ISSN: 2664-9640

An edge-cloud collaborative task offloading model with minimal latency in edge cloud

Jiaqi Zhang [a], Ruijuan Zheng [b, *]

School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China.

[a] jqzhang@stu.haust.edu.cn, [b] zhengruijuan@haust.edu.cn

Abstract

Transmitting the tasks generated by the user device to an edge cloud for processing is an useful way to break even the confine of the user devices' own computing resources and enhance its processing capability. However, processing a task by an edge cloud or the user equipment itself solely, which will cause the edge cloud computing resources or the user equipment computing resource to be idle. Faced with this problem, we consider how to unify user device and edge cloud to form an edge-cloud integrated collaborative task processing system. We focus on tasks generated by data partitioned oriented applications (DPOAs). These tasks do not have much internal dependencies and can be arbitrarily divided. This kind of task is very common in our life, such as data compression, video transmission, virus scanning, etc. We use the shortest path method to determine which edge cloud to offload the task will cost the least time firstly. After that, we use model derivation to calculate how much data is offloaded to the edge cloud, which can further reduce the waiting time difference between the edge cloud and the user device. Finally, the edge-cloud collaborative processing model was analyzed by numerical experiments from multiple perspectives.

Keywords

Task offloading; Edge-cloud collaborative; Shortest path; Edge cloud.

1. Introduction

The number of smart devices connected to the network has been growth explosively. The network restricted bandwidth resource can no longer meet the increasing data requests. In addition, user devices are limited due to its physical size, its energy power, storage capacity, and computing power are limited. If a computationally intensive task is processed by the user device itself, a better processing result will not be obtained[1][2]. Faced with this problem, the tasks that need to be processed are processed by the user device itself or offloaded to an edge cloud near the task source for processing, which not only break through the restricted computing power of the user device, but also reduces bandwidth pressure on the central network [3]. Task offloading means that the tasks generated by the user device are transmitted to the edge cloud for processing through the network. Task offloading is an effective way for user device to break through its own computing power limitations and improve its computing power [4].

In recent years, some experts and scholars have achieved some results in task offloading. Guo et al. [5] optimized the task offloading problem by optimizing the clock frequency and the transmission power. Based on these optimization sub-problems, proposed their own task offloading scheme. Chen [6] considered the multiple user devices task offload scenario, and formulated a game problem. Moreover, the game was proved that it had a NEP, and an offloading mechanism was proposed to realize efficient offloading. However, these studies treat

ISSN: 2664-9640

user device and edge cloud as separate processing devices, in other words, tasks are either processed by the device itself or processed by an edge cloud. This causes the user device or edge cloud to be in an idle state when tasks are being processed, thereby reducing the overall utilization of computing resources.

Faced with this problem, we consider how to unify user device and edge cloud into an edge-cloud collaborative task processing network. The tasks processed by the edge-cloud collaborative processing mode to further reduce task processing time cost delay. In our paper, the tasks generated by data partitioned oriented applications (DPOAs) is mainly considered. These tasks do not have much internal dependencies and can be arbitrarily divided into subtasks, such as video transmission, file compression etc. When a user device generates a task, it use the shortest path method to determine which edge cloud to offload the task for processing will cost the least time firstly. After that, we use model derivation to calculate how much data is offloaded to the edge cloud, which can minimize the waiting time after the user device and the edge cloud have processed the task, thereby further reducing the task processing time cost delay. Finally, the edge-cloud collaborative task processing model was analyzed by numerical experiment.

The main contributions includes:

1. We propose an edge-cloud collaborative task processing mode, and the time cost and energy cost when processing a task was modeled.

2. We first used the shortest path method to determine which edge cloud the was offloaded for processing the task, and then proposed our scheme for the specific offloading amount of data.

3. We analyzed the performance of the edge-cloud collaborative processing model by numerical experiments from multiple perspectives.

The rest part are as follows: In Section 2, the system model was modeled. In Section 3, the optimization problem and solution scheme were proposed. The edge-cloud collaborative offloading model was analyzed from multiple perspectives by numerical experiments in Section 4. The Summary was in Section 5.

2. Related Work

In the edge cloud, some experts have proposed a number of different task offloading schemes. At present, some researchers focus on meta-task offloading, these tasks cannot be divided again in the process of offloading for processing. These studies were described in the previous section; Some graduate students are oriented to the separable task offloading which can be divided. When the task is processed, the edge cloud and the user device are all can be used for collaborative processing, so as to further reduce the task processing delay. Now, we analyze the research status of oriented separable task.

