Task Offloading Decision in Fog Computing System

Qiliang Zhu[1], Baojiang Si[1], Feifan Yang[1], You Ma[2]

[1] School of Information Engineering, North China University of Water Resources and Electric Power, Zhengzhou 450046, China
[2] National Satellite Meteorological Center, Beijing 100081, China

Abstract: Fog computing is an emerging paradigm of cloud computing which to meet the growing computation demand of mobile application. It can help mobile devices to overcome resource constraints by offloading the computationally intensive tasks to cloud servers. The challenge of the cloud is to minimize the time of data transfer and task execution to the user, whose location changes owing to mobility, and the energy consumption for the mobile device. To provide satisfactory computation performance is particularly challenging in the fog computing environment. In this paper, we propose a novel fog computing model and offloading policy which can effectively bring the fog computing power closer to the mobile user. The fog computing model consist of remote cloud nodes and local cloud nodes, which is attached to wireless access infrastructure. And we give task offloading policy taking into account executi+on, energy consumption and other expenses. We finally evaluate the performance of our method through experimental simulations. The experimental results show that this method has a significant effect on reducing the execution time of tasks and energy consumption of mobile devices.

Keywords: fog computing; task offloading; energy consumption; execution time

I. INTRODUCTION

With the advancement of mobile computing, mobile users experience a large number of fascinating applications. However, while the demand for newly developed applications continues to grow, the computing power of mobile devices is still limited due to its portable size. The contradiction between computation intensive applications and resource limited devices becomes the bottleneck for providing satisfactory quality of experience. And the gap between demand for executing complex tasks and availability of limited resources becomes more and more remarkable [1-3].

A traditional approach to overcome the resource poverty of mobile devices is to leverage the rich computing resource of clouds. A mobile device can reduce its workload and prolong its battery life by offloading its computation-intensive tasks to a remote cloud [4,5]. However, one of important limitations of offloading tasks to a remote cloud is that the distance between the mobile user and remote cloud is usually long. Such a long distance resulting in long delays and even lags in applications with heavy user interactions, reduce the user experiences. Fog computing is an emerging paradigm of cloud computing to meet the growing computation demand of mobile application. It can help mobile devices to overcome resource constraints by offloading the computationally intensive tasks to the remote cloud servers [6].

Fog computing aims to enable the mobile devices to execute computationally intensive tasks directly at the network edge. The advantages of Fog computing are closeness

Received: May 16, 2017
Revised: Jun. 12, 2017
Editor: Shangguang Wang

to end-users, location awareness, low latency, and mobility support. It can help mobile devices to overcome resource constraints by offloading the computationally intensive tasks to the remote cloud servers [7]. The challenge of the cloud is to minimize the time of data transfer and task execution to the user, whose location changes owing to mobility. To provide satisfied computation performance is particularly challenging in fog computing environment.

In fog computing system, task offloading is a critical point which enables mobile devices to support resource intensive applications by executing them on the resource of the fog cloud. Task offloading involves additional data communication, which may increase the overhead. Thus, to determine whether task offloading is beneficial or not, the mobile device user should check whether the time and energy consumption executing the task on the cloud is less than executing it locally. The charges of processing data in the cloud should also be a concern for mobile users. If the charge is too high, even to save time and energy consumption, the users will not agree, so the charges should also be considered in the task offloading. In order to solve this problem, we propose a novel fog computing model and offloading policies in this paper, which can effectively bring the fog computing power closer to the mobile user. The fog computing model consists of remote cloud nodes and local cloud nodes, which is attached to wireless access infrastructure. And we give task offloading policies taking into execution time, energy consumption and other expenses.

The remainder of this paper is organized as follows: In the next section we give the related work of fog computing and task offloading. Section III describes our approach in detail. Section IV presents results and discussion. Finally, conclusions and future work are given in Section V.

II. Related Work

Fog computing is referred to as a cloud system which enables resources that cannot be executed on mobile devices to other mobile devices or to the cloud [14-19]. Habak et al. [19] present a FemtoCloud system architecture that provide cloud services at the edge by coordinating multiple mobile devices. FemtoClouds leverage the nearby available mobile devices to serve "compute as a service" at the network edge, thereby reducing the network latency during task offloading to the traditional cloud. In [18], the authors propose REPLISOM, a cloud resources augmented NodeB(eNB) model with an LTE-optimized memory replication protocol for the Internet of things services and applications. REPLISOM effectively schedules the memory replication occasions to settle contentions for the radio resources as a massive number of devices transmit their memory replicas simultaneously.

