Journal of Computer and Communications, 2019, 7, 94-104
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Edge-Cloud Collaborative Optimization Scheduling with Micro-Service Architecture

Qiuyan Liu[1], Jiajun Li[2], Huazhang Lv[1], Zhonghao Zhang[1], Mingxuan Li[1], Yi Feng[1]

[1]Institute of Network Technology, China Unicom Co. Ltd., Beijing, China
[2]Huasheng Terminal, China Unicom Co. Ltd., Beijing, China
Email: liuqy95@chinaunicom.cn, lijj49@chinaunicom.cn, lvhz7@chinaunicom.cn, zhangzh306@chinaunicom.cn, limx59@chinaunicom.cn, fengyi12@chinaunicom.cn

How to cite this paper: Liu, Q.Y., Li, J.J., Lv, H.Z., Zhang, Z.H., Li, M.X. and Feng, Y. (2019) Edge-Cloud Collaborative Optimization Scheduling with Micro-Service Architecture. *Journal of Computer and Communications*, **7**, 94-104.
https://doi.org/10.4236/jcc.2019.710009

Received: September 17, 2019
Accepted: October 9, 2019
Published: October 16, 2019

Abstract

The architecture of edge-cloud cooperation is proposed as a compromising solution that combines the advantage of MEC and central cloud. In this paper we investigated the problem of how to reduce the average delay of MEC application by collaborative task scheduling. The collaborative task scheduling is modeled as a constrained shortest path problem over an acyclic graph. By characterizing the optimal solution, the constrained optimization problem is simplified according to one-climb theory and enumeration algorithm. Generally, the edge-cloud collaborative task scheduling scheme performance better than independent scheme in reducing average delay. In heavy workload scenario, high blocking probability and retransmission delay at MEC is the key factor for average delay. Hence, more task executed on central cloud with abundant resource is the optimal scheme. Otherwise, transmission delay is inevitable compared with execution delay. MEC configured with higher priority and deployed close to terminals obtain more performance gain.

Keywords

Edge-Cloud Collaboration, Micro-Service, Scheduling Policy, Markov Process

1. Introduction

As digital applications proliferate, 5G network is deeply integrated with enhanced mobile broadband (eMBB), ultra reliable low latency communications (uRLLC), and massive machine type communication (mMTC) services, such as virtual reality/augmented reality, 4K/8K live video, artificial intelligence decision and big data analysis [1] [2] [3] [4]. Multiple access edge computing (MEC) is a promise technology which allows various access mode, high bandwidth, low latency services executed locally by deploying multiple access edge cloud at the edge of the network [5] [6]. As a simplified version of central cloud, MEC executes the cloud computing task locally by deep data inspection, which can reduce the transmission delay and avoid the flow storm towards backhaul and core network [7] [8] [9]. More and more broadband and low latency services are applied with the assistant of MEC instead of centralized cloud. However, MEC is deployed at the edge of network with limited resources including computing, storage, cache, even physical space, air-conditioning environment and energy consumption. Compared with MEC, central cloud is assumed to offers virtually unlimited computing capacity, massive storage and abundant cache in core data center (DC). Hence, the architecture of edge-cloud cooperation is proposed as a compromising solution that combines the advantage of MEC and central cloud [10] [11]. In the cooperation scenario, it is critical to utilize the limited resource efficiently and balance the workload between edge and cloud by offloading mechanism and resource management schemes.

Offloading mechanism has been studied extensively in mobile edge computing scenario [12]-[19]. Generally, there are three kinds of offloading policies aiming at minimal delay, minimal energy and balance between delay and energy. Liu adopt a Markov decision process approach to handle the delay and optimal computation offloading policy, incorporating different timescales in the task execution process [12] [13] [14]. Several strategies are proposed to minimize the energy consumption with predefined delay constrains [15] [16] [17] [18]. Kamoun suggested continuous optimization algorithms preconfigured offline and based on network status online [15] [16]. Resource allocation is considered in a MEC offloading system comprising multiple users sharing the same edge cloud [17] [18]. Furthermore, some offloading proposals are considered for energy efficiency in latency constrained [19] [20]. However, most of the literatures are assumed to solve the optimal cooperation between the mobile terminal and MEC platform with fading channels. In fact, the cooperation between MEC and central cloud is also significant in task scheduling [21]. In this scenario, not only backhaul bandwidth but also transmission delay with various MEC deployment positions are the critical factors.

