Task-Based Resource Allocation Bid in Edge Computing Micro Datacenter

Yeting Guo[1], Fang Liu[2, *], Nong Xiao[1] and Zhengguo Chen[1, 3]

Abstract: Edge computing attracts online service providers (SP) to offload services to edge computing micro datacenters that are close to end users. Such offloads reduce packet-loss rates, delays and delay jitter when responding to service requests. Simultaneously, edge computing resource providers (RP) are concerned with maximizing incomes by allocating limited resources to SPs. Most works on this topic make a simplified assumption that each SP has a fixed demand; however, in reality, SPs themselves may have multiple task-offloading alternatives. Thus, their demands could be flexibly changed, which could support finer-grained allocations and further improve the incomes for RPs. Here, we propose a novel resource bidding mechanism for the RP in which each SP bids resources based on the demand of a single task (task-based) rather than the whole service (service-based) and then the RP allocates resources to these tasks with following the resource constraints at edge servers and the sequential rule of task-offloading to guarantee the interest of SPs. We set the incomes of the RP as our optimization target and then formulate the resource allocation problem. Two typical greedy algorithms are adopted to solve this problem and analyze the performance differences using two different bidding methods. Comprehensive results show that our proposal optimizes resource utilization and improves the RP's incomes when resources in the edge computing datacenter are limited.

Keywords: Edge computing, resource allocation, task-offloading alternative.

1 Introduction

The growing size, popularity and number of applications available on the network have introduced more complex network structures and more serious network congestion [Cai, Wang, Zheng et al. (2013); Tan, Liu, Wang et al. (2019); Tan, Liu, Xie et al. (2019)]. High packet loss rates, delays and delay jitter act to prevent online service providers (SP) from promoting their services and meeting the growing functional demand from mobile users [Liu, Liu, Liu et al. (2019); Teng, Liu, Liu et al. (2019); Guo, Wu, Zhang et al. (2018)].

Edge computing [Liu, Guo, Cai et al. (2019); Zhao, Liu, Cai et al. (2018)] is a good solution for SPs that intelligently processes service requests at the edge of the network. In particular, edge computing has gradually evolved to play an irreplaceable role in application scenarios

[1] College of Computer, National University of Defense Technology, Harbin, 410073, China.

[2] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, 510006, China.

[3] Computer Science Department, University of Pittsburgh, Pittsburgh, 15213, USA.

[*] Corresponding Author: Fang Liu. Email: liufang25@mail.sysu.edu.cn.

that require enhanced mobile broadband, ultrahigh reliability and low latency communication, or massive Internet of Things communications [Guo, Liu, Cai et al. (2018); Kiss, Reale, Ferrari et al. (2018); Liu, Tang, Li et al. (2019)]. Increasingly, SPs seek to expand their profits with the help of edge computing.

An edge computing resource provider (RP) manages an edge computing micro datacenter that consists of several physical edge servers located near a base station [Aazam and Huh (2015)]. The value of edge computing as an emerging technology is widely recognized, but both SPs and RPs are still unsure about how to best deploy, manage and operate services in edge computing micro datacenters. This problem has become a hot research topic in both academia and industry [Sun, Zhou and Xu (2017); Fajardo, Taboada and Liberal (2015)]. However, few consider the acceptability of SPs. Suppose that each RP adopts a totally different edge computing system architecture; then, SPs must expend considerable effort to learn the corresponding user guide, which inevitably inhibits the scale and the commercialization of edge computing. To make edge computing friendlier and more acceptable to SPs, we believe that the existing architecture of a Content Distribution Network (CDN) [Shanmugam, Golrezaei, Dimakis et al. (2013)] is a good reference for edge computing system architecture design because the CDN service model is quite familiar to SPs, and traditional CDNs desperately need edge intelligence to transform and upgrade their services. Thus, we design a novel architecture, XDN, that combines existing CDN and the promising edge computing. The basic idea in XDN is that the edge computing micro datacenter does not simply cache content requested by users; instead, it caches computing instances that can handle the computing tasks most frequently involved in users' requests.

With the increasing emergence of novel services, demands for resources have also increased. Since resources in the edge computing micro datacenter are limited and unsatisfactory for meeting all SPs' resource demands, the major challenge in XDN is to determine which computing instances should be allocated resources and activated. Consequently, an efficient resource allocation scheme is vital to both RPs and SPs.

Resource allocation in edge computing has attracted significant attention in recent years [Alshuwaili and Simeone (2017); Liu, Bennis and Poor (2017)]. Most existing works assume that SPs have completed an evaluation of their resource demands and are ready to negotiate with the RP to obtain a resource supply. An SP who loses in a bidding war for resources with other SPs will obtain no resources from the edge computing micro datacenter; instead, all requests for services must be processed by the remote cloud datacenter without the help of the edge computing datacenter. We note that services always consist of several functionally independent tasks [Wang, Shen, Li et al. (2018); Wu, Pang, Dai et al. (2018); Fang, Cai, Sun et al. (2018)]. This fact allows SPs to maintain one or more task-offloading alternatives. Research on task offloading in edge computing always attempts to find the best task-offloading solution that achieves the highest service performance from the perspective of a single service. Each SP wants to adopt the best task-offloading solution but rarely can everyone get what they want due to the resource constraints in the edge computing micro datacenter. Beyond the best task-offloading solution, other task-offloading alternatives can be beneficial to the service even though they are somewhat less advantageous. We define such alternatives as 'acceptable alternatives'. If an SP were 1to flexibly adjust to some acceptable task-offloading alternatives to decrease resource demands when the best solution is rejected by the RP, it would enable the RP to make finer-grained edge computing resource allocation and further optimization and then increase SPs' possibilities of obtaining resources. In this paper, we are motivated to design a resource bidding mechanism for RPs to optimize edge computing resource allocation and maximize incomes. Each SP evaluates the gains resulting from each acceptable task-offloading alternative (task-based) rather than only one alternative (service-based), and then provides the RP with the corresponding resource demands, and the bidding prices. For each SP, only one acceptable alternative can be selected. Given more alternatives from the SPs, the RPs could make finer-grained global resource allocation optimization decisions [Liu, Dai and Wang (2004)].