In [15], the authors propose to deploy cloud servers at the network edge and design the edge cloud as a tree hierarchy of geo-distributed servers, so as to efficiently utilize the cloud resources to serve the peak loads from mobile users. The hierarchical architecture of edge cloud enables aggregation of the peak loads across different tiers of cloud servers to maximize the amount of mobile workloads being served. Tong et al. [16] propose proposes a Mobile Cloud model for Assistive Healthcare (MoCAsH). In addition to inheriting the advantages of cloud computing, MoCAsH embraces significant concepts of mobile sensing, active sensor records, and collaborative planning by deployment of intelligent mobile agents, context aware middleware, and collaborative protocol for resource planning and sharing.

In order to improve the performance of mobile computing, a number of recent works propose different methodologies to offload tasks for specific applications [8-13]. To achieve energy saving while satisfying given application execution time requirement, Huang et al. [9] present a dynamic offloading algorithm based on Lyapunov optimization. This algorithm is dedicated to reducing the complexity of the offloading problem and saving more

energy for mobile devices. To provide satisfactory computation performance, Mao et al. [12] investigate a green fog computing system with energy harvesting devices and develop an effective computation offloading strategy. The decisions of this system depend only on the current system state without requiring information of the computation task request distribution, wireless channel, and energy harvesting processes.

Other works to solve the offloading problem based on the trade-off analysis [20-25]. In order to study the question of how to rationally selection the best provider among all candidates, Gelenbe et al. [24] propose a mathematical model of energy-QoS trade-offs at the local and remote cluster, and solve this problem by formulating an optimization problem. Wu et al. [25] employ queuing theory to model the offloading systems and utilize the energy-response time weighted sum as metric to capture energy-performance trade-offs. The optimal assignment policy can get an appropriate trade-off between minimizing energy costs and minimizing delay.

In [26], the authors discuss the trade-off between shortening execution time and saving energy consumption in cloud offloading systems. Based on the trade-off analysis, they propose and study a novel adaptive offloading system. Chen et al. [27] propose an efficient k-out-of-n task offloading algorithm that reduces the task execution time and minimizes the wasted energy in executing duplicated tasks on multiple processor nodes. In addition, the trade-off between system reliability and overhead is analysed in terms of more storage space and redundant tasks. The work in [28] presents a sensing platform that implants a novel adaptive sampling scheme based on machine learning algorithm, and a dynamic computation distribution mechanism based on decision theory. The sensor sampling scheme adaptively samples from Bluetooth, accelerometer, and sensors while balancing energy-latency-accuracy trade-offs.

Most of the previous works only study a kind of offloading strategy or a framework of fog computing, however, the offloading problem in a variety of situations is seldom studied. In this paper, we propose a novel fog computing model, and investigate the task offloading strategy taking into account execution time, energy consumption and other expenses.

III. Task Offloading Model

In this section, we discuss task migration strategies in fog computing system. Firstly, we will provide a brief introduction of fog computing system. There are four major components involved in the system, mobile user, agent, local cloud and remote cloud as illustrated in figure 1.

Mobile users interact with cloud through mobile devices, which access the cloud via WiFi access points or Internet. Mobile devices are always subject to strict constraints on their resource such as CPU, memory and battery, when run a very complex application. The local cloud deployment around the user, which can provide limited computing resource, but directly accessible by users and can avoid additional communication delay. The remote cloud can provide abundant computing resources, but located on physically far from the