In this research, we investigate problem of how to reduce the average delay of MEC application with cloud assistant in a more generally scenario. Compared with one dimension linear or parallel task execution topology, the realistic MEC platform allows both serial and parallel computation tasks executed. The parallel computation task execution is designed to support micro-service application. Nevertheless, most tasks cannot be divided into parallel elements completely. Some elements are executed with others' output as input. Hence, we build a grid topology model to describe the task generally. The model is consists of two dimensions topology. The micro-service sets are executed serially and the micro services are executed in parallel in each set. Then, we characterize the optimal solution to the optimization problem by one-climb theory and enumeration algorithm [20].

The rest of this paper is organized as follows. System model and problem formulation is presented in Section 2. Section 3 provides the characterization of the optimal solution to the optimization problem. In Section 4, some simulation results are listed to compare the performance gain in average delay. Finally, the conclusions of this paper and suggestions for future work are given in Section 5.

2. System Model and Problem Formulation

In this section, we present the model for tasks with micro-service architecture collaborated between MEC and cloud. Generally, an application task is a hybrid of services executed sequentially and in parallel.

1) Micro-Service Task Model

We assume that an application task Φ is presented by a sequence of micro-service sets with a linear topology. Each micro-service set consists of micro-services which can be executed in parallel. **Figure 1** illustrates the micro-service task model in a grid topology. S_0 and D are the initiator and destination of the application task, respectively. There are n sequentially executed micro-service sets $\{\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots, \boldsymbol{S}_n\}$ in the task, which can be modeled as a Poisson process. We denote the output of $\boldsymbol{S}_i$ as $\boldsymbol{\alpha}_i$ for $i = 1, 2, \cdots, n$. And $\boldsymbol{\alpha}_0$ is the input initialized by the initiator S_0. Each set $\boldsymbol{S}_i$ $(0 < i \leq n)$ is consist of m_i independent micro services $\{s_{i,j} \mid 0 < i \leq n, 0 < j \leq m_i\}$ which are executed in parallel. Specially, $\{w_{i,j} \mid 0 < i \leq n, 0 < j \leq m_i\}$ is the computing workload of the jth micro service in the micro-service set $\boldsymbol{S}_i$. However, micro-service set $\boldsymbol{S}_i$ $(0 < i \leq n)$ is executed based on the output of previous set $\boldsymbol{S}_{i-1}$ $(0 < i \leq n)$. Hence, notice that $\boldsymbol{\alpha}_i$ $(0 \leq i \leq n)$ is a non-zero matrix. Generally, there are three kinds of edge-cloud collaborative policy: local execution, partial offload and all offload. Hence, ξ is defined as the ratio of micro-service set that offloaded to MEC platform.

2) Execution Model

In this paper, we focus on the collaborative task execution between MEC and cloud. Specially, each micro service in the grid topology can be executed on the local MEC platform or offload to the cloud. In this configuration, we establish atomic modules involved during the collaboration between MEC and cloud.

MEC execution. If the jth micro service in the micro-service set $\boldsymbol{S}_i$ $(0 < i \leq n)$ is executed on the MEC platform, the completion time is given by

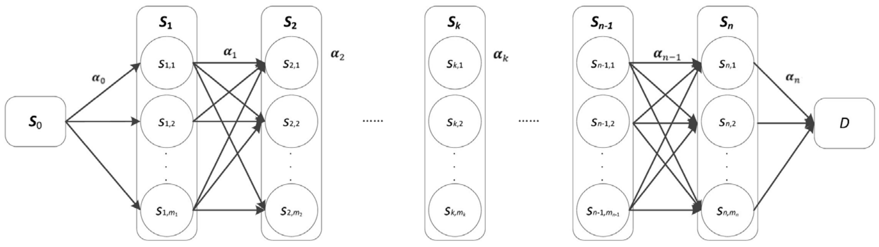

Figure 1. Illustration of the micro-service architecture model in a grid topology.

