Delay-Aware and Profit-Maximizing Task Migration for the Cloudlet Federation

Hengzhou Ye, Junhao Guo, Xinxiao Li*
Guangxi Key Laboratory of Embedded Technology and Intelligent Systems
Guilin University of Technology, Guilin, China

***Abstract*—As the result of Open Edge Computing (OEC) project, cloudlet embodies the middle layer of edge computing architecture. Due to the close deployment to the user side, responding to user requests through cloudlet can reduce delay, improve security, and reduce bandwidth occupancy. In order to improve the quality of user experience, more and more cloudlets need to be deployed, which makes the deployment and management costs of Clouldlet service Providers (CLP) significantly increased. Therefore, the cloudlet federation appears as a new paradigm that can reduce the cost of cloudlet deployment and management by sharing cloudlet resources among CLPs. Facing the cloudlet federation scenario, more attention still needs to be paid to the heterogeneity of resource prices, the balance of benefits among CLPs, and the more complex delay computation when exploring task migration strategies. For delay-sensitive and delay-tolerance tasks, a delay-aware and profit-maximizing task migration strategy is proposed considering the relationship between the delay and the quotation of different tasks, as well as the dynamic pricing mechanism when resources are shared among CLPs. Experimental results show that the proposed algorithm outperforms the baseline algorithm in terms of revenue and response delay.**

***Keywords*—Cloudlet federation; task migration; delay-aware; dynamic pricing; profit-maximizing; edge computing**

I. INTRODUCTION

With the development of intelligent Mobile Devices (MD), a large number of computation-intensive and delay-sensitive mobile applications keep emerging [1]. To address the resource constraints of mobile devices, tasks are offloaded to remote cloud centers in traditional cloud computing [2][3], but it is followed by the problem of high response delay [4]. This makes it difficult for requirements of delay-sensitive task to be satisfied [5].

To overcome these problems, Mobile Edge Computing (MEC) [6] is proposed and regarded as a promising technology by developing servers at the edge of networks. The proximity of the MEC to the user allows the user to access and use resources with lower response latency. Cloudlets are small cloud at the edge of network and typically deployed in a distributed manner to provide storage and computing resource to MD.

Compared to existing cloud computing, MEC still suffers from constrained resources. A Cloudlet service Provider (CLP) has to deploy more and more proprietary edge devices to meet the needs of its users, resulting in increasing costs to deploy and manage edge devices. Therefore, [7] proposes the edge federation architecture, which improves the flexibility of resource utilization and reduces the deployment and management costs by seamlessly integrating the edge resources of each CLP. When cloudlets act as edge devices, a cloudlet federation architecture is formed.

Since MD is not uniformly distributed in edge computing systems, some clouds are heavily loaded while others are lightly loaded. Task migration is the principal method used to resolve load imbalance between cloudlets [8-11]. Although many studies have been conducted on cloudlet-based load migration strategies, they have one or more shortcomings: they do not take into account the heterogeneity of resource prices, authentication delays between cloudlet resources of different CLPs, or the impact of service delays on task quotes.

This paper focuses on the task migration problem in the cloudlet federation scenario. Two types of user requests are considered: delay-sensitive and delay-tolerant, and their quotation delay curves are designed respectively. Considering the heterogeneity of resource prices among CLPs, a delay-aware and profit-maximizing task migration strategy is proposed. Our main contributions are as follows:

- For delay-sensitive and delay-tolerant tasks, a delay model and a price model are constructed by taking into account the differences in resource prices of different CLPs.
- A distributed Delay-aware and Profit-maximizing Task Migration (DPTM) algorithm is designed to schedule tasks, which tries to minimize latency and maximize profit.
- A series of experiments are conducted to verify the necessity of considering the cloudlet federation and the advantages of our method in terms of revenue and response delay.

The rest of this article is organized as follows. Section II demonstrates the related work on the task migration in different scenarios. Section III introduces the system model. In Section IV, a task scheduling algorithm considering both profit and delay is proposed. In Section V, we conduct the extensive experiments to evaluate the proposed method and gives the discussion. Finally, Section VI concludes this paper and gives the future works.

*Corresponding Author.

II. Related Work

We review the related works about task migration and schedule in the multi-cloudlet scenario and the edge federation scenario respectively.

A. Task Migration in Multi-cloudlets Scenario

The existing works about task migration in the multi-cloudlets scenarios focus on minimizing delay and energy consumption. The authors in [12] propose an application-aware cloudlet adaption and VM selection framework has been devised for balancing the load in a multi-cloudlet environment. The authors in [13] propose a novel Multi-layer Latency Aware Workload Assignment Strategy (MLAWAS) to allocate the workload of E-Transport applications into optimal computing nodes. In [14] user mobility and task deadline are taken into account when task migration is optimized. Three variants of this problem are analyzed, and a group migration algorithm with known user trajectories is designed. In [15], a heuristic Task Migration Computing Offloading (TMCO) scheme is proposed for the challenges brought by complex network environment and end-user mobility, which can dynamically select the appropriate location to offload tasks for mobile users within the deadline. In [16], for collaborative vehicle edge computing group environment, a computational task migration problem is defined to balance the load and minimize the migration cost, and reinforcement learning algorithm is adopted to solve this problem.

The above researches are based on multi-cloudlet scenarios to optimize the delay and energy consumption of task migrating. However, the traditional multi-cloudlet scenario has the problem of limited resources, and a smaller number of resources are available for task migration. Different from the multi-cloudlet scenario, in the cloudlet federation, the cloudlet resource prices of different CLPs may be different, there are more options for task migration, and the delay computation is more complex.