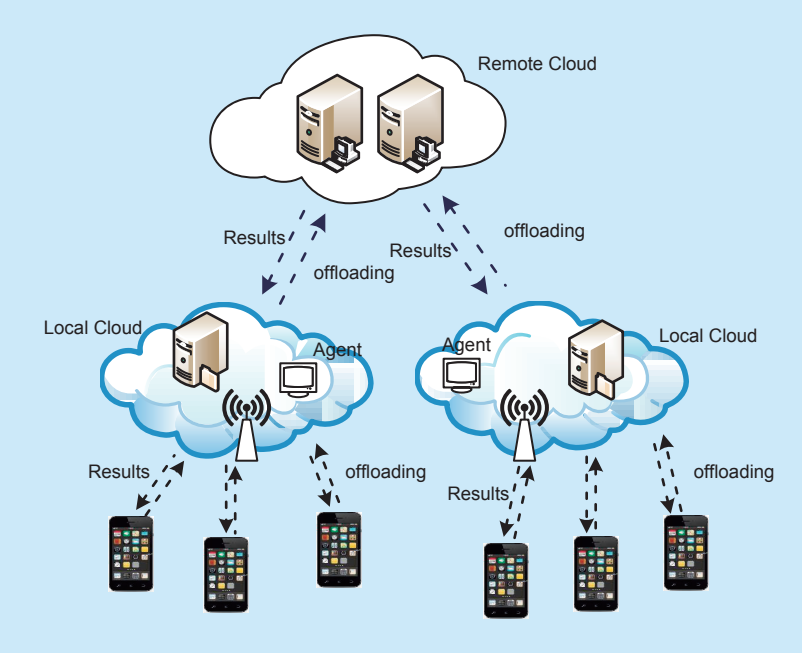


Fig. 1 *Task offloading model*

users and the transmission delay is large. The agent is a control center, which can collect resource information such as CPU utilization, battery level, transmission speed and network bandwidth. Based on these information, the agent decides that the tasks are running on the mobile device, offloading to the local cloud or offloading to the remote cloud.

Once a new task is generated by a user, it can be executed on the mobile device. If the mobile device is not able to perform this task or take too long, the manager would decide that the task should be offload to the local cloud through the wireless link. If the local cloud is not able to perform this task or take too long, the manager would decide that the task should be offload to the remote cloud through the Internet.

3.1 Problem description

There are several cloud service providers (such as Amazon, Alibaba) to provide rich cloud computing services. Let $R_i = \{ID, C^r, B^r, E^r, P^r\}$ represent a remote cloud node *i*, where *ID* is the identifier, C^r represent the computation capability of cloud node i, and B^r is the available bandwidth between the remote cloud node and mobile device in Mbps. E^r is the energy consumption for each unit of data transferred to the remote cloud node. P^r represents the charges for processing unit data on remote cloud node.

There are also a number of local cloud nodes around users to provide cloud services. Let $L_j = \{ID, C^l, B^l, S_j, E^l, P^l\}$ denote a local cloud node, where *ID* is the identifier, C^l represent the computation capability of local cloud node *j*. B^l is the available bandwidth between the local cloud node and the mobile device in Mbps. *S* indicates the signal area of cloud node *j*, beyond this area the cloud node cannot provide cloud services. E^l is the energy consumption for each unit of data transferred to the remote cloud node. P^l represents the charges for processing unit data on remote cloud node.

The task generated by the mobile user can be expressed as follows: $T_j = \{D_j, M_j, D_j^r, t_j^d\}$, where D_j is the upload data size of task j, M_j is the amount of tasks to be processed on the cloud node, D_j^r represent the data size of task result, t_j^d indicates the deadline time a task execution. P^r is the charges for processing unit data on local cloud node.

When a task is offloaded, the mobile user may have two main expectations, including saving the energy of the mobile device, reducing the execution time of the task. Next, we will analyse the two parts separately.

3.2 Execution time analyses

Execution time is not only a factor of great concern to mobile users, but also an important indicator of the quality of user experience. According to task offloading process, the task execution time can be divided into 4 parts.

1) Offloading decision phase t_1. It mainly including information collection, offloading decision based on the collected information, and necessary waiting time.

2) Data transfer phase t_2. After the offloading decision is made, the data need to be transfered from the source node to the cloud node (local cloud or remote cloud). And the transfer time can be expressed as follows:

$$t_2 = D / B \ , \tag{1}$$

where *D* is the upload data size of the task, *B* (B^r or B^l) is the network bandwidth.

3) Task execution phase t_3. It refers to the time to perform tasks on a cloud node, the time overhead is as follows:

$$t_3 = M / C \ , \tag{2}$$

where M is the amount of tasks need to be processed on the cloud node, C (C^r or C^l) is the computation capability of cloud node.

4) Results return phase t_4. It refers to the time required to return the result data from the cloud node to the mobile device, here, we assume that the bandwidth remains the same for the bidirectional communication. The time can be expressed as follows:

$$t_4 = D^r / B, \quad (3)$$

where D^r is the data size of task result, B (B^r or B^l) is the network bandwidth.

In summary, the time overhead for task offloading is:

$$T = t_1 + \frac{D + D^r}{B} + \frac{M}{C}. \quad (4)$$

Generally speaking, the purpose of the task offloading is to reduce the time of task execution, that is to say the execution time meets the following condition,

$$T_m > t_1 + \frac{D + D^r}{B} + \frac{M}{C}, \quad (5)$$

where T_m is the time a task executed on mobile device, and other parameters are the same with above. It is worth offloading the task to the cloud node, if the task can be performed faster on the cloud than on the mobile device.