$$t_{ij}^{MEC}(\xi) = l_{ij}(\xi) * f_{MEC}^{-1} \tag{1}$$

and the energy consumption of MEC platform is given by

$$e_{ij}^{MEC} = t_{ij}^{MEC}(\xi) * p_{MEC} = \kappa_{ij} t_{ij}^{MEC}(\xi) f_{MEC}^{3} \tag{2}$$

where f_{MEC} is the computation power function related with CPU frequency, κ_{ij} is a constant related to the hardware architecture, and p_{MEC} is the energy consumption model proportional to f_{MEC}^{3}. Therefore, the completion time of MEC platform for micro-service $\boldsymbol{S}_i$ $(0 < i \le n)$ with m_i micro services executed in parallel is given by

$$t_{i}^{MEC}(\xi) = \max\{l_{ij}(\xi) * f_{MEC}^{-1} \mid j[1, m_i] \cap Z\} \tag{3}$$

Edge-cloud transmission. MEC transmits necessary input $\boldsymbol{\alpha}_i$ for micro-service set $\boldsymbol{S}_{i+1}$ when its buffer is full. The transmission bandwidth for MEC is defined as a constant B. L_{EC} is the transmission distance from MEC to cloud. Generally, the typical transmission delay of backhaul is a constant related with distance, e.g. 5us per kilometer. Then the loop transmission delay of output data $\boldsymbol{\alpha}_i$ $(0 \le i \le n)$ between MEC and cloud is given by

$$t_{i}^{EC}(\xi) = 2 * t_0 * L_{EC} * \alpha_i / B \tag{4}$$

where t_0 is the transmission delay per kilometer and B is the transmission rate of backhaul between MEC and central cloud.

The energy consumption of output data $\boldsymbol{\alpha}_i$ $(0 \le i \le n)$ transmission between MEC and cloud is given by

$$e_i = t_i^{t}(\xi) * p_t \tag{5}$$

where t_i^{t} is the transmission delay to MEC or central cloud, p_t is the energy consumption function with fixed parameter during the computation.

Cloud execution. Similarly, if the jth micro service in the micro-service set $\boldsymbol{S}_{i-1}$ $(0 < i \le n)$ is executed on the cloud, the completion time is given by

$$t_{ij}^{C}(\xi) = l_{ij}(\xi) * f_{C}^{-1} \tag{6}$$

and the energy consumption of cloud is given by

$$e_{ij}^{MEC}(\xi) = t_{ij}^{MEC}(\xi) * p_{MEC} = \kappa_{ij} t_{ij}^{MEC}(\xi) f_{MEC}^{3} \tag{7}$$

where f_C is the computation power function related with central CPU frequency, κ_{ij} is a constant related to the hardware architecture, and p_C is the energy consumption model proportional to f_C^3. Generally, cloud is more powerful and efficient than MEC with higher f_C. Therefore, the completion time of cloud for micro-service $\boldsymbol{S}_i$ $(0 < i \le n)$ with m_i micro services executed in parallel is given by

$$t_{i}^{C}(\xi) = \max\{l_{ij}(\xi) * f_{C}^{-1} \mid j[1, m_i] \cap Z\} \tag{8}$$

3) Markov Process

The collaborative task execution between MEC and cloud can be modeled by a directed acyclic graph $G = (V, A)$, with the finite node set V and arc set A, as shown in **Figure 2**. We denote the number of nodes and arcs as $|V|$ and $|A|$, respectively. Node *i* represents that the *i*th micro-service set $\boldsymbol{S}_i$ is executed on MEC platform and node *i'* represents that the *i*th micro-service set $\boldsymbol{S}_i$ is executed on cloud. We can find that $|V| = 2n + 2$ and $|A| = 4n$. However, cloud can complete the micro-service sets in advance without prior information from the output of MEC. Hence, the application task collaborated between MEC and cloud can be divided into two parts, as shown in **Figure 3**. The previous $i - 1$ nodes on cloud are independent from special MEC platform service. Hence, non-time limited and services independent task can be executed on cloud previously, such as pre-configuration and model training in machine leaning. The collaborative task execution between and cloud model is modified as the latter part, from node $i + 1$ or $i + 1'$ to node *n* or *n'* in **Figure 3**. Hence, the number of nodes $|V|$ and arcs $|A|$ can be modified as $|V| = 2n - i + 2$ and $|A| = 4n - 3i$.