In an edge computing micro datacenter, heterogeneous resources are distributed among multiple physical edge servers. Various resource constraints exist when allocating resources [Liu, Cai, Xu et al. (2015)]. Based on the above design and resource constraints in the edge computing micro datacenter, we model the edge computing resource allocation optimization problem among multiple SPs. Greedy algorithm is widely adopted to solve the resource allocation problem in the edge computing datacenter [Bahreini, Badri and Grosu (2018); Nakamura, Mizumoto, Suwa et al. (2018)], so we use two typical greedy algorithms to efficiently find the suboptimal solution for resource allocation. The main results and contributions of this paper are summarized as follows.

- Exploration of service deployment and management in edge computing: We propose a novel architecture, XDN, that caches computing instances to handle users' service requests at the edge of the network. The service deployment and management in XDN are based on existing CDN concepts, making edge computing more understandable and applicable to most SPs.
- Formulation of the edge computing resource allocation problem: We then discuss the edge computing resource allocation problem from two aspects. For SPs, we propose that each SP provides one or more acceptable task-offloading alternatives to increase their competitiveness during the edge computing resource auction. Given such alternatives, resources could be allocated in a more flexible and finer-grained way. From RPs, we take into account the various resource constraints in the edge computing datacenter and the sequential rules of task offloading in edge computing. Then we formulate the problem with the aim of maximizing the incomes of RPs.
- Evaluation of different resource bidding method: We adopts two typical greedy algorithms to match resource supplies and demands in the above resource allocation problem. Numerical results demonstrate that the task-based bidding method is superior to the service-based bidding method no matter what greedy algorithms we select. And we also analyze the impact of several main parameters on the performance differences.

The remainder of this paper is organized as follows. Section 2 introduces related works. Section 3 presents the mobile edge computing system architecture. The resource allocation problem formulation and the design of our algorithm are presented in Section 4. We present the evaluation results in Section 5. Finally, Section 6 concludes the paper.

2 Related work

Service functions are usually implemented by multiple tasks, so SPs have multiple task-offloading alternatives. Many research studies have been carried out on task offloading mechanism for SPs. LAVEA in Yi et al. [Yi, Hao, Zhang et al. (2017)] proposed to offload tasks between mobile terminals and edge servers based on their computational resources and the network bandwidth between them. In Mao et al. [Mao, Zhang and Letaief (2016)] the execution delay and energy assumption of edge server execution are investigated to offload tasks in a reasonable energy-saving way. In Zhao et al. [Zhao, Liu, Cai et al. (2017)] it senses the bandwidth of the network and the queue of requests to be processed in real time, dynamically adjusts the parameters of the deep learning model, and achieves efficient data processing. It could be obviously found that the resources that the RP allocates is vital to select the best task-offloading scheme. However, these works only consider their own performance, without considering that other services are also competing for limited resources in the edge computing micro datacenter.

Since resource supply in the micro datacenter always fails to meet the increasing demands of SPs, resource allocation draws more and more attention of researchers. Some works studied the allocation among multiple users [Mao, Zhang, Song et al. (2017); Guo, Song, Cui et al. (2017); Li, Martinez-Ortega and Diaz (2018)]. The authors in [Mao, Zhang, Song et al. (2017)] consider the dynamical resource demands and develop an online joint radio and computational resource management algorithm for multi-user edge computing systems. The authors in Guo et al. [Guo, Song, Cui et al. (2017)] discuss the resource allocation from the perspective of energy saving. The authors in Li et al. [Li, Martinez-Ortega and Diaz (2018)] take into account the interference among users and design a game theory to achieve distributed power control. But these works are applicable to mobile users but not to SPs. SPs tend to deploy their services in edge computing micro datacenters or cloud datacenters for security and management. Most works discuss the resource allocation among multiple SPs in an auction way [Zhang, Xiong and Lou (2014); Jin, Song and Zhuang (2018)]. RAERA in [Prasad, Arumaithurai, Koll et al. (2017)] proposed that RPs kept a reserved price based on historical data and the SP whose bidding price is higher than the reserved price and others' price is the winner. Zenith in Xu et al. [Xu, Palanisamy, Ludwig et al. (2017)] evaluated the utility of SPs and RPs, established resource sharing contact between them and then proposed a latency-aware scheduling and resource provisioning algorithm. The authors in Bahreini et al. [Bahreini, Badri and Grosu (2018)] design an envy-free auction mechanism to allocate resources to SPs. These works simply regard the resource demands of each SP as an indivisible unit. In this paper, we call this bidding method service-based. As we mentioned above, each SP has multiple choice to offload tasks. Different task-offloading alternatives require different resource demand and make different improvements on performance. But few works discuss the elastic prosperity of SPs' resource demands in the resource allocation.