In the study of task offloading which considering task partitioning. Golchay et al. [7] optimized an task processing algorithm based on machine learning. The tasks generated by user device are regarded as multiple task components. The ant colony optimization algorithm is used to find the best solution for the problem, and then the tasks are partitioned to achieve collaborative processing. Debnath et al. [8] studied the scheduling algorithm of distributed application collaborative processing. they analyzed the dependencies of each subtask component, network state and other factors, and processes multiple components in the application through multiple servers, so as to realize collaborative processing. Sladana et al. [9] minimized the service delay, an approach to offload computing tasks using shared resources is proposed by considering the autonomous interaction between the operator (who is responsible for allocating computing resources). It is found that most of the current collaborative deployment achievements are not perfect in service slicing and other aspects. The above scholars put forward different opinions and schemes for task collaborative processing.

ISSN: 2664-9640

In this paper, we focus on tasks for data oriented applications, as they have very little interdependence in task internal and can be arbitrarily divided, such as data compression, video transmission, file scanning, etc. We consider the user device and the edge cloud to form a collaborative edge-cloud task processing system. After the user device generates a task to be processed, part of the data can be offloaded to the edge cloud through the network, so as to achieve the effect of edge-cloud collaborative processing, and further reduce the task processing delay. In the process of solving this problem, we use the shortest path method to determine which the edge cloud is used for processing the task, and then calculate the amount of offloaded data, so as to further reduce the delay of task processing. Below, we will introduce our work in detail.

3. System Model

In this section, scenario model, application model, local computing model, communication model and edge cloud computing model will be introduced specifically.

3.1. Scenario Model

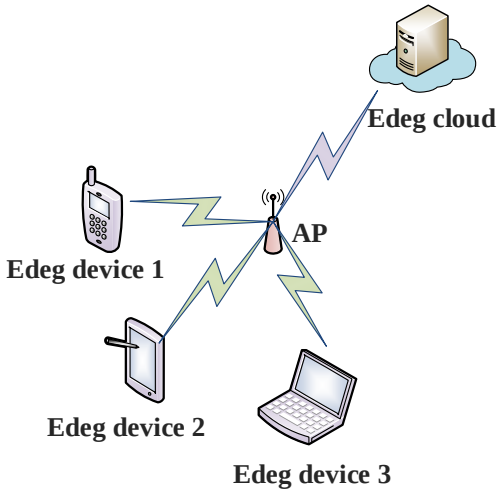


Fig. 1 System diagram of Edge-Cloud collaborative processing mode

In the hypothetical edge cloud system, $\mathcal{N}=\{1,\ldots,N\}$ and $\mathcal{E}=\{1,\ldots,E\}$ are used to denote the set of edge devices and edge clouds respectively. Each user device n generates a pending task that need to be processed periodically. The task can be handled either by the user device itself or by an edge cloud. In this paper, we mainly considered computing intensive tasks, and these tasks are separable, especially for data partitioned oriented applications (DPOAs) [10].

As shown in Fig. 1, it contains one edge cloud and three edge devices. When the user device 1 generates a task, it can be processed by the device itself, or offloaded to an edge cloud for processing. However, if the task is processed by the device itself or the edge cloud individually. In this way, the computing resources of the user device or the edge cloud will be idle when the task is being processed, which resulting in a waste of computing resources. In our task processing model, the task which need to be processed is offloaded the part of data to an edge cloud for processing, thereby forming an edge-cloud collaborative processing mode. $\eta_e, 0 \le \eta_e \le 1$ represents the portion of the data that is offloaded to edge cloud for processing. Therefore, the $1-\eta_e$ part of the tasks to be processed is processed by the device itself in local. At the same time, the rest η_e part is offloaded to an edge cloud for processing. The task is processed by the user device itself and the edge cloud, and reduce the task processing time.