State (i, j) represents the state where the current number of requests at MEC is *i* and at cloud is *j*. Let p_{ij} be the stationary probability of state (i, j).

Unlike cloud, computing and cache resource are limited at MEC. The subsequence service has to be transmitted to the cloud if its request buffer is full. Hence, the micro-service sets executed on the MEC platform can be expressed as a modified M/M/c/K queue with cache capacity K_1. And the arrival rate λ_1 and the response completion time obey negative exponential distribution with parameter as μ_1.

$$\mu_{1(\xi)} = \mathbb{E}\left[\frac{1}{t^{MEC}(\xi) + t^{E}(\xi)}\right] \tag{9}$$

where $\mathbb{E}[\ ]$ is the average operator.

Then, the micro-service sets executed on cloud can be modeled as a modified M/M/c/K queue with cache capacity K_2. And the arrival rate λ_2 and the response completion time obeys negative exponential distribution with parameter as μ_2.

$$\mu_2(\xi) = \mathbb{E}\left[\frac{1}{t^{C}(\xi) + t^{EC}(\xi)}\right] \tag{10}$$

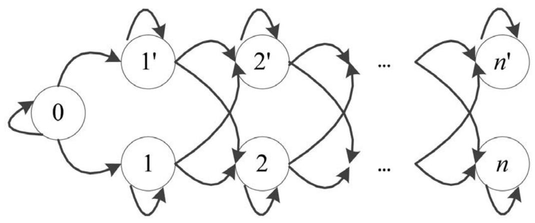

Figure 2. Markov model of edge-cloud collaboration.

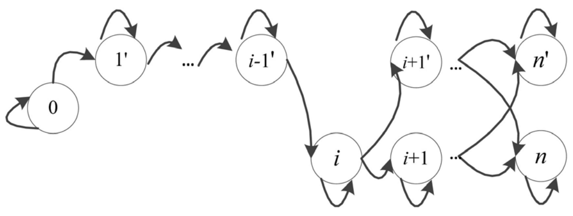

Figure 3. Modified Markov model of edge-cloud collaboration.

3. Characterization of Optimal Solution to Task Scheduling Policy

The arc of the adjacent modes, *i* and *j*, is associated with the nonnegative delay $t_{i,j}$ for a corresponding service. Under this framework, we can transform the edge-cloud collaborative scheduling problem to the shortest distance route problem between node *i* and *D*, subject to the constraint that the expected completion time of that path should be less than or equal to the energy ceiling. A path p is feasible if the expected energy consumption satisfies the energy constraint. A feasible path p^* with the minimum time delay is the optimal solution among all the feasible paths. Mathematically, it can be formulated as a constrained shortest path problem,

$$\min_{p\in\mathcal{P}} \mathbb{E}\left[T(\xi)=\sum\nolimits_{(i,j)\in p} t_{i,j}(\xi)\right] \tag{11}$$

s.t.

$$\mathbb{E}\left[\sum\nolimits_{(i,j)\in p} e_{i,j}(\xi)\right]\le E_{max} \tag{12}$$

$$\sum\nolimits_{i=1}^{n}\sum\nolimits_{j=1}^{n} p_{ij}=1 \tag{13}$$

$$p_{ij}>0,\forall i,j \tag{14}$$

$$0\le\xi\le 1 \tag{15}$$

where $\mathcal{P}$ is the set of all possible paths from node *i* to *D* and $E_{\max}$ is the maximal energy consumption of the entire application. Since each micro service can be executed by offloading to MEC or on cloud, there are 2^{n-i} possible options for the solution. This constrained optimization problem is shown to be NP-complete. Based on the optimality of the one-climb policy, the minimal delay optimal execution only migrates once from MEC to the cloud after node *i*. Then, we design an efficient algorithm for task scheduling. We can enumerate all the paths under the one-climb policy rather than all the 2^{n-i} paths [20]. We define $\mathcal{P}'$ as the set of all the paths under one-climb policy. There are $\left((n-i+1)(n-i)/2\right)+1$ paths in $\mathcal{P}'$, thus the optimal solution is available by reformulating $\boldsymbol{T}$ into a series of linear programming problems with non-convex $\boldsymbol{T}$.