Besides these works simply assume that the resources in the edge computing datacenter form a resource pool and the RP allocates resources from the pool. Actually, the resource allocation problem is more complex. These resources are distributed at physical edge servers. And demands on certain resources cannot be simply split into several parts and satisfied by different edge servers. Few works discuss the resource allocation problem among several edge servers in the same edge computing datacenter. The resource allocation schemes discussed for cloud infrastructure providers [Luo, Wang, Cai et al. (2019); Xie, Yuan, Zhou et al. (2018); Cheang, Wang, Cai et al. (2018)] are not suitable. These schemes are under the assumption that all these services could be deployed in the cloud when cloud resources are allocated in a reasonable way. But the assumption is not always supported in edge computing. Besides the placement of services is no longer confined to the remote datacenter but also could be an edge computing microdata center. Thus, it introduces new considerations to allocate edge resources to deploy these services.

3 System architecture

A content delivery network (CDN) caches content at geographically distributed edge servers; these proximity edge servers can respond quickly to requests. The rapid development of the IoT and AI technology has caused content to become more complicated. A traditional CDN configures the remote cloud data center to process dynamic computing services. This approach results in substantial overhead due to the long transmission distances. In contrast, edge computing leverages computing resources, storage resources and network resources available through edge servers to offload most computing tasks to the network edge. Only tasks that have high demand for computing resources and storage space are executed in the cloud.

Because both a CDN datacenter and an edge computing datacenter benefit from network edge proximity and because the CDN operation model is already familiar to most SPs, combining CDN and edge computing will become a future development trend [Zhang, Leng, He et al. (2018)]. Therefore, we introduce a novel XDN architecture that references the existing CDN architecture, making XDN more understandable and applicable to SPs. Compared with a CDN, XDN introduces the concept of edge computing to promote edge intelligence, that is, caching computing instances to handle computing tasks involved in service requests. The XDN architecture is shown in Fig. 1. Additional service deployment and management details in this architecture are presented below.

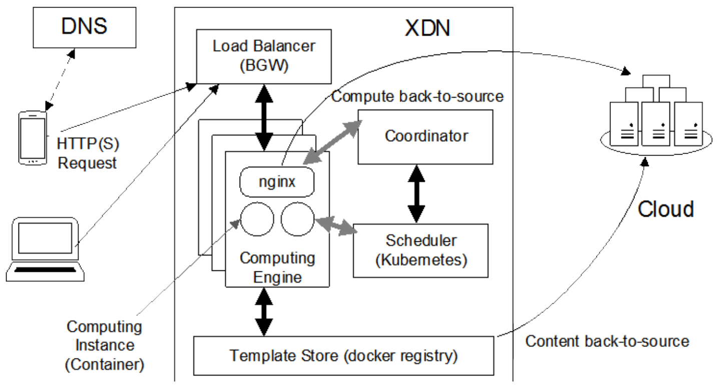


Figure 1: Architecture of XDN

SPs encapsulate their services into docker containers [Ma, Yi and Li (2017)] and migrate these containers to Template Store module in the edge computing micro datacenter. When a mobile terminal user accesses a service, the DNS parses the domain name and returns the IP address of the micro datacenter closet to the mobile user. Based on the IP address, the user sends an http(s) request to that datacenter. Upon receiving the request, the Load Balancer module further resolves the request header and then distribute the request to one of the computing engines based on certain rules. This module also fulfils the role of a billing gateway (BGW) to record the request and charge accordingly. Each computing engine runs several computing instances, where a computing engine and a computing instance correspond to an edge server and a container, respectively. Nginx is a powerful high-performance web and reverse proxy service [Reese (2008)] that is used to resolve the request content and then determine whether computing instances in the computing engine are able to fulfil the service request. When one can, Nginx distributes the request to the corresponding computing instance and redirects the response from the computing instance to the Load Balancer module. Finally, the module returns the result to the mobile user. When no computing instances are available to fulfil the request, Nginx suggests that the Load Balancer distributes the request to other computing engines similar to the way a traditional CDN redistributes requests when an edge server has not cached requested content. When the Load Balancer notices that the request has still not been responded to after multiple distributions, it finally asks the cloud to process the request.

From the description above, it's observed that computing instances running in computing engines have a direct influence on the performance of edge computing micro datacenters. Due to the limited resources, the computing engine are unable to execute all computing instances. The Coordinator module determines the resource allocation among computing instances. The Scheduler module is designed to manage containers and is implemented by Kubernetes [Bernstein (2014)]. When the Coordinator needs to modify the state of a container, it informs the Scheduler module to transfer that container from the Template Store to a corresponding computing engine or to remove the container from a computing engine. The Template Store module stores containers for various services; we apply a docker registry to register the stored containers. When the container that the Coordinator wants to activate does not exist in the Template Store module, the module would ask the cloud to migrate that container to the edge computing micro datacenter. In this way, the edge computing micro datacenter can respond to service requests at the edge of the network.

In this paper, we discuss the resource allocation schemes in the Coordinator module. Generally, the high-level functions of a service are always composed of several tasks. Each task is independent and achieves specific sub-functions, which are encapsulated into independent containers because they were likely developed by different research and development teams. Each container can be migrated to the edge computing micro datacenter or be kept in the remote cloud datacenter. Thus, resources at edge servers can be allocated based on resource demands of a single task.