As shown in Fig. 2, we use green region, purple region and red region to represent the time consumed by the local processing, the time consumed by the transmission of the task from the

ISSN: 2664-9640

user device to an edge cloud and the time consumed by the processing of the task on the edge cloud respectively. Both modes A and B are all processed by the user device and an edge cloud cooperatively. However, the amount of data offloaded to the edge cloud in mode A is small, so that there is still a lot of data left unprocessed by the user device after the edge cloud has processed the data. Different from mode A, mode B calculates the amount of data offloaded to the edge cloud after calculation, so that the time required for the edge cloud to process the data is the same as the time required for the user device to process the data. Therefore, reducing the time difference between the user device and the edge cloud to complete the task has an important impact on reducing the task processing delay. In this paper, the amount of data offloaded to the edge cloud will be analyzed, and the optimal task offloading ratio scheme will be proposed.

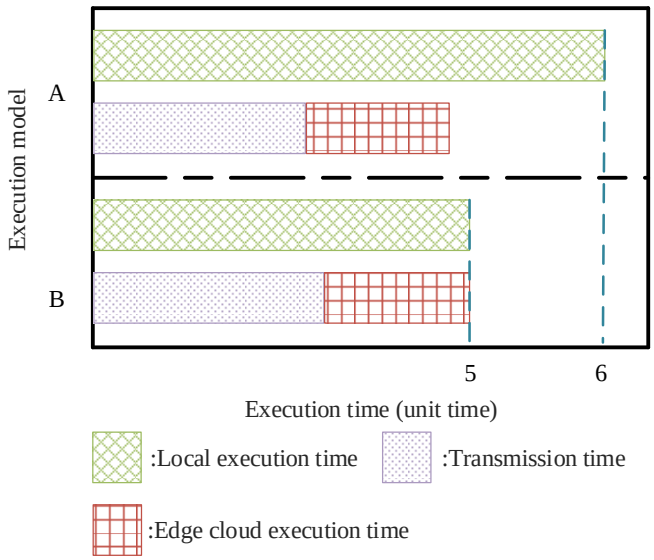


Fig. 2 The influence of processing completion time with different offloading data volume

3.2. Local Computing Model

The task denote by $M \triangleq < D, L >$, D is the data volume of the task (bit), L is the required total CPU cycles to complete the task. The CPU frequency of the edge device n is denoted by f_n (CPU cycles per unit time). As we known, when the device has many tasks to handle, the actual processing capacity is reduced. The ratio of actual availability of the computing power is represented by γ_n. Hence, if a task is processed by device itself, the processing time can be expressed by:

$$T_n^{exe} = \frac{(1-\eta_e)L}{\gamma_n f_n} \tag{1}$$

The energy cost can be expressed by:

$$E_n^{exe} = p_n^{exe} T_n^{exe} \tag{2}$$

In the task processing phase, the task processing power of the edge device is represented by p_n^{exe}.

3.3. Communication Model

When a task is offloaded to an edge cloud, the task should be transmitted to the wireless access point (AP), next the AP transmit the task to an edge cloud for processing. Note, the transmission cost between the AP and an edge cloud is ignored [11], so we only consider the transmission

ISSN: 2664-9640

time cost between the edge device to an AP. The uploading rate of the edge device transmits the task to an AP can be expressed as:

$$r_n^{tran} = W \log_2 \left(1 + \frac{p_n^{tran} h_n}{\omega_n} \right) \tag{3}$$

The bandwith of the transmittion channel is represented by W, the transmittion power of the task is transmitted by edge device n to an AP is represent by p_n^{tran}. h_n is the channel gain, and ω_n is the noise power [12]. Hence, the time cost for the task is transmitted from the edge device n to an network access point expresent by:

$$T_{n,ap}^{tran} = \frac{\eta_e D}{r_n^{tran}} \tag{4}$$

The energy cost for the edge device n transmits the task to an AP can be expressed by:

$$E_{n,ap}^{tran} = p_n^{tran} T_{n,ap}^{tran} \tag{5}$$

Note, the channel state remain unchanged in the offloading phase [13]. In next subsection, the edge cloud computing will be modeled.