The probability flows from the previous level to the current level are captured by the matrix $\boldsymbol{A}_0$ with fixed ξ, for queue length from 0 to K_1, and by the matrix $\boldsymbol{A}_0=-\lambda\xi\boldsymbol{I}$ for the other levels.

$$\boldsymbol{A}_0=-\lambda\xi\boldsymbol{I} \tag{16}$$

where $\boldsymbol{I}$ is the identity matrix. The queue length keeps the same with expression

$$\boldsymbol{A}_1=\begin{bmatrix} \mu_1+\lambda & (\xi-1)\lambda & 0 & & 0 \\ -\mu_2 & \mu^*+\lambda & (\xi-1)\lambda & \cdots & 0 \\ 0 & -\mu_2 & \mu^*+\lambda\xi & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mu^*+\lambda \end{bmatrix} \tag{17}$$

and

$$\boldsymbol{A}_1' = \begin{bmatrix} \lambda & (\xi-1)\lambda & 0 & & 0 \\ -\mu_2 & \mu^* + \lambda & (\xi-1)\lambda & \cdots & 0 \\ 0 & -\mu_2 & \mu^* + \lambda & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mu^* + \lambda \end{bmatrix} \tag{18}$$

$$\boldsymbol{A}_1^* = \begin{bmatrix} \mu_1 + \lambda & -\lambda & 0 & & 0 \\ -\mu_2 & \mu^* + \lambda & -\lambda & \cdots & 0 \\ 0 & -\mu_2 & \mu^* + \lambda & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mu^* \end{bmatrix} \tag{19}$$

where $\boldsymbol{A}_1'$ and $\boldsymbol{A}_1^*$ are matrix for state (0, 0) and (*K*, *K*).

Finally, the matrix from the next state to the current state is given by

$$\boldsymbol{A}_2 = \begin{bmatrix} -\mu_1 & 0 & 0 & & 0 \\ \mu_2 & -\mu^* & 0 & \cdots & 0 \\ 0 & \mu_2 & -\mu^* & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\mu^* \end{bmatrix} \tag{20}$$

Then the average delay are given by

$$\min_{p\in\mathcal{P}} \mathbb{E}\left[\boldsymbol{T}(\xi)\right] = \min_{p\in\mathcal{P}} \mathbb{E}\left[\frac{1}{\mu_1}\sum_{i=1}^{K_1}\sum\nolimits_{j=1}^{K_2}(i+1)p_{ij} + \frac{1}{\mu_2}\sum\nolimits_{j=1}^{K_2}(j+1)p_{K_1-1,j}\right]$$

s.t.

$$\begin{cases} \mathbb{E}\left[\sum\limits_{(i,j)\in p} e_{i,j}(\xi)\right] \le E_{max} \\ \sum\limits_{i=1}^{n}\sum\nolimits_{j=1}^{n} p_{ij} = 1 \\ p_{ij} > 0, \forall i,j \\ 0 \le \xi \le 1 \end{cases} \tag{21}$$

Searching space of the one-climb policy is much smaller than that of the brute-force search. Detailed procedures for solving minimal delay $\boldsymbol{T}$ are summarized as follows.