Meanwhile, these tasks are also correlated to achieve the overall function. Sequential relationships are common among tasks, which adds basic constraints to most task-offloading mechanisms in edge computing. Thus, when considering offloading tasks, we divide the tasks into two parts: predecessor and successor tasks. In most cases, the edge computing micro datecenter plays a preprocessing role in service requests. Predecessor tasks are offloaded to the micro datacenter, and successor tasks are offloaded to the remote cloud datacenter when the micro datacenter is unable to execute them.

To better understand our work, we first provide an example, as shown in Fig. 2. In this scenario, the service is composed of three tasks, and the SP has four task-offloading alternatives. Alternative 4 provides maximum profit to the SP; however, the SP fails to acquire the appropriate edge resources in a bidding process with other SPs. In the existing research works, the losing SP would not be able to obtain any edge resources; that is, it would reluctantly be forced to adopt Alternative 1. These works neglect the possibility that the edge resource supply may be able to satisfy the resource demands of Alternative 3. The benefit accrued by alternative 3 is lower than that of Alternative 4 but higher than that of Alternative 1. Thus, if the SP were to adopt Alternative 3 and obtains the corresponding resources from a RP, it would be a win-win for both the RP and the SP.

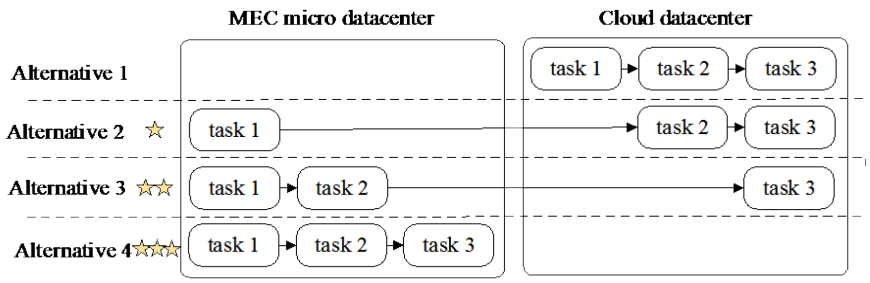


Figure 2: Example of task-offloading alternatives

This win-win possibility inspired us to propose a new resource bidding mechanism in which every SP evaluates the service performance advantages accrued by different task-offloading alternatives, provides the corresponding resource demands and bidding prices. Bidding prices are given by SPs and are usually determined by task performance improvement and resource demands. If adopting task-offloading alternatives can provide obvious performance improvements, the SP increases its profits and will certainly be willing to increase the bidding price to ensure that the resource demands of those alternatives are satisfied. In addition, bidding prices sometimes relate to service types and reflect the quality or importance of services. In this paper, we discuss resource allocation from the perspective of the RP to maximize the incomes. Thus, price models are not within the scope of this paper; readers can refer to works such as Xu et al. [Xu, Palanisamy, Ludwig et al. (2017); Aazam and Huh (2015)] for more information. Then we assume that the bidding prices have been calculated by SPs.

4 Problem

In this section, we fully consider the constraints when allocating edge resources, formulate the resource allocation problem and then adapt two typical greedy algorithms to efficiently allocate edge resources to multiple SPs and maximize the incomes of the RP. Tab. 1 summarizes the main notations in the formulation.

Table 1: Notations of the model

Notation	Definition
C	The number of edge servers in the edge computing datacenter
K	The number of resource types
$\{s_l^1, s_l^2 \cdots s_l^K\}$	Resource supply at server l
N	The number of services, that is, the number of SPs
h_i	The number of tasks in service i
$\{d_{i,j}^1, d_{i,j}^2 \cdots d_{i,j}^K\}$	Resource demands of the container $ct_{i,j}$
$u_{i,j}$	The payment of container $ct_{i,j}$
$x_{i,j}$	The server which deploys container $ct_{i,j}$
$sup_{i,j}$	The priority of container $ct_{i,j}$ in obtaining resources
w_v	The weight of resource v

Assume that the edge computing micro datacenter is a cluster of C homogeneous edge servers. Each server supplies K types of resources. $\{s_l^1, s_l^2 \cdots s_l^K\}$ $(l \in \{1,2 \cdots C\})$ denotes the resource supply of the l_{th} server. There are N OSPs attempting to obtain edge resources. The $i_{th}(i \in \{1,2 \cdots N\})$ service is composed of h_i tasks. Each task is encapsulated in a container. $\{d_{i,j}^1, d_{i,j}^2 \cdots d_{i,j}^K\}$ $(j \in \{1,2 \cdots h_i\})$ denotes the resource demand of the container $ct_{i,j}$ which encapsulates j_{th} task of i_{th} service. When deploying the container $ct_{i,j}$, the OSP's payment is $u_{i,j}$. $x_{i,j}$ indicates the server that the container $ct_{i,j}$ is deployed on. If $x_{i,j}$ is equal to l ($x_{i,j}$ == l), it means that the container is placed in the l_{th} edge server. If $x_{i,j}$ is equal to 0, it means that the container is kept in the cloud. Then the resource constraints in the edge computing micro datacenter are described as

$$\sum_{i=1}^{N}\sum_{j=1}^{h_i}\left(x_{i,j} == l\right) \times d_{i,j}^v \le s_l^v, \ \ \forall l \in \{1,2,\dots C\}, \forall v \in \{1,2,\dots K\} \tag{1}$$