3.3 Energy consumption analysis

Energy is another important indicator that must be considered when making offloading decisions. The survey shows that users believe that longer battery life is more important than other features. Mobile devices are more and more frequently used for watching videos, interactive gaming, augmented reality and other purposes which consume huge electricity and shorten the battery life as a result. In addition, these applications are too computation intensive to be executed on mobile devices.

In order to study and analyze the energy consumption of offloading model, we do not consider the task switching process in the offloading process. Although the execution of the task will produce energy consumption, but the energy consumption is mainly generated by the cloud node, and we mainly consider the energy consumption of mobile devices, so the energy consumption of the task execution process is negligible. The energy consumption of the task offloading process mainly includes three parts: the energy consumption of uploading data and the energy of consumption returning data. The two parts are expressed as follows:

1) The energy consumption of uploading data E_1: The energy consumed by uploading data from the mobile device to the local cloud node.

$$E_1 = D_t * E_t, \quad (6)$$

where D_t is the data size of uploading task, E_t (E^r or E^l) is the energy consumption per unit data transmission.

2) The energy consumption of returning data E_2: The energy consumed by returning result data from local cloud node to mobile device.

$$E_2 = D_r * E_t, \quad (7)$$

where D_r is the data size of results, E_t is the same as above.

In summary, the energy consumption of task offloading process is:

$$E = (D_t + D_r) * E_t, \quad (8)$$

where the parameters are the same with above.

The other purpose of the task offloading is to reduce the energy consumption of task execution, that is to say the execution energy consumption should meets the following condition,

$$E_m > (D_t + D_r) * E_t, \quad (9)$$

where E_m is the energy consumption a task executed on mobile device. Here, the idle time of mobile device is not to be considered. So It is worth offloading the task to the cloud node, if the energy consumption on the cloud than on the mobile device.

3.4 Task offloading policy

In this section, we discuss task offloading policy based on execution time and energy consumption. A computationally intensive task should be offloaded to the cloud node, while the execution performance and energy consumption on the cloud node are improved at the same time. So, it has to satisfy the following three conditions,

$$\min_{\{c\}} \{\alpha * \bar{T}_c + \beta * \bar{E}_c + \gamma * M * P_c\}$$

s.t.

$$T_m > t_1 + \frac{D + D^r}{B} + \frac{M}{C}, \quad (10)$$

$$E_m > (D_t + D_r) * E_t$$

where $\bar{T}_c$ is the normalization value of the execution time a task offloaded to a cloud node c, and $\bar{E}_c$ is the normalization value of the energy consumption a task offloaded to a cloud node c. M is the amount of tasks to be processed on the cloud node, P is the charges for processing unit data on remote cloud node.

$M*P$ represent the cost of offloading tasks. α, β and γ are the weights of execution time, energy consumption and offloading costs, and $\alpha+\beta+\gamma=1$. These parameters can help users to adjust their preference for execution time, energy consumption and offloading cost. The meaning of other parameters are the same as above.

In order to offload a task to cloud node to enhance user experience, the system agent collects information for all available candidate cloud node. Based on these information, find the most suitable cloud node. Algorithm 1 is the proposed task offloading policy. First, the system calculates the execution time T_m and energy consumption T_m of the task executed on mobile device, and the execution time *T(i)* and the energy consumption *E(i)* for every candidate cloud node are calculated. Then, filtering the candidate cloud node based on execution time and energy consumption (and distance for local cloud). All cloud nodes that do not meet the conditions will be excluded. Finally, the benefit function is computed for each candidate cloud node, and the cloud node with the minimum value will be the target cloud node for task offloading.