One-dimensional search algorithm

```
set $\xi = 0$, and $\Phi$ as a sufficiently large integer
for $\varphi = 0:1:\Phi$ do
    ($T$,$E$) = Shortest Path ($S_0$, $D$, $t$, $\xi$)
    //solve the shortest path in terms of delay with a fixed $\xi$
        if $E \le E_{\max}$
            return $T$
        else
            update the offload rate $\xi$ as $\xi + 1/\Phi$
        end
find the optimal solution $\xi^*$ for minimal delay $T(\xi^*)$
```

4. Performance Evaluation

In this section, we evaluated the performance of the proposed edge-cloud collaborative task scheduling scheme with independent task scheduling scheme. In this simulation, it is assumed that the input data packet obeys Poisson process. Compared with the independent scheme, the workload ratio ξ is adaptive to obtain the minimal delay. **Figure 4** illustrated the average delay comparison of cooperative and independent schemes with ARQ (Automatic Repeat-Request). Generally, edge-cloud collaborative scheme performs better than the independent scheme. The average delay gap is reduced as the workload increases with higher task density λ and less buffer size. In heavy workload scenario, high blocking probability with intensive retransmission is the most significant factor in average delay. Hence, the heavy workload case is a resource limited case. There is no extremely obvious difference whether MEC is configured with higher priority or not. Otherwise, high and inevitable transmission delay is a more critical factor in sparse task scenario. In this resource sufficient case, more task is executed at MEC platform until its buffer is full. Cooperative task scheduling obtains more performance gain than the independent scheme. **Figure 5** illustrated the average delay comparison between different task scheduling policies with MEC deployed in edge DC and local DC. Compared with execution delay, transmission delay is a more significant factor in edge-cloud collaborative task scheduling with MEC deployed in different DC. Hence, the average delay is increased extensively with MEC deployed on a higher layer, especially with insufficient buffer size.

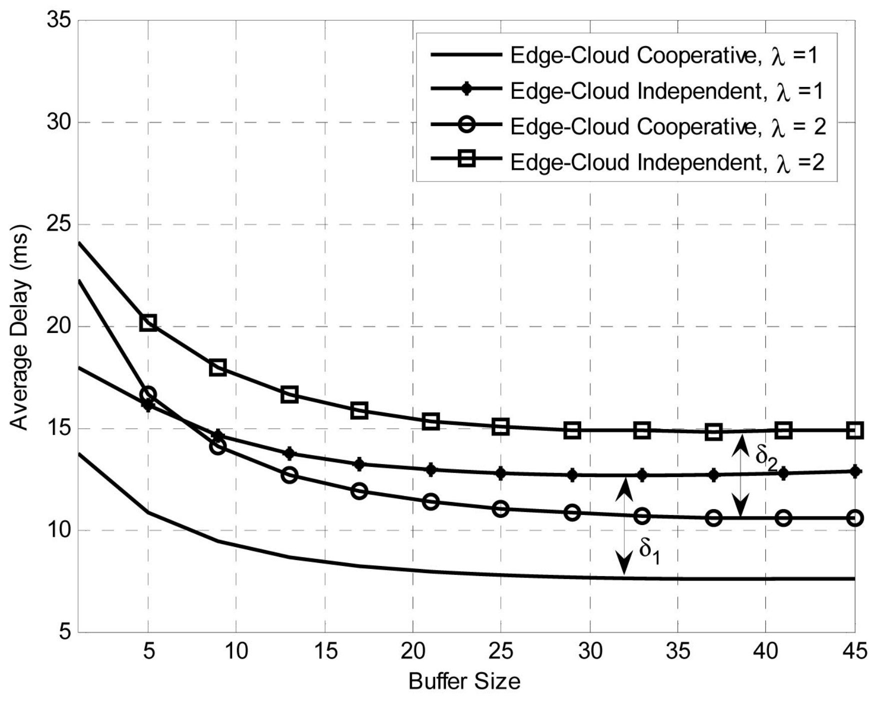

Figure 4. Average delay comparison between different task scheduling policies with ARQ and various λ.