Predecessor tasks are offloaded to the edge computing micro datacenter, and successor tasks are offloaded to the cloud datacenter. If j_{th} task is offloaded to edge servers, that is, $x_{i,j}$ is not equal to zero ($x_{i,j}! = 0$) , $(j\text{-}1)_{th}$ task would also be offloaded to the micro datacenter ($x_{i,j-1}! = 0$). Thus, the sequential rule of task-offloading can be described as

$$(x_{i,j}! = 0) \le (x_{i,j-1}! = 0) \tag{2}$$

From the incomes of the RP, the optimization goal is expressed as

$$max \sum_{i=1}^{N}\sum_{j=1}^{h_i}(x_{i,j}! = 0) \times u_{i,j} \tag{3}$$

In essence, the problem is Multiple-dimension Multiple-choice Multiple-knapsack knapsack problem (MMMKP). More specifically, various resource constraints and ordering constraints make the knapsack problem multidimensional. Every SP has one or more scheme choices. And containers that encapsulate tasks can be placed in alternative available edge servers, which could be regarded as knapsacks. Therefore, the problem is NP hard problem.

Greedy algorithm (GA) is simple and high-efficient method to solve resource optimization problems [Bahreini, Badri and Grosu (2018); Nakamura, Mizumoto, Suwa et al. (2018); Zhang, He, Suto et al. (2018)]. We investigate two typical greedy algorithms among them and make simple modifications to them so that they could be applicable to the task-based resource bid. In the first greedy algorithm, the prices of tasks represent the priority in obtaining resources, that is, $sup_{i,j} = u_{i,j}$. And the second greedy algorithm calculates the weight of each resource $\{w_1, w_2 \cdots w_K\}$ according to the Equation 4 and then sets the priority of tasks based on the prices of tasks, the resource demands and the weights of resources according to Eq. (5). Since the priority of each task is determined, the following step in the two algorithms is to allocate resources to tasks, shown in Algorithm 1.

$$w_v = \frac{\sum_{l=1}^{C} s_l^v}{\sum_{i=1}^{N} \sum_{j=1}^{h_i} d_{i,j}^v}, \forall v \in \{1,2 \cdots K\} \tag{4}$$

$$sup_{i,j} = \frac{u_{i,j}}{\sum_{v=1}^{K} w_v \times d_{i,j}^v} \tag{5}$$

Algorithm 1: task-based resource allocation

Input: the priority of each task $sup_{i,j}$, the resource demand of each task $\{d_{i,j}^1, d_{i,j}^2 \cdots d_{i,j}^K\}$ and the resource supply at each edge server $\{s_l^1, s_l^2 \cdots s_l^K\}$

Output: the placement of each task $x_{i,j}$

```
1.  **Initialize** all $x_{i,j}$ // $x_{i,j}$ is initialized to be zero
2.  $wait\_tasks = \{sup_{1,1}, sup_{2,1} \cdots sup_{N,1}\}$ // collect the priority of the first task in each
    service
3.  $allocate\_task = [0] \times N$
4.  **While** $wait\_tasks != []$ **do**
5.     $[sv, tk] = max(wait\_tasks)$ // the corresponding service and task whose priority
    is the highest among wait tasks
6.      $penalty = [0] \times C$ // initial the penalty of each server
7.      **for** $l = 1; l \le C; l = l + 1$ **do**
8.          $penalty_l = \sqrt{\sum_{v=1}^{K} (\frac{d_{sv,tk}^v}{s_l^v})^2}$
9.      // no servers satisfy the resource demands of this task
10.     **if** $min(penalty) > 1$ **then**
11.         // make the resolution follow the sequential rule of task-offloading
12.         **for** $i = tk; i \le h_{sv}; i = i + 1$ **do**
13.             $x_{sv,i} = 0$
14.         delete $sup_{sv,tk}$ from $wait\_tasks$
15.     **else**
16.         $x_{sv,tk} = penalty.index(min(penalty))$
17.         update the resource supply in server $x_{sv,tk}$
```

```
18.             $allocate\_task_{sv} = tk$
19.         delete $sup_{sv,tk}$ from $wait\_tasks$
20.         **if** $tk \leq h_{sv}$ **then**
21.             add $sup_{sv,tk+1}$ to $wait\_tasks$
22.   return all $x_{i,j}$
```

5 Evaluation

To evaluate the performance of our scheme, we did multiple simulations in this section. In the simulation, we mainly compared the performance differences caused by different bidding methods. In the service-based bidding method, we also adopted the basic idea of the two algorithms and made the following modifications. The priority of a service is the sum of the priorities of all tasks that make up the service. And if no edge servers could satisfy the resource demands of one task in the service, all the tasks in the service would not obtain resources from the edge computing micro datacenter.

5.1 Experimental setup

We simulate that there exist 14 edge servers ($N = 14$) in the micro datacenter. And the configuration of these edge servers is the same as ecs.ga1.14xlarge provided by AWS, shown in Tab. 2.