3.4. Edge Cloud Computing Model

When a task is offloaded to an edge cloud. We only consider the processing time on the cloud. In general, we consider the cloud services as prepaid in advance, hence the energy cost for processing the task on cloud is ignored [14]. The time cost for the task is processed on an edge cloud can be expressed by:

$$T_e^{exe} = \frac{\eta_e L}{\gamma_e f_e} \tag{6}$$

Here, f_e represents the CPU/GPU/TPU frequency of the edge cloud (CPU/GPU/TPU cycles per unit time). γ_e is represented the ratio of actual processing capacity. Now, the total time cost of the task generated by edge device is transmited to an edge cloud can be obstained, which can be expressed by:

$$T_c = T_{n,e}^{tran} + T_e^{exe} \tag{7}$$

4. Problem Formulation

4.1. Optimization Problem Formulation

We use λ_T and λ_E, $0 \le \lambda_T \le 1, \lambda_T + \lambda_E = 1$ to represent the weight of time cost and energy cost [15], the total cost for processing a task can be expressed by:

$$\begin{aligned} C_{total} &= \lambda_T (T_n^{exe} + T_c) + \lambda_E (E_n^{exe} + E_{n,ap}^{tran}) \\ &= \lambda_T \left(\frac{(1-\eta_e)L}{\gamma_n f_n} + \frac{\eta_e D}{r_n^{tran}} + \frac{\eta_e L}{\gamma_e f_e} \right) \\ &+ \lambda_E \left(p_n^{exe} T_n^{exe} + p_n^{tran} T_{n,ap}^{tran} \right) \end{aligned} \tag{8}$$

In this paper, our optimization goal is to minimize the total cost, which can be expressed by:

$$\min C_{total} \tag{9}$$

ISSN: 2664-9640

4.2. Problem Solution

We divide the problem into two sub-problems: (1). Determine which edge cloud will be used for collaborative processing the task. (2). How much data needs to be offloaded to the edge cloud. In the first sub-problem, we use the shortest path method to solve it.

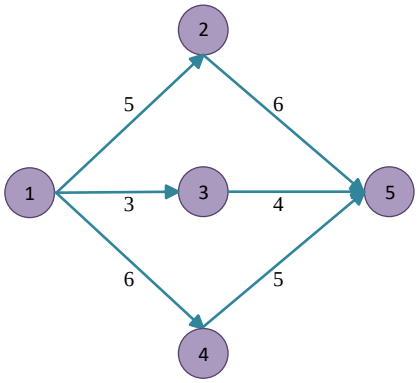


Fig. 3 Example graph of consumption relationship between events

As shown in Fig. 3. The event that we need to reach is represented by each node, and the connection between nodes represents the cost from the current event reach to the next event. Event 1 needs to reach event 5. It Fig. 3, the event 1 first goes to event 3, and then to event 5, the cost is minimal. Therefore, if event 1 reaches event 5, it will pass event 3. Similarly, we first select the edge cloud with the least time required to process the task based on the multiple edge clouds in the environment. In other perspective, it is to select one edge cloud with the highest actual processing frequency.

First, we calculate the actual computing frequency of each edge cloud based on the processing frequency f_e of each edge cloud and the actual computing resource availability rate γ_e. Putting the actual computing frequency $\gamma_e f_e$ of each edge cloud into the set $\mathcal{F}$, and get: $\mathcal{F}=\{\gamma_1 f_1,\ldots,\gamma_E f_E\}$. Then find the edge cloud e with the largest actual calculation frequency:

$$e = argmax\{\mathcal{F}\} \tag{10}$$

In the second sub-problem, we will determine how much data of the task to offload. If a task offloading a part of the data to an edge cloud for processing, if the data processed by the device itself is too large, the processing result of the edge cloud will need to wait a long time. If the task offloading too much data to an edge cloud for processing, then, the user device will need wait a long time after the user device complete the task processing.

Therefore, in the sub-problem 2, how much data need to be offloaded will be considered, and minimizing the waiting time after the edge device complete the task processing or the edge cloud complete the task processing. Hence, when $T_n^{exe}=T_c$, the waiting time difference between the device itself and the edge cloud is the smallest. Therefore, when $T_n^{exe}=T_c$, we can get:

$$\begin{aligned} & T_n^{exe}=T_c \\ \Leftrightarrow & \frac{(1-\eta_e)L}{\gamma_n f_n}=\frac{\eta_e D}{r_n^{tran}}+\frac{\eta_e L}{\gamma_e f_e} \\ \Leftrightarrow & \eta_e=\frac{Lr_n^{tran}\gamma_e f_e}{Lr_n^{tran}\gamma_e f_e+D\gamma_n f_n+L\gamma_n f_n r_n^{tran}} \end{aligned} \tag{11}$$

After analyzing and solving the sub-problems 1 and 2, when the user device generates a task that needs to be processed. First select an edge cloud to be offloaded through the sub-problem 1, and then determine how much data of the task which need to be offloaded to an edge cloud. Next, we will analyze our proposed scheme through numerical experiments.

