secondary category ($T_{est,i} + T_1 \leq wait_i \leq T_{est,i} + T_2$) when decide the service priority in edge server. That means no task will be placed into third category ($wait_i > T_{est,i} + T_2$), which causing congestion in second category. Some tasks which can be placed in third category would have less completion time because downgrade queue might have space, which could have less edge server usage cost and higher bonus score.

No significant performance difference observed between current setting and dynamic setting, because two parameters 25% and 75% of average latency requirements are proposed based on current setting in current latency range. And current setting is the optimal setting in current latency range based on experiment.

6.5.2 Scale up latency requirements

In scale up settings, latency range: 2000 ms ~10000 ms, task size = 300 MB. Thresholds settings are the same as in section 6.5.1. Performance evaluation can be found in Fig. 13 to Fig. 15.

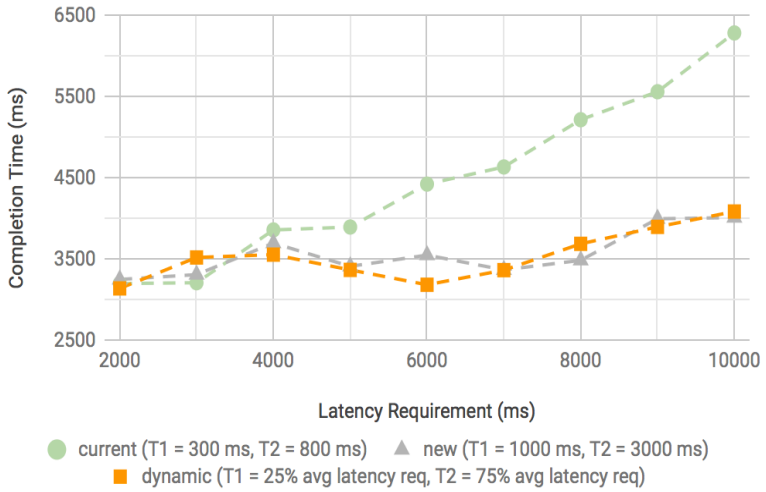

Fig. 13 Completion time vs. latency requirements at scale up system

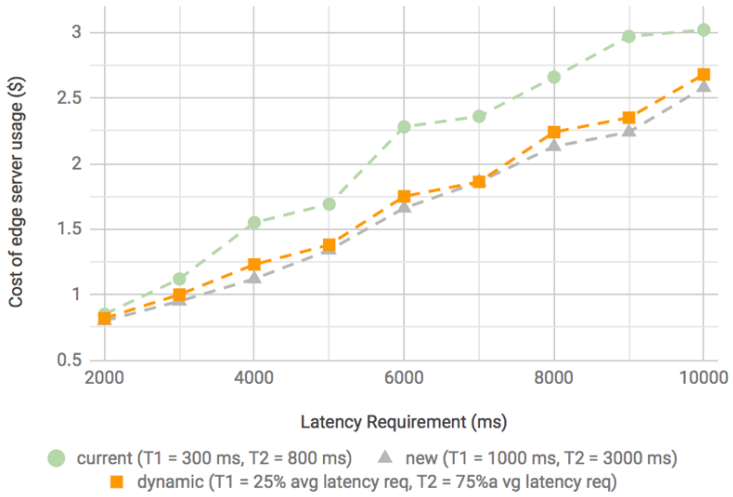

Fig. 14 Cost of edge server usage vs. latency requirements at scale up system

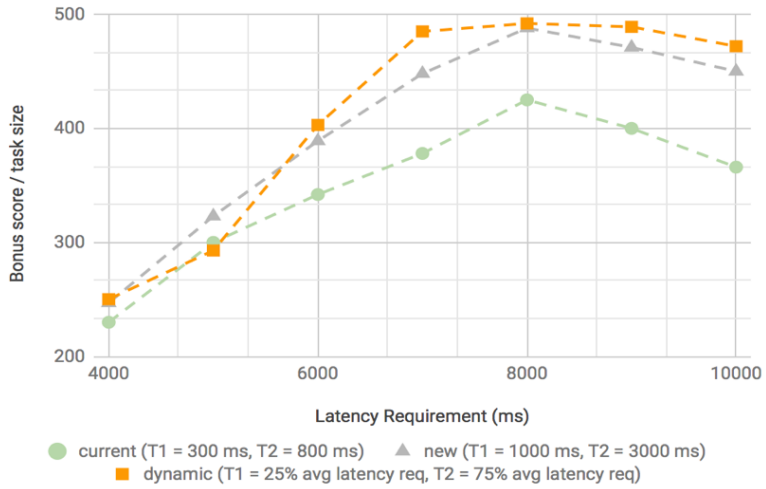

Fig. 15 Bonus score vs. latency requirements at scale up system

Results show that new setting and dynamic setting has less task completion time, less edge server usage cost and better bonus score compared to current setting. No significant difference in performance between new setting and dynamic setting. Current setting: T_1 = 300 ms, T_2 = 800 ms is not fit in scale up system.

In current setting, T_2 setting is too small that most of the tasks will be fall into third category ($wait_i > T_{est,i} + T_2$) when latency range scale up. That means no task will be placed into secondary category ($T_{est,i} + T_1 \leq wait_i \leq T_{est,i} + T_2$). Some tasks which have stringent latency requirements should be placed in secondary category would have less completion time. Because tasks will be placed into its subscribed priority queue without downgrade priority level, which could have less edge server usage cost and higher bonus score. No significant performance difference observed between new setting and dynamic setting, indicates that dynamic setting of two parameters 25% and 75% of average latency requirements can be used in scale up system.

6.5.3 Scale down latency requirements

Similarly, we also discuss the system performance in scale down latency requirements. In scale down settings, latency range: 200 ms ~1000 ms, task size = 20 MB. Thresholds settings are the same as in section 6.5.1. Performance evaluation can be found in Fig. 16 to Fig. 18.