Table 2: Configuration of edge server

Resource type	Resource supply
vCPU	56
Memory (GiB)	160
Local dists (GiB)*	1400
GPU	4
Bandwidth (Gbit/s)**	10
Packet forwarding rate (Thousand pps) ***	120
NIC queues ****	4
ENIs *****	8

The resource demands of each task are randomly generated without exceeding the scope of the resource supply of an idle server. It reflects the different needs of different tasks for different resources. Since there are various resources in edge servers, multiple supply and demand ratios exist. For the convenience to present simulation results, we introduce a new variable, denoted by p, and defined as Eq. (6), to represent the overall resource supply and demand ratio. The resource allocation scheme is designed for the case that SPs' resource demands exceed resource supplies in the micro datacenter. Thus, we set the range of p from 0.1 to 2.0 with an interval of 0.1. Note that it does not mean that all SPs could be satisfied even if p is equal to 1 because p is calculated based on the statistics of the overall resource demand and supply, but actually resource constraints are independent, and the resource allocation scheme must strictly meet all resource constraints. These tasks are randomly assigned to services. Each service is composed of 7 tasks at most ($h_i \leq 7$).

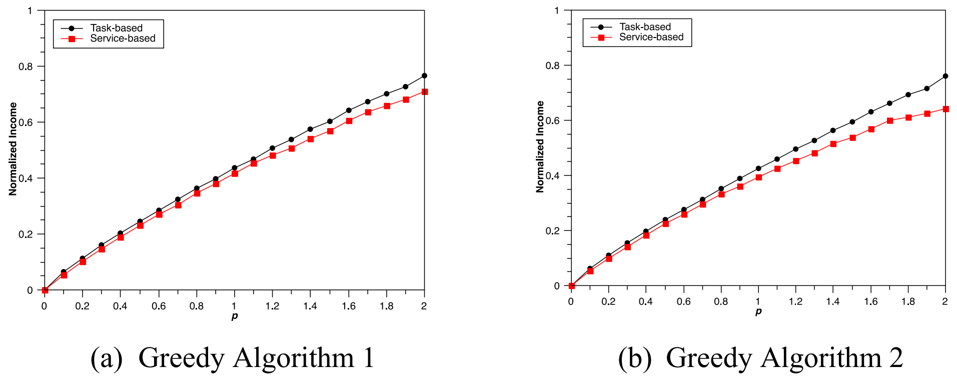
$$p = min\{w_1, w_2 \cdots w_K\} \qquad (6)$$

The bidding price of each task in services $u_{i,j}$ is randomly generated in the range from 1000 to 1200. It aims to make the competitiveness of SPs roughly equal. Otherwise, weak competitive SPs may never get the resources they want and make no difference in evaluating the performance. And we repeat experiments 500 times in each set of simulations.

5.2 Analysis of result

As our scheme aims to optimize edge resource allocation for RPs, the most important performance matrix is the incomes of RPs. We measure the maximum incomes that the RP gains when meeting all resource demands, then calculate the ratio of the incomes gained in different bidding methods and the maximum income to normalize the income. The normalized income intuitively presents the performance differences among methods in terms of income. And then we analyze the impact of the maximum number of tasks in a service and the number of servers respectively.

Firstly, we evaluate the normalized incomes under different overall supply and demand ratios. Results are shown in Fig. 3. It illustrates that task-based bidding method always shows greater superiority than service-based bidding method no matter what greedy algorithm we select. When the ratio is greater than 1, the task-based bidding method improves the incomes by 2.12%-11.87%. With the ratio increasing, the relative performance difference is decreasing but the absolute performance difference is increasing. The reason is that when the ratio is extremely small, the RP can hardly meet any resource demands of any SPs whatever the bidding method is adopted.

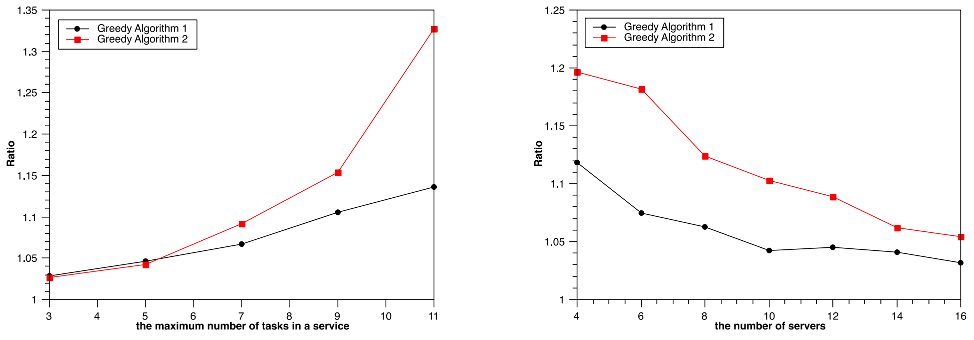


(a) Greedy Algorithm 1 (b) Greedy Algorithm 2

Figure 3: Normalized income of schemes

Then we analyze the impact of the maximum number of tasks in a service on the performance differences between the two bidding methods. In this simulation, we set this parameter range from 3 to 11 and set the overall supply and demand ratio to be 1. Other parameters are kept the same as Section 5.1. Then we calculate the ratio of normalized incomes between the task-based bidding method and the service-based bidding method. Results are shown in Fig. 4(a). The experiment results of the two algorithms present out the same change trend. When the maximum number of tasks in a service increases, the ratio is also increasing. It means that their performance differences are becoming more and more significant. Thus, the maximum number of tasks is the main cause of performance differences.