B. Task Migration in Edge Federation Scenario

Some studies have explored task migration strategies in the context of edge federation or cloudlet federation. The authors in [17] design a task migration strategy for multiple edge servers in mobile networks to minimize the overall service time and develop an intelligent task migration scheme using deep reinforcement learning and Q-learning technology. In [18], the authors propose a latency minimization model to provide higher efficient service provisioning in horizontal edge federation and propose a two-phase iterative approach, which alternately determines optimal task dispatching and computation resource allocation. In order to simultaneously meet the SLA requirements of IoT (Internet of Thing) devices and edge service providers, the authors in [19] design an intelligent request service provisioning system based on reinforcement learning as part of a smart edge orchestrator in the edge federation. Considering the delay and capacity constraints, [20] proposes an optimization model to minimize the total energy cost and the energy efficient offloading ratio of edge nodes.

The above studies either ignore the role of the remote cloud, or do not consider the heterogeneity of resource prices, or do not consider the impact of task delay on user quotation. In our work, tasks can be migrated to the CLP's own cloudlet, the cloudlet of federate CLP, or the remote cloud. We consider two different types of tasks, time-sensitive and time-tolerance, and consider that their delays have different effects on user quotation. We attempt to optimize the profit and delay of the whole federation.

III. System Model

In this section, we keep our focus on the cloudlet federation framework, communication and computation models.

A. Cloudlet Federation Framework

Compared with the cloudlet, the cloud usually has sufficient resources and is quite far away from the user. For simplicity, the cloud platform owned by each CLP is not distinguished. Therefore, as shown in Fig. 1, we consider that a cloudlet federation contains a cloud and several CLPs. Each CLP has its own cloudlet servers, and each has a set of users that need to be served. The resource of a cloudlet server is provided in the form of a virtual machine (VM) instance. Let's assume that a VM instance performs only one task at a time.

We assume each CLP has a server as the Federated Cloudlet Manager (FCLM) to provide service and resource interactions. The status of the cloudlet servers is collected by the FCLM, such as available capacity, resource utilization, and cost. Meanwhile, CLPs can communicate with each other through FCLMs and exchange their collected information. When a CLP receives a task from a terminal user, it decides where to offload the task. This could be its own cloudlet, one of the federated cloudlets, or the cloud.

B. Task Requests from Terminal Users

Similar to works in mobile cloud computing [21] and mobile edge computing [22], the quasi-static scenario is considered, where the mobile users remain unchanged during migration. To simplify, we use task or request instead of task request without causing confusion.

Two types of task requests are considered: delay tolerance (DT) and delay sensitive (DS). For a DS task, CLP should minimize its delay while a certain amount of revenue is guaranteed. For a DT task, the priority is profit maximization.

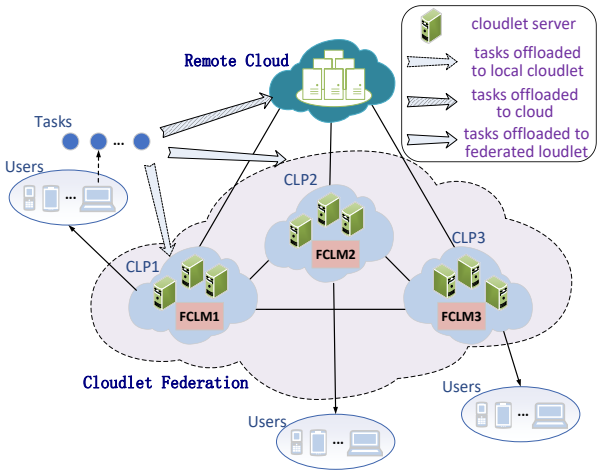


Fig. 1. Cloudlet Federation Architecture.

Regardless of the type of task, we believe that the response delay will affect the user's willingness to pay. In general, the shorter the delay, the higher the user is willing to pay.

A seven-tuple $A = <d, req^{ins}, \alpha, t^{ept}, t^{max}, P^{max}, P^{low}>$ is used to describe a task where, d denotes the size of input data, req^{ins} denotes the Million Instructions (MIs) required. α is a 0-1 variable that describes the type of tasks. When α is 0, the type of the task is DT, and when α is 1, the type of the task is DS. We assume a task has an expected delay t^{ept} and a tolerable maximum latency t^{max}. P^{low} and P^{max} represent the minimum and maximum price a user is willing to pay for a task, respectively.

For a DT task, if it can be completed within the expected time, marked as t^{ept}, the user will be satisfied and willing to pay in full, marked as P^{max}. If t^{ept} is exceeded, but the deadline, marked as t^{max}, is not exceeded, the willing fee of the user will gradually decrease, and eventually reach a balance value, marked as P^{low}, between the CLP and the user. Thus, for a DT task, the relationship between the delay t and the user's quotation $Q(t)$ can be shown in (1).

$$Q(t) = \begin{cases} P^{max}, & t < T^{ept} \\ k_{dt} \cdot (T^{max} - t) + P^{low}, & T^{ept} \le t \le T^{max} \\ P^{low}, & t > T^{max} \end{cases} \quad (1)$$

Where, k_{dt} is determined by (2).

$$k_{dt} = \frac{P^{max} - P^{min}}{T^{max} - T^{ept}} \quad (2)$$

For DS tasks, users will be willing to pay in full if they can be completed within the expected time. If the delay exceeds the user's expectation, the user's willingness to pay will decrease sharply. If the delay exceeds the deadline, the user will be unwilling to pay, or only willing to pay at a very low price. We chose the latter, meaning that when the deadline is exceeded, the user will be willing to offer only a very low price, which is usually close to or even lower than the CLP cost. We design (3) to describe the relationship between the delay t of a DS task and the user's quotation Q(t).

$$Q(t) = \begin{cases} P^{max}, & t < T^{ept} \\ P^{max}((1 - \alpha(t - T^{ept}))^{\beta}, & T^{ept} \le t \le T^{max} \\ P^{low}, & t > T^{max} \end{cases} \quad (3)$$

Where, α and β are adjustable parameters, and their values should ensure that the curve described by (3) is continuous on the interval $[T^{ept}, T^{max}]$. α indicates the delay sensitivity of the user. The larger the α, the higher is the sensitivity. β reflects the descent gradient of user willingness. The larger the β, the larger is the gradient.