ISSN: 2664-9640

5. Numerical Experiment Analysis

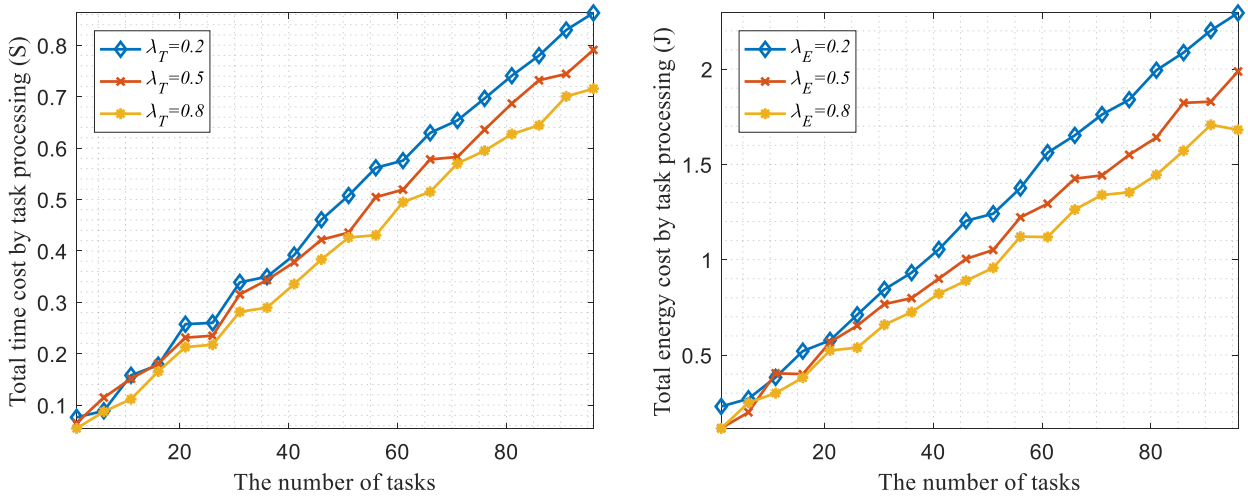
In this section, we first describe the experimental environment and experimental parameters, and then analyze the performance of the offloading scheme proposed in this paper.

5.1. Experiment Parameters Setting

In this experiment, the edge device set 10, the edge clouds set 5; the frequency of edge device is set $f_n = 0.5\text{-}1.5(GHz)$, the frequency of every edge cloud is set $f_e = 100\text{-}150(GHz)$ [15]. The data volume of the task (bit) is set $D = 200\text{-}500(kb)$, the CPU cycles that required to complete the task is set $D = 300\text{-}1000(Megacycles)$ [14]错误!未找到引用源。. The bandwidth of the channel $W = 5(MHz)$, the channel gain power is set $h_n = 10^{-3}$, the channel noise power is set $\omega_n = 10^{-9}$ [15]. The transmission power of edge device is set $p_n^{tran} = 100(mw)$. The time cost weight and the energy cost weight are initially set $\lambda_T = \lambda_E = 0.5$, and then we will set different values to analyze time cost and energy cost.

5.2. Performance Analysis

In Fig. 4, we analyzed the effect on time and energy cost for different time weight λ_T and energy weight λ_E. First, let's first explain the meaning of different time or energy consumption weights. When $\lambda_T = 0.2$, it means the task does not require low delay. However, when $\lambda_T = 0.8$, it means that the task requires a very low time delay. Similarly, when $\lambda_E = 0.2$, it means the task does not require low energy cost. However, when $\lambda_E = 0.8$, it means that the task requires a very low energy cost.