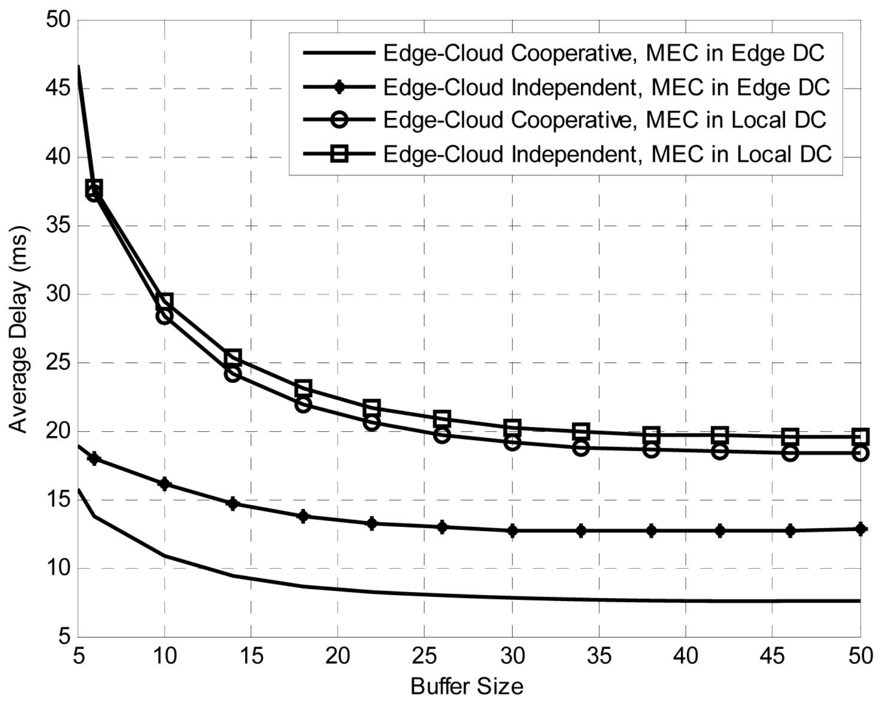

Figure 5. Average delay comparison between different task scheduling policies with different MEC deployment position.

5. Conclusions

In this paper we investigated the problem of how to reduce the average delay of MEC application by collaborative task scheduling. The collaborative task scheduling is modeled as a constrained shortest path problem over a acyclic graph. By characterizing the optimal solution, the constrained optimization problem is simplified according to one-climb theory and enumeration algorithm. Generally, the edge-cloud collaborative task scheduling scheme performs better than independent scheme in reducing average delay. In heavy workload scenario, high blocking probability and retransmission delay at MEC is the key factor for average delay. Hence, more task executed on central cloud with abundant resource is the optimal scheme. Otherwise, transmission delay is inevitable compared with execution delay. MEC configured with higher priority and deployed close to terminals obtain more performance gain.

For future work, we will consider various extensions of this work. First, more mathematical models of practical details in cloud computing are necessary to be considered in edge-cloud collaborative execution. Second, power saving on MEC and cloud platform is also important. In addition, the task workflow topology can be extended into more general modes.

Acknowledgements

The authors would like to thank the reviewers for their insightful comments. This work was support by Key R & D Program (Z91170GZX03011) grant from the Ministry of Science and Technology, Innovation and Development of Engineering Project of Ministry of Industry and Information Technology.

Funding

This work was support by Key R & D Program (Z91170GZX03011) grant from the Ministry of Science and Technology, Innovation and Development of Engineering Project of Ministry of Industry and Information Technology.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cisco System (2019) Cosic Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022 White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[2] Barbarossa, S., Sardellitti, S. and Di Lorenzo, P. (2014) Communicating While Computing: Distributed Mobile Cloud Computing over 5G Heterogeneous Networks. *IEEE Signal Processing Magazine*, **31**, 45-55. https://doi.org/10.1109/msp.2014.2334709

[3] Hu, Y., Patel, M., Sabella, D., Sprecher, N. and Young, V. (2015) Mobile Edge Computing: A Key Technology towards 5G-First Edition.