Fig. 2(a) and Fig. 2(b) depict the curves of delay versus quotation for DT and DS tasks, respectively.

C. Delay Model

When the FCLM of a CLP receives a task, the FCLM may schedule it to the local cloudlet, or to the federated cloudlet, or to the cloud. In general, the amount of data output after the task execution is much smaller than the amount of data input before the task execution. Similar to [23] and [24], the transmission time required for the output information of a task to be returned to the user is not considered. Meanwhile, the queuing delay is negligible when the federated resources are sufficient to handle user tasks.

1) Local cloudlet: When a task is scheduled to the local cloudlet, its delay can be divided into two parts. The first part includes transmission delay and propagation delay, denoted as t_{cl}^1, which can be determined by (4).

$$t_{cl}^1 = \frac{d}{R} + \frac{D_{cl}}{S^p} \quad (4)$$

Where d is the amount of data to be transmitted, R is the data transmission rate, D_{cl} is the distance of the selected local cloudlet from the user, and S^P is the propagation rate. Assume that the value of S^P is the same for all channels and the value of R is the same for all devices.

The second part is the time required for the task execution on the selected cloudlet, denoted as t_{cl}^2, which is determined by (5).

$$t_{cl}^2 = \frac{req^{ins}}{CPU_{proc} \cdot N_{cpu}} \quad (5)$$

Where req^{ins} is the MIs required to execute the task, CPU_{proc} is the Million Instructions per Second (MIPs) for the CPU provided by the selected cloudlet, and N_{cpu} is the number of CPUs configured by the selected cloudlet.

Thus, the total delay, denoted as t_{cl}^{total}, for a task scheduled to a local cloudlet can be calculated by (6).

(6)

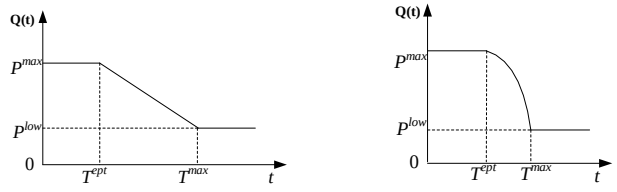
2) Remote cloud. When a task is scheduled to the remote cloud, its delay calculation is very similar to that when it is scheduled to a local cloudlet. However, since the user is usually far away from the cloud, the delay of the task will be huge.

The total delay, denoted as t_c^{total}, for a task scheduled to the remote cloud can be calculated by (6).

$$t_c^{total} = t_c^1 + t_c^2 = \frac{d}{R} + \frac{D_c}{S^p} + \frac{req^{ins}}{CPU_{proc} \cdot N_{cpu}} \quad (7)$$

Where, D_c is the distance of the cloud from the user, CPU_{proc} is the MIPs for the CPU provided by the cloud, and N_{cpu} is the number of CPUs configured by the cloud.

3) Federated cloudlet. If a task is assigned to the federated cloudlet, its delay includes the following parts: authentication delay between alliances, task offloading delay from user to FCLM, task offloading delay from FCLM to the selected cloudlet, and task execution delay.



(a) For DT task. (b) For DS task.

Fig. 2. Curves of Delay Versus Quotation.

When a task is transferred from the FCLM receiving it to the FCLM of a federation, some authentication information often needs to be exchanged between FCLMs, so as to introduce authentication delay, marked as t_{fed}^1, which is considered to be a constant.

To avoid CLP user information being detected by the federated CLP, tasks are delivered first to the FCLM that receives them and then to the selected Cloudlet. The delay of offloading the task from the user to the FCLM, denoted as t_{fed}^2, can be calculated by (8).

$$t_{fed}^2 = \frac{d}{R} + \frac{D_1}{S^p} \quad (8)$$

Where D_1 represents the distance between the user and the FCLM associated with it.

The delay of offloading tasks from FCLM to the selected cloudlet is denoted as t_{fed}^3, which can be calculated by (9).

$$t_{fed}^3 = \frac{d}{R} + \frac{D_2}{S^p} \quad (9)$$

Here, D_2 represents the distance between the FCLM associated with the user and the selected cloudlet.

The execution delay of the task, denoted as t_{fed}^4, can be determined by (10).

$$t_{fed}^4 = \frac{req^{ins}}{CPU_{proc} \cdot N_{cput}} \quad (10)$$

Where, CPU_{proc} is the MIPs for the CPU provided by the selected cloudlet, and N_{cpu} is the number of CPUs configured by the selected cloudlet. CPU_{proc} and N_{cpu} in (5), (7) and (10) are not symbolically distinguished. In fact, they correspond to different computing resources and take different values.

Thus, the total delay, denoted as t_{fed}^{total}, can be calculated by (11).

$$t_{fed}^{total} = t_{fed}^1 + t_{fed}^2 + t_{fed}^3 + t_{fed}^4 \quad (11)$$

D. CLP Resource Pricing Model

When a CLP needs to rent the cloudlet resources of another CLP, the leasing price of the resources needs to be determined. The consistent pricing strategy is adopted, and the leasing price of resources is considered to be related to the user's quotation, resource cost and resource utilization of CLP.