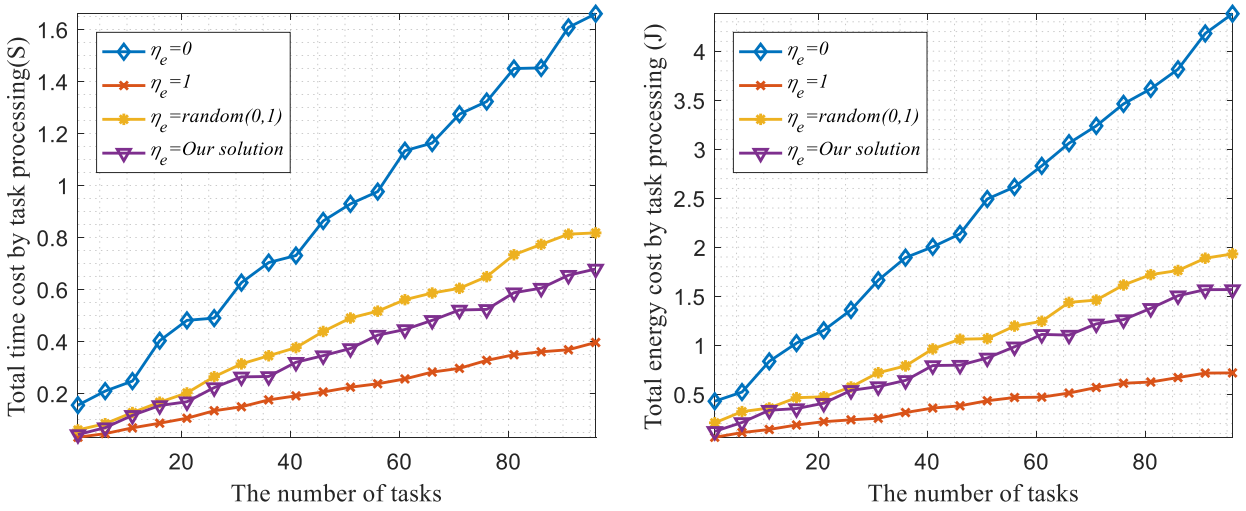


(a) Time cost (b) Energy cost

Fig. 4 The impact of different λ_T and λ_E on time and energy cost

As shown in Fig. 4(a), when processing the same number of tasks, it consumed less time when $\lambda_T = 0.8$ than $\lambda_T = 0.2$. In Fig. 4(b), when processing the same number of tasks, it consumed less energy when $\lambda_E = 0.8$ than $\lambda_E = 0.2$. As shown in Fig. 4, adjusting the time weight λ_T and energy weight λ_E appropriately, we can meet tasks with different requirements for delay and energy consumption.

ISSN: 2664-9640



(a) Time cost (b) Energy cost

Fig. 5 The impact of different η_e on time and energy cost

In Fig. 5, we analyzed the effect on time and energy cost for different η_e. In order to reduce the task processing time cost , when a task generated by an edge device, offloading a part of the data to an edge cloud for processing, thereby forming an edge-cloud collaborative processing mode. The $\eta_e, 0 \le \eta_e \le 1$ represents the portion of the task that is offloaded to edge cloud for processing. When $\eta_e = 0$, it means that the tasks generated by edge device are all processed by the user device itself, in other words, it is commonly referred as local processing. When $\eta_e = 1$, it means that the tasks are all processed by edge clouds. When $\eta_e = random(0,1)$, it means that the tasks are processed by edge devices and edge clouds collaboratively, but the part of data offloaded to the edge cloud is determined randomly. When $\eta_e = our\ solution$, it means that the tasks are processed by edge devices and edge clouds collaboratively, but the part of the task data, which need to be offloaded to the edge cloud is determined according to formula 11 in this paper.

As shown in Fig. 5(a), when processing the same number of tasks, the time cost is the least when $\eta_e = 1$; The time cost is the most when $\eta_e = 0$; The time consumed is between $\eta_e = 1$ and $\eta_e = 0$ when $\eta_e = random(0,1)$ and $\eta_e = our\ solution$, but the time consumed is less when $\eta_e = our\ solution$ than $\eta_e = random(0,1)$. As shown in Fig. 5(b), when processing the same number of tasks, the energy cost is the least when $\eta_e = 1$; The energy cost is the most, when $\eta_e = 0$; The energy cost is between $\eta_e = 1$ and $\eta_e = 0$ when $\eta_e = random(0,1)$ and $\eta_e = our\ solution$, but the energy cost is less when $\eta_e = our\ solution$ than $\eta_e = random(0,1)$.