In the next set of simulations, we investigate the impact of the number of servers, C, on the ratio of normalized incomes. The number of servers is another factor that the RP concerns because every RP wants to make the best use of existing server resources to meet the increasing demands instead of simply increasing the number of servers. We firstly generate resource demands when the number of servers is 10 and the overall supply and demand ratio is 1, and then update the resource supply, that is, the number of servers ranges from 4 to 16. We calculate the performance ratio under different number of servers. Results are shown in Fig. 4(b). It is observed that the ratio is always greater than 1 regardless of the number of servers. It means that when the resource supplies at these edge servers cannot meet the demand of SPs and the number of servers remains unchanged, task-based bidding method shows more superiority to service-based bidding method in resource utilization. We also observed that the performance gap is disappearing as the number of servers increases. The reason is that the resource supply at edge servers is gradually becoming saturated, and the normalized incomes of both task-based bidding method and service-based bidding method are approaching to 1.



(a) The impact of the maximum number of tasks in a service (b) The impact of the number of servers

Figure 4: The impact of parameter settings on performance differences

6 Conclusion

In this paper, we first design a novel architecture, XDN, to explore service deployment and management in edge computing micro datacenters. Then, we focus on optimizing resource allocation for the benefit of RPs. Our design proposes a task-based resource bidding mechanism among multiple SPs which allows the RPs allocate resources in a more fine-grained and flexible way. In the resource allocation model formulation, we fully consider the resource constraints and SPs' expectations of edge computing when allocating resources. Then we investigate two typical greedy algorithm that are often introduced in solving resource allocation problem. We have made simple modifications to these two algorithms so that they can be applicable to task-based resource bidding method.

Simulation compared the performance differences of the task-based resource bid and the service-based resource bid. Results demonstrated that the task-based bidding method improves resource utilization and the incomes when edge resources are in short supply. In the future, we will continue to augment XDN to perfect its functionality and make it more suitable to SPs. Besides we will design an efficient and effective algorithm that is more suitable to solve the resource allocation problem in the edge computing micro datacenter.

Acknowledgement: This work is supported by The National Key Research and Development Program of China (2016YFB1000302) and National Natural Science Foundation of China (61433019, 61832020).

References

Aazam, M.; Huh, E. (2015): Dynamic resource provisioning through fog micro datacenter. *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 105-110.

Aazam, M.; Huh, E. (2015): Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. *IEEE International Conference on Advanced Information Networking and Applications*, pp. 687-694.

Alshuwaili, A.; Simeone, O. (2017): Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398-401.

Bahreini, T.; Badri, H.; Grosu, D. (2018): An envy-free auction mechanism for resource allocation in edge computing systems. *IEEE/ACM Symposium on Edge Computing*, pp. 313-322.

Bernstein, D. (2014): Containers and cloud: from lxc to docker to kubernetes. *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81-84.

Cai, Z.; Wang, Z.; Zheng, K.; Cao, J. (2013): A distributed TCAM coprocessor architecture for integrated longest prefix matching, policy filtering, and content filtering. *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 417-427.

Cheang, C. F.; Wang, Y.; Cai, Z.; Xu, G. (2018): Multi-VMs intrusion detection for cloud security using dempster-shafer theory. *Computers, Materials & Continua*, vol. 57, no. 2, pp. 297-306.

Fang, S.; Cai, Z.; Sun, W.; Liu, A.; Liu, F. et al. (2018): Feature selection method based on class discriminative degree for intelligent medical diagnosis. *Computers, Materials & Continua*, vol. 55, no. 3, pp. 419-433.

Fajardo, J. O.; Taboada, I.; Liberal, F. (2015): Improving content delivery efficiency through multi-layer mobile edge adaptation. *IEEE Network*, vol. 29, no. 6, pp. 40-46.

Guo, S.; Wu, D.; Zhang, H.; Yuan, D. (2018): Resource modeling and scheduling for mobile edge computing: a service provider's perspective. *IEEE Access*, vol. 6, no. 99, pp. 35611-35623.

Guo, Y.; Liu, F.; Cai, Z.; Xiao, N.; Zhao, Z. (2018): Edge-based efficient search over encrypted data mobile cloud storage. *Sensors*, vol. 18, no. 1189, pp. 1-13.

Guo, J.; Song, Z.; Cui, Y.; Liu, Z.; Ji, Y. (2017): Energy-efficient resource allocation for multi-user mobile edge computing. *IEEE Global Communications Conference*, pp. 1-7.

Jin, A.; Song, W.; Zhuang, W. (2018): Auction-based resource allocation for sharing cloudlets in mobile cloud computing. *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 45-57.

Kiss, P.; Reale, A.; Ferrari, C. J.; Istenes, Z. (2018): Deployment of IoT applications on 5G edge. *IEEE International Conference on Future IoT Technologies*, pp. 1-9.

Li, N.; Martinez-Ortega, J.; Diaz, V. H. (2018): Distributed power control for interference-aware multi-user mobile edge computing: a game theory approach. *IEEE Access*, vol. 6, pp. 36105-36114.

Liu, Y.; Liu, A.; Liu, X.; Huang, X. (2019): A statistical approach to participant selection in location-based social networks for offline event marketing. *Information Sciences*, vol. 480, pp. 90-108.

Liu, C.; Bennis, M.; Poor, H. V. (2017): Latency and reliability-aware task offloading and resource allocation for mobile edge computing. *IEEE Global Communications Conference Workshops*, pp. 1-7.