Assuming that the user quotation of task *A* is $Quote_A$, the cost of resources needed to execute A is $Cost_A$, and the total cloudlet resources and residual resources owned by some CLP are $Total_{CLP}$ and Res_{CLP} respectively, then the cost of CLP renting the resources required by task A, denoted as $Rent_A^{CLP}$, can be determined by (12).

$$Rent_A^{CLP} = Cost^A + \frac{Total_{CLP} - Res_{CLP}}{Total_{CLP}} (Quote_A - Cost^A) \quad (12)$$

The intuition of (12) is as follows: the rental price of resources should be higher than that of the cost. The net profit between rental price and cost should be positively correlated with the profit of the lease and the resource utilization of the lessor accordingly. The willingness of renting resource is growth with the user's quoted price, and as the increasing of the resource utilization, the tenant willingness of renting is decreasing.

When a task is executed on the cloud, its profit is the difference between the user's quote and the cost of the cloud resource. When executed on the local cloudlet, its profit is the difference between the user's quote and the local cloudlet resource cost. When executed on the federated cloudlet, its profit is the difference between the user's quote and the rental resource price.

IV. Proposed Algorithm

Based on the distributed processing idea and greedy strategy, for each CLP, when its FCLM receives a task, we design a Delay-aware and Profit-maximizing Task Migration (DPTM) algorithm to schedule the task. DPTM tries to maximize the profit of CLP while guaranteeing the user delay according to the task type. It first determines the type of tasks. For DS tasks, a scheduling algorithm called Task Migration for DS (TMDS) is proposed. For the DT task, an algorithm called Task Migration for DT (TMDT) is proposed.

A. TMDS Algorithm

For DS task, the goal is to minimize delay on the basis of ensuring profit. For this purpose, locally arrived DS tasks are divided into High Delay Sensitive (HDS) and Low Delay Sensitive (LDS) tasks according to their delay requirements. For a HDS task, local scheduling is preferred. If it cannot be scheduled locally, it is processed as an LDS task. For a LDS task, if the cloud has the highest profit and can guarantee the delay, the cloud is regarded as the best scheduling. Otherwise, on the premise of ensuring profit, the cloudlet with the minimum delay should be sought.

The TMDS algorithm for scheduling DS tasks is shown in Algorithm 1. In Algorithm 1, Avg_k records the average t^{ept} of the last k DS tasks, $t^{ept}(A)$ and $t^{max}(A)$ represent the t^{ept} and t^{max} of task *A* respectively, t_{cl}^{total} (*VM*) represents the delay of *A* when executed on the *VM* of a cloudlet, p_{cl} (*VM*) represents the profit when *A* is executed on the *VM* of a cloudlet, and p_c (*A*) represents the profit when *A* is executed on the cloud. A task is identified as HDS when its t^{ept} is less than Avg_k. In this case, firstly, the VM that can execute *A* and delay less than $t^{max}(A)$ are searched on the local cloudlet. Otherwise, find all VMs that can execute *A* with latency less than $t^{max}(A)$ and schedule *A* to the one with the least latency. If no suitable VM is found after the above two steps, *A* will be treated as an LDS type.

For a task *A* identified as a LDS, the latency and profit of *A* when executed on the cloud are calculated, denoted as $t_c^{total}(A)$ and p_c (*A*), respectively. If $t_c^{total}(A)$ is less than $t^{max}(A)$ and p_c (*A*) is greater than 0, the VM with executable *A*, delay less than $t^{max}(A)$, and profit greater than p_c (*A*) is sought on the cloudlets of all CLPs. If one or more VMs can be found, the VM with the minimum latency is selected for *A*. Otherwise, *A* is scheduled to the cloud.

If $t_c^{total}(A)$ is not less than $t^{max}(A)$ or p_c (*A*) is not greater than 0, the VM that can execute A and the profit is greater than 0 is searched on the cloudlets of all CLPs, and the one with

minimum delay is the best for *A*. Otherwise, *A* will be abandoned.

In Algorithm 1, a maximum of three scheduling attempts is required on each VM for a task. If *n* is used to represent the number of VMs, the time complexity of algorithm 1 is *O*(*n*), that is, algorithm 1 has linear time complexity.

Algorithm 1 TMDS

Input: task A with type of DS
Output: VM is arranged for A
Calculate Avg_k;
Set VM_T<VM, Time> = ∅;
If ($t^{ept}(A)$ < Avg_k){
  For each VM in local cloudlet {
    Calculate t_{cl}^{total} (VM) using (6);
    If (t_{cl}^{total} (VM) < $t^{max}(A)$)
      add <VM, t_{cl}^{total} (VM)> to VM_T;
  }
  if (VM_T is not null)
    return VM with minimum delay in VM_T;
}
Calculate $t_c^{total}(A)$ using (7) and p_c (A);
If ($t_c^{total}(A)$ < $t^{max}(A)$&& p_c (A) > 0){
  For each VM in all cloudlets {
    Calculate t_{cl}^{total} (VM) using (6) or (12);
    If (t_{cl}^{total} (VM) < $t^{max}(A)$ && p_{cl} (VM) > p_c (A))
      add <VM, p_{cl} (VM)> to VM_T;
  }
  If (VM_T is not null)
    return VM with minimum delay in VM_T;
  else
    return VM in cloud;
}
For each VM in all cloudlets {
  Calculate t_{cl}^{total} (VM) using (6) or (12);
  If (p_{cl} (VM) > 0)
    add <VM, t_{cl}^{total} (VM)> to VM_T;
}
If (VM_T is not null)
  return VM with minimum delay in VM_T;
else
  return null;

B. TMDT Algorithm

For the DT task, the goal is to minimize the delay while pursuing the maximum profit. Considering that the delay of a task executed on the local cloudlet is usually less than that on the federated cloudlet or the cloud, in order to save the local cloudlet as much as possible for scheduling DS tasks, DT tasks can only be scheduled to the federated cloudlets with the lower quotation or the cloud. The criterion is profit maximization.