Algorithm 1 Task offloading policy

Input: The set of candidate cloud nodes, *cand_set*; The set of category for each cloud node, *cate*; The task need to be offloaded, *task.*

Output: The target cloud node, $node_t$

1: Calculate the execution time of the task on mobile device T_m

2: Calculate energy consumption of the task on mobile device E_m

3: $n1 \leftarrow cand\_set.size()$

4: **for** each $i=1$ to $n1$ **do**

5: **if** $cate(i)=local$ **then**

6: Calculate the distance $dis(m,c_i)$

7: **if** $dis(m,c_i)>s_i$ **then**

8: $cand\_set(i) \leftarrow \phi$

9: **end if**

10: **else**

11: Calculate the execution time for $node(i)$

$$T(i)=t_i+\frac{D+D^r}{B_i}+\frac{M}{C_i}$$

12: Calculate the energy consumption for $node(i)$

$$E(i)=(D_t+D^r)*e_i$$

13: **if** $T(i)>T_m$ or $E(i)>E_m$ **then**

14: $cand\_set(i) \leftarrow \phi$

15: **end if**

16: **end if**

17: **end for**

18: $n2 \leftarrow cand\_set.size()$

19: **for** each $i=1$ to $n2$ **do**

20: Normalize $T(i)$ and $E(i)$

$$\bar{T}(i)=\frac{T(i)}{\sum_{i=1}^{n}T(i)}$$

$$\bar{E}(i)=\frac{E(i)}{\sum_{i=1}^{n}E(i)}$$

21: Calculate the benefit function

$$F(i)=\alpha*\bar{T}(i)+\beta*\bar{E}(i)+\gamma*M*P(i)$$

22: **end for**

23: Find the minimum expense of the cloud node

24: **for** each $i=1$ to $n2$ **do**

25: **if** $F(i)=\min(F)$ **then**

26: $t=i$

27: **end if**

28: **end for**

29: **return** $cand\_set(t).ID$

IV. Experimental Simulation

In this section, we will verify the theoretical results of the proposed offloading policy. Our simulation experiment employs matlab 8.3 on an IBM server with an Inter Xeon E5-2670 eight-core 2.60 -GHz CPU and 32G of RAM.

The simulation platform consists of three modules, the information collection module, the offloading decision module and the offloading execution module. The information collection module is responsible for collecting information about cloud nodes (performance parameters, location and charges) and tasks (data size, result size). The offloading decision module is used to process the collected infor-

mation to select the optimal cloud node as the offloading target node. The offloading execution module is responsible for offloading the task to the target node and returning the result to the user after the task is completed. In this experiment, we randomly generate the remote cloud node, the local cloud node and the mobile task in the range of 100*100, the performance parameters and the size of the mobile task are randomly generated within a certain range. In the process of offloading decision, we set α=0.4, β=0.4, γ=0.2.

4.1 Performance metrics

The performance metrics we used are described below.

1) Average execution time (T_a) [12].

$$T_a = \frac{\sum_{i=1}^{n} t_i}{n}, \tag{11}$$

where n is the number of tasks, t_i is the execution time of task i. This metric mainly reflects the efficiency of task completion. The shorter the average execution time, the higher the efficiency of task allocation

2) Average energy consumption (E_a)[22].

$$E_a = \frac{\sum_{i=1}^{n} e_i}{n}, \tag{12}$$

where e_i is the energy consumption of task i. This metric reflects the energy consumption of the task offloading, the less energy consumption, the better the user experience.

4.2 Comparison methods

In order to validate the proposed offloading policy, we introduce three benchmark policies, namely, Mobile execution, Random Offloading Policy [26] and Dynamic Offloading [12]. Their work are as follows:

1) Mobile Device Execution (**MDE**): The mobile task will be executed on mobile devices, do not consider offloading policy.

2) Random Offloading Policy (**ROP**): Arriving tasks are randomly assigned to the cloud nodes, assuming that each cloud node has the same probability of being chosen.

3) Dynamic Offloading Policy (**DOP**): Dynamically compute the ability of each candidate cloud node, and the feasible cloud node that incurs smaller execution delay will be chosen.

The proposed offloading policy considers various factors, namely **VOP**.

4.3 Comparison results

In the simulation experiment, the mobile device randomly generates n tasks, and the attributes of tasks and cloud nodes are randomly generated. By controlling the number of task nodes and the number of cloud nodes, we observe the execution time and energy consumption under different offloading policies. we set the number of remote cloud nodes $n_r = 2$, and the number of local clouds $n_l = 10$. Observe the execution time and energy consumption under different offloading policies by adjusting the number of tasks nodes.