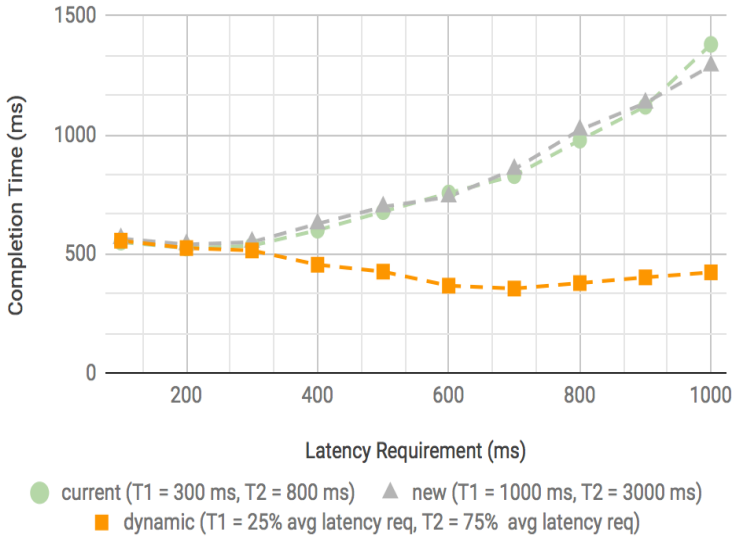

Fig. 16 Completion time vs. latency requirements at scale down system

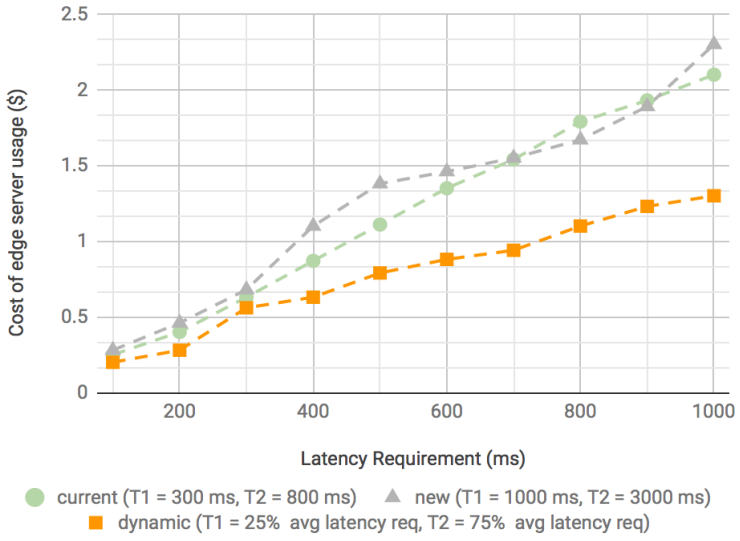

Fig. 17 Cost of edge server usage vs. latency requirements at scale down system

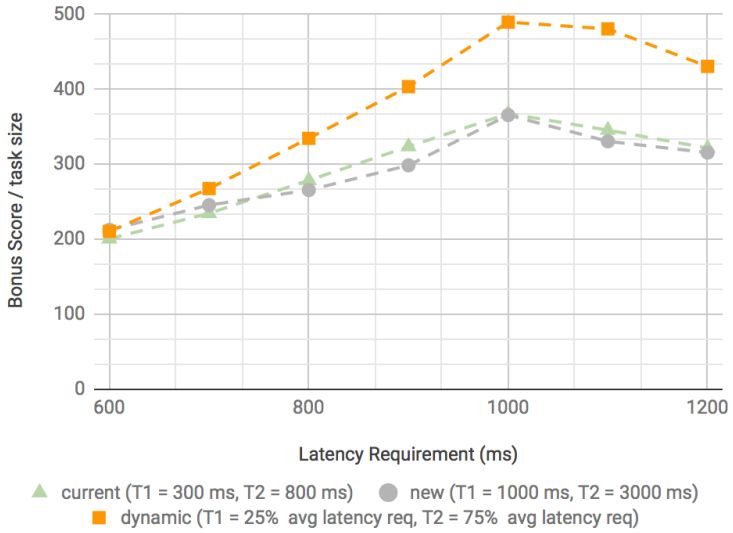

Fig. 18 Bonus score vs. latency requirements at scale down system

Results show that current setting and new setting requires more completion task completion time, more edge server usage cost and gain less bonus score compared to dynamic setting. No significant performance difference observed between current setting and new setting. Current setting: T_1 = 300 ms, T_2 = 800 ms and new setting: T_1 = 1000 ms, T_2 = 3000 ms are not fit in scale down system.

In current setting and new setting, both T_2 settings are too large for scale down system. Therefore, most of the tasks will be placed into secondary category ($T_{est,i} + T_1 \leq wait_i \leq T_{est,i} + T_2$), which causing congestion in second category. That means some tasks that have loose latency requirement in the range which can be placed into third category ($wait_i > T_{est,i} + T_2$) are also be placed into second category. So, completion time and cost of edge server usage increases due to inappropriate and inefficient task scheduling, and in turn lowers bonus score. Dynamic setting T_1 and T_2 realizes the optimal adjusting based on the priority of the tasks. Dynamic setting of two parameters 25% and 75% of average latency requirements can also be used in scale down system.

7. CONCLUSION

This paper proposes a joint computation offloading, priority-based task scheduling in a multi-user mobile-edge computing system, and evaluate its performance against existing computation offloading and scheduling schemes. Performance evaluation results show that proposed Joint Computation Offloading and Prioritized Scheduling strategy has reduced task completion time, and the cost of edge server VM usage. Specifically, the proposed scheme greatly improves QoS in terms of bonus score by increasing throughput of MEC system and guarantee SLA. Moreover, we also discuss dynamic change threshold values in prioritized scheduling algorithm, realizes optimal adjusting based on the priority of the tasks. Our study provides a viable approach that can be applied to other IoT-Edge systems. Future work would continue in the use of optimal computation offloading algorithm in mobile layers, and append a cloud layer in the mobile edge computing system for a complete research about task allocation and scheduling.