The TMDT algorithm is used to schedule tasks of type DT, as shown in Algorithm 2. For a task *A*, the profit executed on the cloud, denoted as p_c (*A*), is firstly calculated. If p_c (*A*) is greater than 0, the VM that can execute *A* and have higher profit than p_c (*A*) is found on all the federated cloudlets, and *A* is scheduled to the one with the maximum profit. If no such VM can be found, *A* is scheduled to the cloud. If p_c (*A*) is not greater than 0, the VM that can execute *A* and have higher profit than 0 is found on all the federated cloudlets. If such VMs exist, the one that can make the maximum profit is the best choice. Otherwise, scheduling A is abandoned.

Algorithm 2 traverses all VMs at most two times, so its time complexity is *O*(*n*).

Algorithm 2 TMDT

Input: task A with type of DT
Output: VM is arranged for A
Set VM_P<VM, Profit> = ∅;
Calculate p_c (A);
If (p_c (A) > 0){
  For each VM in federated cloudlets
    If (p_{cl} (VM) > p_c (A))
      add <VM, p_{cl} (VM)> to VM_P;
  If (VM_P is not null)
    Return VM with maximum profit in VM_P;
  Else
    Return VM in cloud;
}
Else {
  For each VM in federated cloudlets
    If (p_{cl} (VM) > 0)
      add <VM, p_{cl} (VM)> to VM_P;
  If (VM_P is not null)
    Return VM with maximum profit in VM_P;
  Else
    Return null;
}

V. Results and Discussion

A. Simulation Parameters Setting

We considered the cloudlet federation scenario composed by three individual CLPs and a remote cloud. Each CLP deploys four servers within its managed area as its cloudlet. Each cloudlet server has 20-30 VMs, while the cloud has 2000 VMs. Referring to Amazon, two types of VMs are supported. One with four Virtual CPUs (vCPU) and 16GB of memory, priced at $0.424, serves DT tasks; the other has two vCPUs and 8GB of memory, priced at $0.212, serves DS tasks. The processing speed of a vCPU is 20000MIPS. The task type is randomly selected. The influence of distance between servers in a cloudlet is not considered. Other parameters are set as shown in Table I.

B. Comparison with the Baseline Algorithm

In order to verify the effect of the proposed method in terms of time delay and profit, the proposed method is compared and analyzed with DATM and CATM proposed in literature [25]. DATM is a delay-aware task migration algorithm that migrates tasks to the cloudlet that minimizes the latency of task execution. And CATM is a cost-aware task migration algorithm that migrates tasks to the cloudlet that minimizes the cost of task execution. Considering the obvious difference in delay expectation between DS and DT tasks, statistics are carried out for DS and DT tasks respectively when comparing the delay.

Fig. 3 and Fig. 4 compare the average delay of DPTM, DATM and CATM with the number of tasks. Here, Fig. 3 counts all DS tasks, while Fig. 4 counts all DT tasks.

TABLE I. PARAMETER SETTINGS

Parameter	Definition	Value
D_{cl}	Distance of user to local cloudlet	10-50m
D_c	Distance of user to the cloud	1-10km
D_{fclm}	Distance between FCLMs	200-500m
S^p	Propagation speed	3×10^8m/s
R	Data transmission rate	3Mbps
t^{ept}	Expected delay of DS task	50-100ms
	Expected delay of DT task	200-400ms
req^{ins}	DS task size	1000-2000MI
	DT task size	500-1000MI]
P^{low}	Minimum quotation for DS task	$2
	Minimum quotation for DT task	$1
P^{max}	Maximum quotation for DS task	$3
	Maximum quotation for DT task	$2
d	Size of input data for task	0.1-2MB
k_{dt}	The parameter in (1)	0.6
β	The parameters in (3)	3

Fig. 3 shows that for DS tasks, DPTM achieves better average delay than DATM, while DATM outperforms CATM. Although DATM is optimized for delay, it does not distinguish between DS and DT tasks. DPTM tries to reserve local cloudlet resources for DS tasks. Therefore, for DS tasks, DPTM obtains a better average delay than DATM.

For the DT task, the results in Fig. 4 show that DPTM achieves better average latency than CATM, but not as good as DATM. DPTM gives priority to DS tasks in terms of latency, thus, for DT tasks, it is not surprising that its average latency is lower than DATM. For CATP, it aims at minimizing task migration cost and does not focus on delay. Although DPTM mainly optimizes the profit for DT tasks, the profit is negatively correlated with the delay, so it optimizes the delay to a certain extent.

Fig. 5 compares the profit with the number of tasks. From Fig. 5, it can be observed that DPTM achieves the best profit, followed by CATM and DATM. The quotation of the task is related to the time delay, while the lease price is related to the resource utilization and the quotation. Therefore, the profit of the task is related to both the time delay and the resource utilization. For CATM, although it aims to minimize the migration cost, the profit decreases due to the high average delay. For DATM, the latency is relatively small, which can increase profits. However, in order to pursue low latency, tasks are preferentially scheduled to the local cloudlet, resulting in high resource utilization of the cloudlet, which pushes up the task quotation and reduces the profit.

In general, DPTM achieves better profit because it treats DT and DS tasks differently and differentiates the relationship between the quotation and the delay of these two types of tasks. For DS tasks, DPTM also achieves better average delay.