Figure 2 shows the effect of the number of tasks on the average execution time. As can be seen in the figure, with the increase of the number of tasks, the average execution time of each offloading strategy is increased, and the proposed VOP showed obvious advantages. Because the method MDE do not offload the task to a cloud node, so it can only perform tasks on the mobile device, the efficiency is

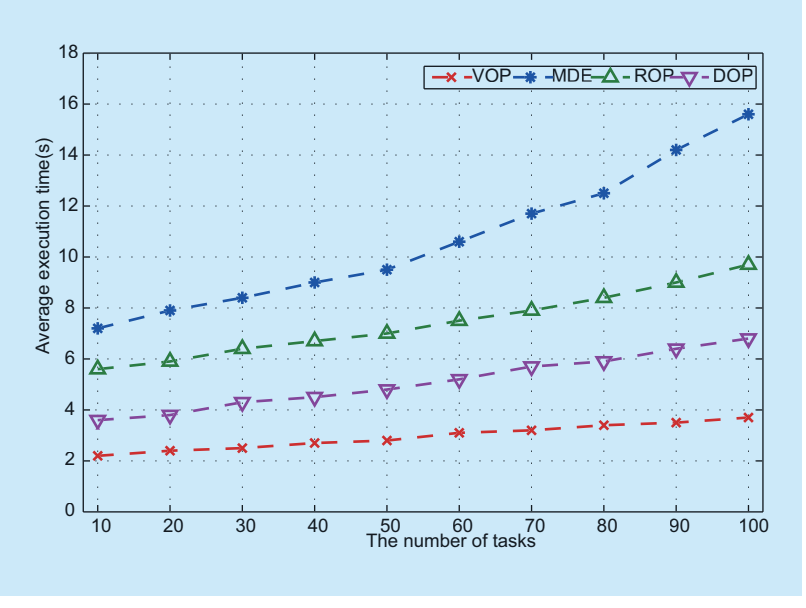


Fig. 2 *Average execution time of different number of tasks*

very low. That's why it's the worst. The method ROP randomly offload a task to a cloud node, which is relatively blind method. Sometimes there's a long wait, so it also doesn't work very well either. DOP is more scientific to select the target node, but the process is complex and time-consuming, so the results are not the best. Our proposed VOP consider various factors, and special emphasis on the cost of a variety of time. So its results are ideal.

The average energy consumption of different number of tasks is shown as figure 3. For every offloading policy, the average energy consumption is basically stable, and the performance of VOP is obviously superior. If all tasks executed on mobile device, the energy consumption must be very high, so the method MDE performs worst. ROP and DOP adopt offloading policy, so it can get better results. However, both the two methods do not take into account the energy consumption factor in the offloading policy, therefore, their performance is not the best.