[4] Markakis, E.K., Karras, K. and Sideris, A. (2017) Computing, Caching, and Communication at the Edge: the Cornerstone for Building a Versatile 5G Ecosystem. *IEEE Communications Magazine*, **55**, 152-157. https://doi.org/10.1109/mcom.2017.1700105

[5] Khan, A.R., Othman, M., Madani, S.A. and Khan, S.U. (2014) A Survey of Mobile Cloud Computing Application Models. *IEEE Communications Survey & Tutorials*, **16**, 393-413. https://doi.org/10.1109/surv.2013.062613.00160

[6] Mao, Y., You, C., Zhang, J., Huang, K. and Letaief, K.B. (2017) A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, **19**, 2322-2358. https://doi.org/10.1109/comst.2017.2745201

[7] Abbas, N., Zhang, Y. and Taherkordi, A. (2018) Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*, **5**, 450-465. https://doi.org/10.1109/jiot.2017.2750180

[8] Mtibaa, A., Fahim, A., Harras, K.A. and Ammer, M.H. (2013) Towards Resource Sharing in Mobile Device Clouds: Power Balancing across Mobile Devices. *ACM SIGCOMM Workshop on Mobile Cloud Computing*, 51-56. https://doi.org/10.1145/2491266.2491276

[9] Kliazovich, D. and Granerlli, F. (2008) Distributed Protocol Stacks: A Framework for Balancing Interoperability and Optimization. *IEEE International Conference on Communications (ICC) Workshop*, 241-245. https://doi.org/10.1109/iccw.2008.51

[10] Kosta, S., Aucinas, A., Hui, P., Mortier, R. and Zhang, X. (2012) ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. *IEEE INFOCOM*, 945-953. https://doi.org/10.1109/infcom.2012.6195845

[11] Li, Z., Wang, C. and Xu, R. (2001) Computation Offloading to Save Energy on Handheld Devices: A Partition Scheme. *International Conference on Compliers*,

Architecture, and Synthesis for Embedded Systems, 238-246. https://doi.org/10.1145/502251.502257

[12] Liu, J., Mao, Y., Zhang, J. and Letaief, K.B. (2016) Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems. *2016 International Symposium on Information Theory*, 1451-1455. https://doi.org/10.1109/isit.2016.7541539

[13] Liu, J., Bai, B., Zhang, J. and Letaief, K.B. (2017) Cache Placement in Fog-RANs: From Centralized to Distributed Algorithms. *IEEE Transactions on Wireless Communications*, **16**, 7039-7051. https://doi.org/10.1109/twc.2017.2737015

[14] Mao, Y., Zhang, J. and Letaief, K.B. (2016) Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications*, **34**, 3590-3605. https://doi.org/10.1109/jsac.2016.2611964

[15] Kamoun, M., Labidi, W. and Sarkiss, M. (2015) Joint Resource Allocation and offloading Strategies in Cloud Enabled Cellular Networks. *IEEE International Conference on Communications*, 5529-5534. https://doi.org/10.1109/icc.2015.7249203

[16] Labidi, W., Sarkiss, M. and Kamoun, M. (2015) Joint Multi-User Resource Scheduling and Computation Offloading in Small Cell Networks. 201*5 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications*, 1093-1098. https://doi.org/10.1109/wimob.2015.7348043

[17] You, C. and Huang, K. (2016) Multiuser Resource Allcoation for Mobile-Edge Computational Offlaoding. 2016 *IEEE Global Communications Conference*, 1-6. https://doi.org/10.1109/glocom.2016.7842016

[18] Liang, Z., Liut, Y., Lok, T. and Huang, K. (2019) I/O Interference Aware Multiuser Computation Offloading for Virtualized Edge Computing. 2019 *IEEE International Conference on Communications*, 1-30. https://doi.org/10.1109/icc.2019.8762042

[19] Munoz, O., Pascual-Iserte, A. and Vidal, J. (2015) Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading. *IEEE Transactions on Vehicular Technology*, **64**, 4738-4755. https://doi.org/10.1109/tvt.2014.2372852

[20] Zhang, W., Wen, Y. and Wu, D.O. (2013) Energy-Efficient Scheduling Policy for Collaborative Execution in Mobile Cloud Computing. *IEEE INFOCOM*, 190-194. https://doi.org/10.1109/infcom.2013.6566761

[21] Beraldi, R., Mtibaa, A. and Alnuweiri, H. (2017) Cooperative Load Balancing Scheme for Edge Computing Resources. 2017 2nd *International Conference on Fog and Mobile Computing*, 94-100. https://doi.org/10.1109/fmec.2017.7946414