REFERENCE

[1] P. Milan, J. Jerome, Y. Valerie, and A. Sadayuki, "Mobile-edge computing introductory technical white paper," *White Paper*, 2014.

[2] X. Zhu, L. Yang, H. Chen, J. Wang, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 168-180, 2014.

[3] Y. Wen, W. Zhang, H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. of INFOCOM*, Orlando, FL, 2012, pp. 2716–2720.

[4] K. Kumar, and Y. Lu, "Cloud computing for mobile users: can offloading computation save energy," *Computer*, vol. 43, no. 4, pp. 51– 56, 2010.

[5] S. Kosta, A. Sucinas, H. Pan, R. Mortier, and X. Zhang, "ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. of INFOCOM*, Orlando, FL, 2012, pp. 945–953.

[6] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer (extended version)," *IEEE J. Sel. Areas Commun*., vol. 34, no. 5, pp. 1757-1771, May, 2016.

[7] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "The smartphone and the cloud: power to the user," In: Gris M., Yang G. (eds) *Mobile Computing, Applications, and Services. MobiCASE* 2010. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 76. Springer, Berlin, Heidelberg.

[8] O. Bouhali, H. Alnuweiri, "Resource Allocation and scheduling in Cloud Computing", in *Proc. Of the IEEE Int. Comput. Netw. Communs (ICNC)*, Maui, Hi, 2012, pp. 309 – 314.

[9] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun*., vol. 11, no. 6, pp. 1991–1995, 2012.

[10] J. Xie, L. Dan, L. Yin, Z. Sun, and Y. Xiao, "An energy-optimal scheduling for collaborative execution in mobile cloud computing," *in IEEE Int. Conf. Comput. Commun. (IEMCON)*, Vancouver, BC, Canada, 2015, pp 1-6.

[11] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile- edge computing," *IEEE Trans. Signal Inf. Process. Over Netw*., vol. 1, no. 2, pp. 89-103, Jun. 2015.

[12] S.-T. Hong and H. Kim, "QoE-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds," in *Proc. 13th Annu. IEEE Int. Conf. Sens. Commun. Netw. (SECON)*, London, U.K., 2016, pp. 1-9.

[13] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510-2523, Dec. 2015.

[14] M. Dakshayini and H. S. Guruprasad, "An Optimal Model for Priority based Service Scheduling Policy for Cloud Computing Environment", *Int. J Computer Applications*, vol. 32, no. 9, pp. 23-29, 2011.

[15] A. Ingole, S. Chavan, and U. Pawde, "An Optimized Algorithm for Task Scheduling based on Activity based Costing in Cloud Computing", in *IJCA Proc. on 2nd National Conference on Information and Communication Technology (NCICT)*, vol. 3, no. 3, pp. 34 -37, Nov., 2011.

[16] S. Ghanbari, M. Othman, M.R.A. Bakar, and W.J. Leong, "Priority- based Divisible Load Scheduling using Analytical Hierarchy Process", *APPL MATH INFORM SCI*, vol. 9, no. 5, pp. 2541- 2552, 2015.

[17] Tejaswiini Choudharik Melody Moh, and T.-S. Moh, "Prioritized Task Scheduling in Fog Computing," in *Proc. of the ACM Annual Southeast Conference (ACMSE),* Richmond, KY, Mar 2018.

[18] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 774 – 777, 2017.

[19] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, 2016, pp. 1451–1455.

[20] Calheiros, R. N. et al. "CloudSim: A ToolKit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software: Practice and Experience*, vol. 41, nol. 1 pp. 23 – 50, 2011.