Liu, F.; Dai, K.; Wang, Z. (2004): Improving security architecture development based on multiple criteria decision making. *Advanced Workshop on Content Computing*, pp. 214-218.

Liu, F.; Guo, Y.; Cai, Z.; Xiao, N.; Zhao, Z. et al. (2019): Edge-enabled disaster rescue: a case study of searching for missing people. *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 5, pp. 36-49.

Liu, F.; Tang, G.; Li, Y.; Cai, Z. P.; Zhang, X. et al. (2019): A survey on edge computing systems and tools. *Proceedings of the IEEE*, vol. 107, no. 10, pp. 17-34.

Liu, S.; Cai, Z.; Xu, H.; Xu, M. (2015): Towards security-aware virtual network embedding. *Computer Networks*, vol. 91, pp. 151-163.

Luo, M. H.; Wang, K.; Cai, Z. P.; Liu, A. F.; Li, Y. Y. et al. (2019): Using imbalanced triangle synthetic data for machine learning anomaly detection. *Computers, Materials & Continua*, vol. 58, no. 1, pp. 15-26.

Ma, L.; Yi, S.; Li, Q. (2017): Efficient service handoff across edge servers via docker container migration. *IEEE/ACM Symposium on Edge Computing*, pp. 11-23.

Mao, Y.; Zhang, J.; Letaief, K. B. (2016): Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590-3605.

Mao, Y.; Zhang, J.; Song, S.; Letaief, K. B. (2017): Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994-6009.

Nakamura, Y.; Mizumoto, T.; Suwa, H.; Arakawa, Y.; Yamaguchi H. et al. (2018): In-situ resource provisioning with adaptive scale-out for regional IoT services. *IEEE/ACM Symposium on Edge Computing*, pp. 203-213.

Prasad, A. S.; Arumaithurai, M.; Koll, D.; Fu, X. (2017): Raera: a robust auctioning approach for edge resource allocation. *ACM Special Interest Group on Data Communication Workshop on Mobile Edge Communications*, pp. 49-54.

Reese, W. (2008): Nginx: the high-performance web server and reverse proxy. *Linux Journal*, vol. 173, no. 2, pp. 1-4.

Shanmugam, K.; Golrezaei, N.; Dimakis, A. G.; Molisch, A. F.; Caire, G. (2013): Femtocaching: wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402-8413.

Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. (2016): Edge computing: vision and challenges. *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646.

Sun, Y.; Zhou, S.; Xu, J. (2017): Emm: energy-aware mobility management for mobile edge computing in ultra dense networks. *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637-2646.

Tan, J.; Liu, W.; Wang, T.; Xiong, N.; Song, H. et al. (2019): An adaptive collection scheme-based matrix completion for data gathering in energy-harvesting wireless sensor networks. *IEEE Access*, vol. 7, pp. 6703-6723.

Tan, J.; Liu, W.; Xie, M.; Song, H.; Liu, A. et al. (2019): A low redundancy data collection scheme to maximize lifetime using matrix completion technique. *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 5.

Teng, H.; Liu, Y.; Liu, A.; Xiong, N. N.; Cai, Z. et al. (2019): A novel code data dissemination scheme for Internet of Things through mobile vehicle of smart cities. *Future Generation Computer Systems*, vol. 94, pp. 351-367.

Wang, R.; Shen, M.; Li, Y.; Gomes, S. (2018): Multi-task joint sparse representation classification based on fisher discrimination dictionary learning. *Computers, Materials & Continua*, vol. 57, pp. 25-48.

Wu, H.; Pang, B.; Dai, D. (2018): Unmanned aerial vehicle recognition based on clustering by fast search and find of density peaks (CFSFDP) with polarimetric decomposition. *Electronics*, vol. 7, no. 364, pp. 1-18.

Xie, X.; Yuan, T.; Zhou, X.; Cheng, X. (2018): Research on trust model in container-based cloud service. *Computers, Materials & Continua*, vol. 56, no. 2, pp. 273-283.

Xu, J.; Palanisamy, B.; Ludwig, H.; Wang, Q. (2017): Zenith: utility-aware resource allocation for edge computing. *IEEE International Conference on Edge Computing*, pp. 47-54.

Yi, S.; Hao, Z.; Zhang, Q.; Zhang, Q.; Shi, W. et al. (2017): Lavea: latency-aware video analytics on edge computing platform. *IEEE International Conference on Distributed Computing Systems*, pp. 2573-2574.

Zhang, J.; Xiong, T.; Lou, W. (2014): Community clinic: economizing mobile cloud service cost via cloudlet group. *IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 208-216.

Zhang, K.; Leng, S.; He, Y.; Maharjan, S.; Zhang, Y. (2018): Cooperative content caching in 5G networks with mobile edge computing. *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80-87.

Zhang, S.; He, P.; Suto, K.; Yang, P.; Zhao, L. et al. (2018): Cooperative edge caching in user-centric clustered mobile networks. *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1791-1805.

Zhao, Z.; Liu, F.; Cai, Z.; Xiao, N. (2018): Edge computing: platforms, applications and challenges. *Journal of Computer Research & Development*, vol. 55, no. 2, pp. 327-337.

Zhao, Z.; Liu, F.; Cai, Z.; Xiao, N. (2017): Edge-based content-aware crowdsourcing approach for image sensing in disaster environment. *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 225-231.