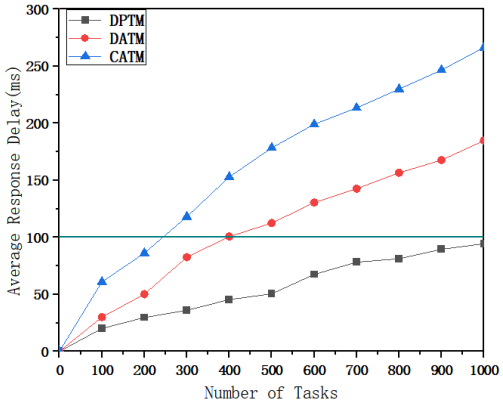


Fig. 3. Average Delay of DS Tasks.

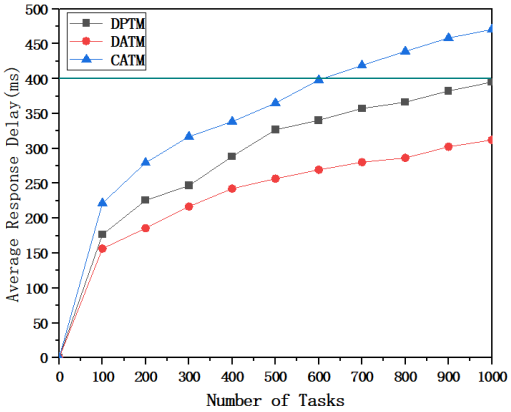


Fig. 4. Average Delay of DT Tasks.

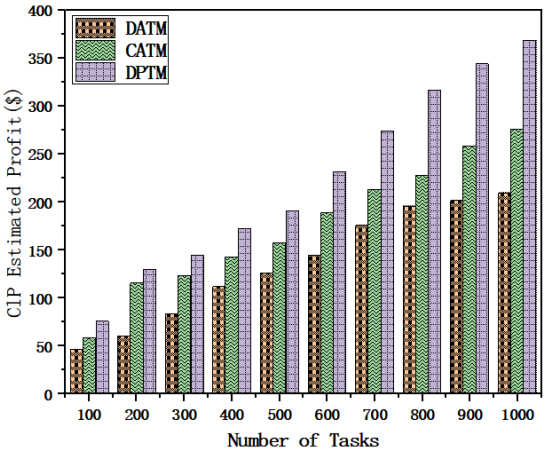


Fig. 5. Average Profit of Tasks.

C. The Necessity of the Federation

To verify the necessity of considering federation, the federation scenario is compared with the non-federation scenario where cloudlet resources cannot be shared between different CLPs.

Fig. 6 compares the average latency of the two scenarios. Where, Fed-DPTM-DS and Fed-DPTM-DT represent the average delay of all DS and DT tasks in the federated scenario, respectively. NonFed-DPTM-DS and NonFed-DPTM-DT indicate the average latency of all DS and DT tasks in the non-

federated scenario, respectively. From Fig. 6, it can be found that the average delay of DS and DT tasks are both significantly reduced when considering federation. This is because in the federation scenario, different CLPs share cloudlet resources, which can reduce the impact of the difference in the number of tasks and the uneven distribution of cloudlet resources among CLPs. For DT tasks, the decrease effect of average delay is more obvious, because DPTM preferentially migrates DT tasks to the federated cloudlet. However, without considering federation, tasks tend to be migrated only to the cloud, resulting in higher latency.

As can be seen from Fig. 7, considering federation can improve profits, and the more tasks there are, the more obvious the improvement effect will be. On the one hand, DPTH is specifically designed for the federation scenario. On the other hand, CLP federation supports cloudlet resources sharing, which reduces the average delay and reduces the probability of tasks being scheduled to the cloud or rejected.

Fig. 8 compares the ratio of the number of rejected tasks to the total number of tasks in the two scenarios. As can be seen from Fig. 8, no matter which scenario, with the increase of the number of tasks, the situation that the task is rejected for scheduling occurs, and the proportion of rejected increases with the increase of the number of tasks. However, compared with the non-federation scenario, the proportion of tasks rejected for scheduling is much lower in the federation scenario.

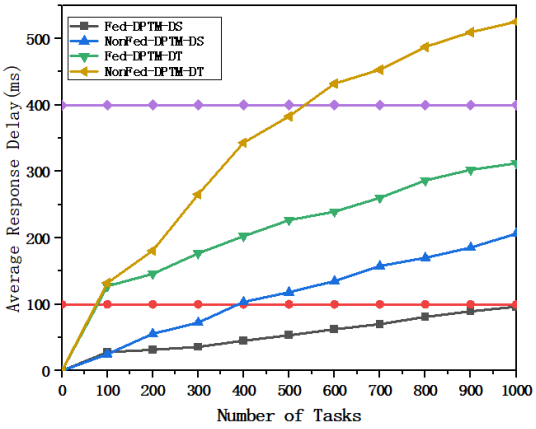


Fig. 6. Profits for the Two Scenarios.

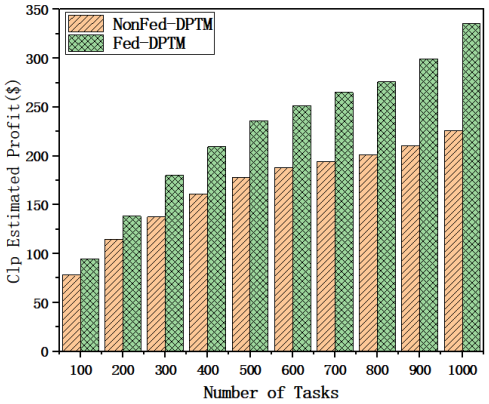


Fig. 7. Average Profit of Tasks.