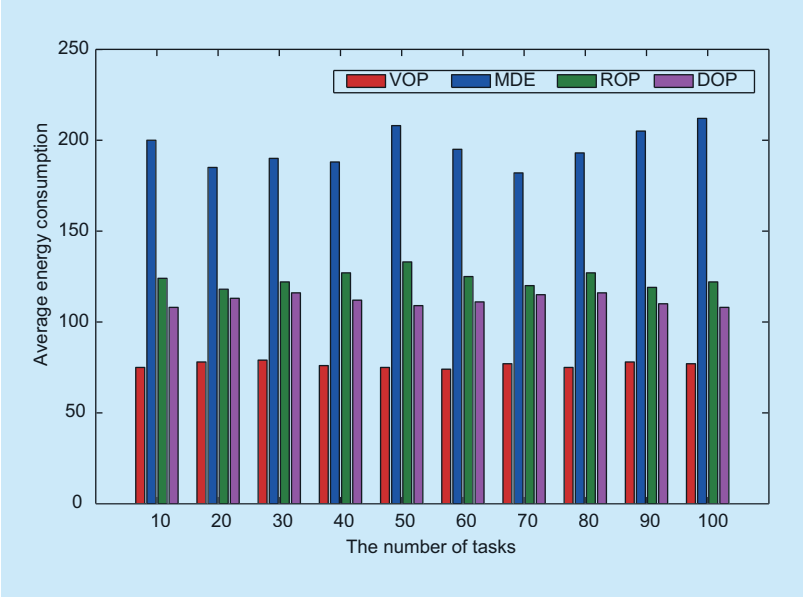


Fig. 3 *Average energy consumption of different number of tasks*

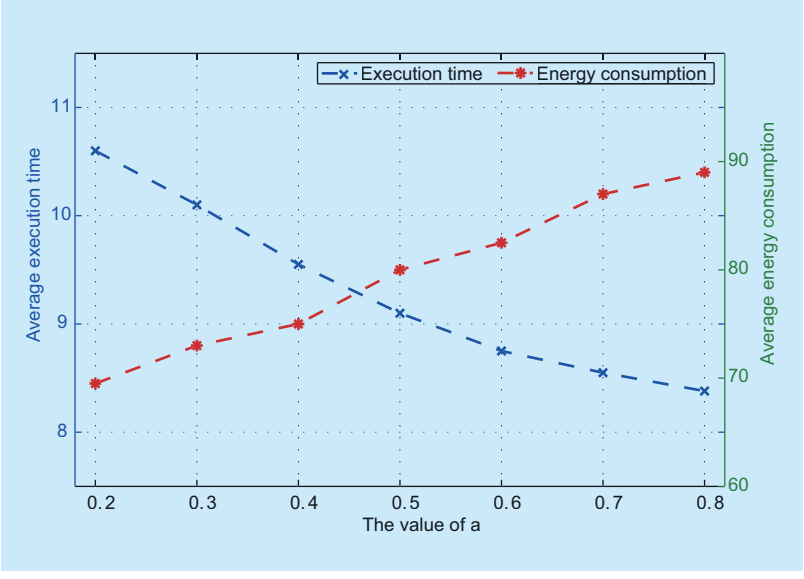


Fig. 4 *parameter analysis.*

4.4 parameters analyses

In order to analyze the influence of parameters α and β on the experimental results, We fix the values of γ=0.2 and change the values of α and β. Figure 4 shows the average execution time and average energy consumption with the change of α and β. Obviously, the average execution time is decreasing as the value of α increases. The greater the value of α, the higher the weight of the execution time in task offloading decision, and the program that facilitates the execution efficiency will be chosen.

Similarly, the greater the value of β, the more favorable it is for energy efficient alternatives. In summary, the parameters α, β and γ is the weight of execution time, energy consumption and charges respectively. Users can adjust different parameter values according to their preferences.

V. Conclusion

The fog computing is a new paradigm which attracts lots of attentions. Providing satisfactory computation performance is particularly challenging in the fog computing environment. In this paper, we propose a novel fog computing model and offloading policy which can effectively bring the fog computing power closer to the mobile user. The fog computing model consist of remote cloud nodes and local cloud nodes. And we propose a task offloading policy taking into account execution, energy

consumption and other expenses. The experimental results show the superiority of our approach in terms of execution time and energy consumption.

In this article, there are some other issues that do not involve, for example, dynamic offloading, virtual machine migration, etc. In future work, we will engage in these areas.

Acknowledgement

We gratefully acknowledge anonymous reviewers who read drafts and made many helpful suggestions. This work is supported by the NSFC (61602126) and the scientific and technological project of Henan province (162102210214).

References

[1] K Kumar, J Liu, Y Lu, et al., "A Survey of Computation Offloading for Mobile Systems," *Mobile Networks and Applications*, vol. 18, no. 1, 2013, pp. 129-140.

[2] S Wang, C Fan, CH Hsu, et al., "A Vertical Handoff Method via Self-Selection Decision Tree for Internet of Vehicles," *IEEE Systems Journal*, vol. 10, no. 3, 2016, pp. 1183-1192.

[3] A Zhou, S Wang, Z Zheng, et al., "On Cloud Service Reliability Enhancement with Optimal Resource Usage," *IEEE Transactions on Cloud Computing*, vol. 4, no. 4, 2016, PP. 452-466.

[4] Q Xia, W Liang, W Xu, "Throughput maximization for online request admissions in mobile cloudlets," *Proc. IEEE Local Computer Networks*, 2013, pp. 589-596.

[5] A Zhou, S Wang, B Cheng, et al., "Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization," *IEEE Transactions on Services Computing*, 2016.

[6] S Wang, A Zhou, CH Hsu, et al., "Provision of Data-Intensive Services Through Energy- and QoS-Aware Virtual Machine Placement in National Cloud Data Centers", *IEEE Transactions on Emerging Topics in Computing,* vol. 4, no. 2, 2016, pp. 290-300.

[7] J Liu, S Wang, A Zhou, et al., "Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability," *IEEE Transactions on Cloud Computing*, 2016.

[8] G Zhai, et al., "A computing resource adjustment mechanism for communication protocol processing in centralized radio access networks." *China Communications*, vol. 13, no. 12, 2016, pp. 79-89.

[9] D Huang, et al., "A Dynamic Offloading Algorithm for Mobile Computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, 2012, pp. 1991-1995.

[10] S Wang, Z Zheng, Z Wu, et al., "Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems," *IEEE Transactions on Services Computing*, vol. 8, no. 5, 2015, pp.755-767.