Now, let's analyze the reason for the above experiment. When $\eta_e = 0$, the tasks generated by user device are processed by the device itself. Since the user device has a relatively low processing capability, it takes the most time. When $\eta_e = 1$, the tasks generated by user device are all offloaded to an edge cloud for processing, as the edge cloud has a strong computing capabilities. So, offloading the tasks to an edge cloud for processing will cost the least time. In the energy cost perspective, if a task is transmitted to an edge cloud for processing, we only need calculate the energy cost of the edge device transmits the task to an edge cloud, so the energy consumption is consumed lowest. When $\eta_e = random(0,1)$, the task is processed by the user device itself and the edge cloud collaboratively, but how much data should be offloaded to

ISSN: 2664-9640

an edge cloud for processing is determined randomly, which results in a certain time difference between the completion time of the user device and the edge cloud, which leads to the time consumption and the energy cost are more than our proposed scheme.

6. Summary

We mainly considered the tasks generated by data partitioned oriented applications (DPOAs). These tasks do not have much internal dependencies and can be arbitrarily divided, such as data compression, virus scanning, etc. We use user device and edge cloud to build a unified edge-cloud collaborative task processing network, and coordinate tasks through the user device itself and the edge cloud. We use the shortest path method to determine which edge cloud to offload the task will cost the least time firstly. After that, we use model derivation to calculate how much data is offloaded to the edge cloud, which can minimize the waiting time after the user device and the edge device have completed their data respective, thereby further reducing the task processing delay. Finally, the performance of the edge-cloud task processed model was analyzed.

Thanks

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61602155.

References

[1] Narkhede B E, Raut R D, Narwane V S, et al. Cloud computing in healthcare - a vision, challenges and future directions[J]. International Journal of Business Information Systems. 2020, 34(1): 1-39.
[2] Ferrer A J, Marquès J M, Jorba J. Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad hoc, and Edge Computing[J]. ACM Computing Surveys. 2019, 51(6): 111:1-111:36.
[3] Josilo S, Dán G. Selfish. Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks[J]. IEEE Transactions on Mobile Computing. 2019, 18(1): 207-220.
[4] Kim Y, Lee H, Chong S. Mobile Computation Offloading for Application Throughput Fairness and Energy Efficiency[J]. IEEE Transactions on Wireless Communications. 2019, 18(1): 3-19.
[5] Guo S, Liu J, Yang Y, et al. Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing[J]. IEEE Transactions on Mobile Computing. 2019, 18(2): 319-333.
[6] Chen X. Decentralized Computation Offloading Game for Mobile Cloud Computing[J]. IEEE Transactions on Parallel and Distributed Systems. 2015, 26(4): 974-983.
[7] Roya G, Frédéric L M, Julien P, et al. Automated application offloading through ant-inspired decision-making[C]. 13th International Conference on New Technologies for Distributed Systems, {NOTERE} 2016, Paris, France, July 18, 2016. 2016: 1-6.
[8] Hillol D, Giacomo G, Antonio C, et al. Collaborative Offloading for Distributed Mobile-Cloud Apps[C]. 6th {IEEE} International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2018, Bamberg, Germany, March 26-29, 2018. 2018: 87-94.
[9] Sladana J, György D. Wireless and Computing Resource Allocation for Selfish Computation Offloading in Edge Computing[C]. 2019 {IEEE} Conference on Computer Communications, {INFOCOM} 2019, Paris, France, April 29 - May 2, 2019. 2019: 2467-2475.
[10] Zhou S, Jadoon W. The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment[J]. Computer Networks. 2020, 178: 107334.
[11] Wang C, Liang C, Yu F, et al. Computation Offloading and Resource Allocation in Wireless Cellular Networks With Mobile Edge Computing[J]. IEEE Transactions on Wireless Communications. 2017, 16(8): 4924-4938.
[12] Lyu X, Tian H, Sengul C, et al. Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds[J]. IEEE Transactions on Vehicular Technology. 2017, 66(4): 3435-3447.

ISSN: 2664-9640

[13] Cao H, Cai J. Distributed Multiuser Computation Offloading for Cloudlet-Based Mobile Cloud Computing: A Game-Theoretic Machine Learning Approach[J]. IEEE Transactions on Vehicular Technology. 2018, 67(1): 752-764.
[14] Meng X, Wang W, Wang Y, et al. Closed-Form Delay-Optimal Computation Offloading in Mobile Edge Computing Systems[J]. IEEE Transactions on Wireless Communications. 2019, 18(10): 4653-4667.
[15] Wang Y, Sheng M, Wang X, et al. Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling[J]. IEEE Transactions on Communications. 2016, 64(10): 4268-4282.