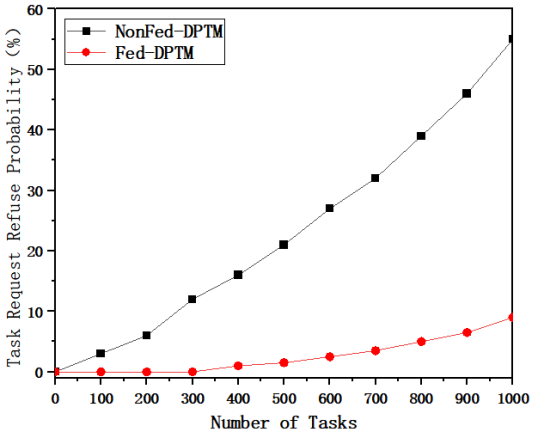


Fig. 8. Refuse Probability of Tasks.

In general, considering the federation scenario, the average delay can be reduced, profits can be increased, and the probability of task rejection can be reduced.

D. Impact of Authentication Delay

During resource sharing, some authentication information may need to be exchanged between CLPs, which may lead to authentication delay. Fig. 9 and Fig. 10 analyze the influence of authentication delay on the average delay of DS and DT tasks, respectively. For DS tasks, the results in Fig. 9 show that when the number of tasks is small (no more than 100), the authentication delay has almost no effect on the average delay, because almost all DS tasks are scheduled to the local cloudlet. However, when the number of tasks is large, the average delay of DS tasks increases with the increase of authentication delay, because more and more tasks have to be scheduled to the federated cloudlet.

As shown in Fig. 10, the average delay of DT tasks increases with the increase of authentication delay, even if the number of tasks is small, because DT tasks are preferentially deployed on the federated cloudlet. Different from DS tasks, authentication delay is not sensitive to the number of DT tasks.

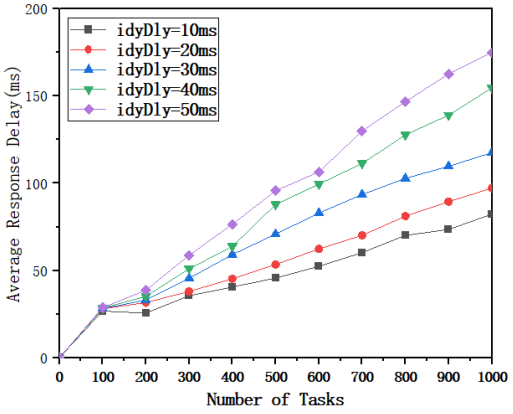


Fig. 9. Impact of Authentication Delay to DS Tasks' Delay.

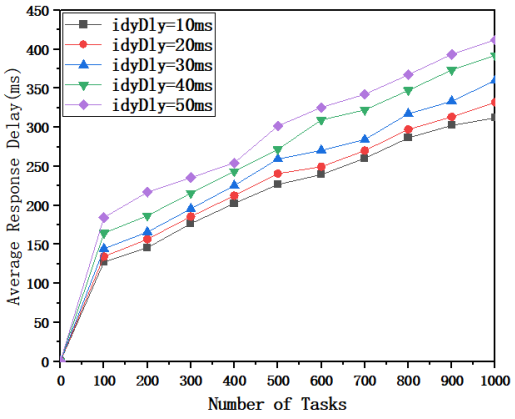


Fig. 10. Impact of Authentication Delay to DT Tasks' Delay.

Fig. 11 analyzes the impact of authentication delay on profit. Fig. 11 shows that the profit decreases as the authentication delay increases. And the longer the delay, the faster the profit decreases. This is because for a DT task, when the delay exceeds the threshold, the quotation is linearly reduced. And for DS tasks, the quotation is exponentially reduced when the delay exceeds the threshold.

In a word, the authentication delay can affect the task delay and profit, so the authentication delay should be minimized.

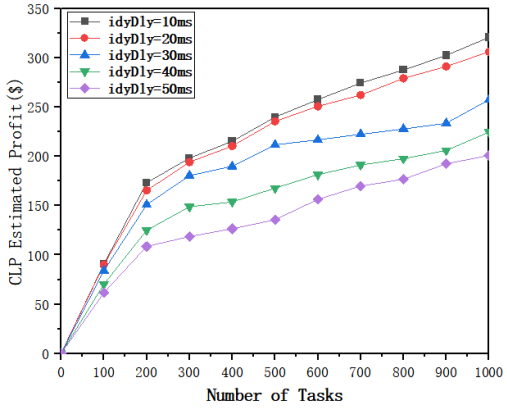


Fig. 11. Impact of Authentication Delay to Profit.

E. Discussion

CLP can share resources with each other in the form of alliances, to solve the problem of resource limitation and the high cost of resource expansion and benefit from the heterogeneity of resource prices provided by different CLPS in alliances. CLP has more options for task migration, resulting in higher benefits. As far as we know, most of the existing studies consider horizontal edge federation or cloud federation, and there are few researches on task migration for the cloudlet federation. Meanwhile, few types of research on task migration for the cloudlet federation combine the powerful computing power of cloud center. For this reason, because of the shortcomings of existing research, this paper considers a cloudlet federation scenario integrating CLP and cloud center computing resources. At the same time, the cost of CLP and task processing delay are considered in the process of task migration, and the authentication delay is considered in the delay calculation.

As shown in Fig. 5 and Fig. 6, the DPTM algorithm proposed in this study balances the needs of users and CLP. The algorithm outperforms the baseline approach in terms of latency and profits of CLP and helps to satisfy more requests. Initially, DPTM divides tasks according to their delay sensitivity. Next, DPTM takes the delay minimization as the optimization goal in the DS task migration and ensures the task delay requirement in the DT task migration. However, the specific calculation process of authentication delay between different CLP resources during task migration and the factors that may affect the authentication delay need to be explored next. At the same time, some users may move during task migration, which is another factor to be considered.