[11] J Liu, S Wang, A Zhou, et al., "Availability-aware Virtual Cluster Allocation in Bandwidth-Constrained Datacenters," *IEEE Transactions on Services Computing*, 2017.

[12] Y Mao, J Zhang, et al., "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no.12, 2016, PP. 3590-3605.

[13] Y Ma, S Wang, et al., "A Highly Accurate Prediction Algorithm for Unknown Web Service QoS Values," *IEEE Transactions on Services Computing*, vol. 9, no. 4, 2016, pp. 511-523.

[14] S Kosta, A Aucinas, P Hui, et al., "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading", *Proc. IEEE Infocom*, 2012, pp. 945-953.

[15] D Sabella, A Vaillant, et al., "Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, 2016, pp. 84-91.

[16] L Tong, Y Li, W Gao, "A hierarchical edge cloud architecture for mobile computing," *Proc. IEEE Infocom*, 2016, pp. 1-9.

[17] S Wang, L Huang, CH Hsu, et al., "Collaboration reputation for trustworthy Web service selection in social networks," *Journal of Computer & System Sciences*, vol. 82, no. 1, 2016, pp. 130-143.

[18] D Kelaidonis, A Rouskas, V Stavroulaki, et al. "A federated Edge Cloud-IoT architecture," *Proc. European Conference on Networks and Communications*, 2016, pp. 230-234.

[19] K Habak, M Ammar, et al., "Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge," *Proc. IEEE, International Conference on Cloud Computing.* 2015, pp. 9-16.

[20] S Abdelwahab, B Hamdaoui, M Guizani, et al., "Replisom: Disciplined Tiny Memory Replication for Massive IoT Devices in LTE Edge Cloud," *IEEE Internet of Things Journal,* vol. 3, no. 3, 2016, pp. 327-338.

[21] DB Hoang, L Chen, "Mobile Cloud for Assistive Healthcare (MoCAsH)," *Proc. IEEE Services Computing Conference,* 2011, pp. 325-332.

[22] H Wu, Y Sun, K Wolter, "Analysis of the Energy-Response Time Tradeoff for Delayed Mobile Cloud Offloading," *Acm Sigmetrics Performance Evaluation Review*, vol. 43, no. 2, 2015, pp. 33-35.

[23] S Wang, T Lei, L Zhang, et al., "Offloading mobile data traffic for QoS-aware service provision

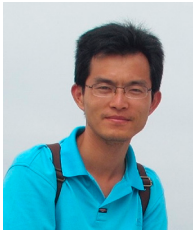
in vehicular cyber-physical systems," *Future Generation Computer Systems,* 2016, pp. 118-127.
[24] E Gelenbe, R Lent, "Energy–QoS Trade-Offs in Mobile Service Selection," *Future Internet*, vol. 5, no. 2, 2013, pp. 128-139.
[25] H Wu, K Wolter. "Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric," *proc. International Conference on PERFORMANCE Evaluation Methodologies and TOOLS*, 2014, pp. 90-97.
[26] H Wu, Q Wang, K Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," *proc. IEEE International Conference on Communications Workshops*, 2013, pp. 728-732.
[27] CA Chen, M Won, et al., "Energy-Efficient Fault-Tolerant Data Storage and Processing in Mobile Cloud," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, 2015, pp. 28-41.
[28] ME Khoda, MA Razzaque, A Almogren, et al., "Efficient Computation Offloading Decision in Mobile Cloud Computing over 5G Network," *Mobile Networks & Applications*, vol. 21, no. 5, 2016, pp. 1-16.

Biographies

Qiliang Zhu, received the B.S. degree and M.S. degree from North China University of Warter Resource and Electric Power, in 2001 and 2006, respectively. He is currently working toward the PhD degree in the Beijing University of Posts and Telecommunications. His research interests are in service computing, cloud computing and recommender systems.

BaoJiang Si, received the B.S. degree and M.S. degree from North China University of Warter Resource and Electric Power, in 1996 and 2006, respectively. His research interests are in Management of water conservancy and hydropower project, Information management of water conservancy , and water park.

Feifan Yang, received the BEng degrees from North China University of Water Resource and Electric Power, in 2015. He is currently working toward the Master degree in North China University of Water Resource and Electric Power since 2016. His research interests are in graphic image processing, virtual reality, and big data technology.



You Ma, is a research associate in the National Satellite Meteorological Center. His research interests are in big data analysis and recommender systems.