VI. Conclusions and Future Work

Cloudlet federation can effectively reduce the deployment and management cost of cloudlet by sharing resources among CLPs. However, the differences in the number of users and resources among CLPs bring new challenges to task migration. When studying the task migration strategy for the cloudlet federation, two types of tasks, DS and DT, are considered, and the relationship functions between user quotation and delay of these two types of tasks are designed. A task migration algorithm, called DPTM, which takes into account both delay and profit, is proposed. The DPTM algorithm consists of two sub-algorithms: TMDS and TMDT. The former is used to schedule DS tasks with delay as the main optimization objective. The latter is used to schedule DT tasks with profit as the primary optimization objective. Simulation results demonstrate the effectiveness of the proposed method.

In the future, we plan to design a centralized task migration strategy. In addition, the secure access problem when resources are shared between different CLPs is further discussed. Meanwhile, we intend to design a more intelligent task scheduling algorithm based on Artificial Intelligence (AI).

Acknowledgments

This research was supported by the National Natural Science Foundation of China [grant number 62262011], and the Foundation of Guilin University of Technology [grant number GUTQDJJ2002018].

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, August 2017.

[2] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," IEEE Transactions on Mobile Computing, vol. 18, no. 3, pp. 319–333, April 2018.

[3] T. H.Noor, S. Zeadally, A. Alfazi, Z. Quan, "Mobile cloud computing: Challenges and future research directions," Journal of Network and Computer Applications, vol. 115, no. 1, pp. 70–85, August 2018.

[4] M. Chen and V. C. M. Leung, "Reprint of: From cloud-based communications to cognition-based communications: A computing perspective," Computer Communications, vol. 131, pp. 77-82, October 2018.

[5] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, et al., "All one needs to know about fog computing and related edge computing paradigms: A complete survey," Journal of Systems Architecture, vol. 98, pp. 289-330, September 2020.

[6] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," Journal of Network and Computer Applications, vol. 59, pp. 46–54, January 2016.

[7] X. Cao, G. tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: towards an integrated service provisioning model," IEEE/ACM Transactions on Networking, vol. 28, no. 3, pp. 1116-1129, March 2020.

[8] R. A. Khan, T. L, and A. Khan, "Cloud Migration: Standards and Regulatory Issues with Their Possible Solutions," Int. J. Advanced Networking and Applications, vol. 10, no. 6, pp. 4113-4119, April 2019.

[9] S. Chen, J. Chen and C. Zhao, "Deep reinforcement learning based cloud-edge collaborative computation offloading mechanism," Acta Electronica Sinica, vol. 49, no. 1, pp. 157-166, 2021.

[10] I. Labriji, F. Meneghello, D. Cecchinato, S. Sesio, E. Perraud, et al., "Mobility aware and dynamic migration of MEC services for the internet of vehicles," IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 570-584, January 2021.

[11] H. Yuan, J. Bi, W. Tian, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," IEEE Transactions on Cybernetics, vol. 47, no. 11, pp. 3658-3668, July 2016.

[12] R. Soluma, R. Sasikala, S. Kshira Sager, K. R. Lakshmana, P. Quoc-viet and Dao. Nhu-Ngoc, "CAVMS: Application-Aware Cloudlet Adaption and VM Selection Framework for Multi-cloudlet Environment," IEEE Systems Journal, vol. 15, no. 4, pp. 5098-5106, 2021.

[13] A. Lakhan, M. A. Dootio, T. M. Groenli, A. H. Sodhro and M. S. Khokhar, "Multi-Layer Latency Aware Workload Assignment of E-Transport IoT Applications in Mobile Sensors Cloudlet Cloud Networks," Electronics, vol. 10, no. 14, pp. 1719, 2021.

[14] S. Moon and Y. Lim, "Task migration with partitioning for load balancing in collaborative edge computing," Appled Sciences-Basel, vol. 12, no. 3, pp. 1168, 2022.

[15] B. Qiao, C. Liu, J. Liu, Y. Hu, K. L. Li, and K. Q. Li, "Task migration computation offloading with low delay for mobile edge computing in vehicular networks," Concurrency and Computation: Practice and Experience, vol. 34, no. 1, July 2022.

[16] S. Moon and J. Park, Y. Lim, "Task migration based on reinforcement learning in vehicular edge computing," Wireless Communications and Mobile Computing, 2021.

[17] S. Huang, K. Lin, and C. Hu, "Intelligent task migration with deep Qlearning in multi-access edge computing," Iet Communications, vol. 16, no. 11, July 2022.

[18] C. Liu, F. Tang, Y. Hu, K. L. Li, Z. Tang, and K. Q. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 7, pp. 1603-1614, July 2021.

[19] M. Z. Nayyer, I. Raza, and S. A. Hussain, "CFRO: Cloudlet federation for resource optimization," IEEE Access, vol. 8, pp. 106234-106246, June 2020.

[20] H. Baghban, C. Huang, and C. H. Hsu, "Resource provisioning towards OPEX optimization in horizontal edge federation," Computer Communications, vol. 158, pp. 39-50, May 2020.

[21] L. Cui, C. X, S. Y, Z. H, X. W and Z. M, "Joint Optimization of Energy Consumption and Latency in Mobile Edge Computing for Internet of Things," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4791-4803, June 2019.

[22] I. A. Elgendy, W. Zhang, Y. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," Future Generation Computer Systems, vol. 100, pp. 531-541, November 2019.

[23] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268-4282, October 2016.

[24] X. Chen, Y. Cai, L. Li, M. Zhao, and B. Champagne, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," IEEE Transactions on Vehicular Technology, vol. 69, no. 2, pp. 2246-2262, February 2020.

[25] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin and X. Shen, "cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing," IEEE Transactions on Cloud Computing, vol. 9, no. 3, pp. 968-980, March 2019.