J Supercomput (2016) 72:3222–3235
DOI 10.1007/s11227-016-1704-0

Optimal mobile device selection for mobile cloud service providing

Ao Zhou[1] · Shangguang Wang[1] · Jinglin Li[1] · Qibo Sun[1] · Fangchun Yang[1]

Published online: 24 March 2016

Abstract With the rapid growth of the mobile devices and the emergence of cloud computing, mobile cloud computing has gained widespread interest. In mobile cloud computing, a large-scale collection of mobile devices cooperate with each other to provide a cloud service at the edge. However, the improper mobile device selection has a negative effect on the quality of service. Existing methods are difficult to solve the problem, because they do not take the status and the historical characteristics of the mobile devices into consideration. This paper introduces a device status-aware and stability-aware mobile device selection method. Firstly, a model is designed to store the status and the historical characteristics of each mobile device. Secondly, an optimized cloud model is employed to evaluate the stability of each mobile device. Lastly, an optimal mobile device searching algorithm is presented to select the optimal mobile device. We provide an extensive evaluation of our method. The results show that our method can increase the quality of mobile cloud service compared with the traditional method.

✉ Shangguang Wang
sgwang@bupt.edu.cn

Ao Zhou
aozhou@bupt.edu.cn

Jinglin Li
jlli@bupt.edu.cn

Qibo Sun
qbsun@bupt.edu.cn

Fangchun Yang
fcyang@bupt.edu.cn

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, People's Republic of China

Keywords Mobile device cloud · Mobile cloud computing · Mobile cloud service · Mobile device selection · Cloud model

1 Introduction

Mobile device cloud [1,2] has received widespread concern recently. Mobile devices, such as laptops, tablets and smartphones, have grown increasingly powerful. With the advent of cloud computing [3–7] and the rapid growth in mobile devices [8,9], many researchers try to orchestrate a large-scale collection of mobile devices to provide a cloud service at the edge. To save the execution time and the energy of some low-energy mobile devices, the researchers try to off-load some computations to nearby powerful mobile devices. Owners who are willing to share their computation can receive financial gains.

After receiving a task from a mobile device (the sender, or an off-loader), the controller needs to assign the task to one or more mobile devices (the receiver, or and off-loadees) in the mobile device cloud. A core problem is which mobile device to select. Each device has an available duration. If a device departs before completing a task, the task should be restarted from the beginning. The quality of service is negatively affected. The method proposed by [10] considers how to maximize the lifetime of the mobile device cloud. The off-loadee selection target is obtained by proposing a method to balance the power across a set of collaborative mobile devices. The method proposed by [11] assumes that the controller perfectly knows the departure time of each device. However, that is not always the truth. Some dishonest owner may leave before the declared departure time. A device may shut down because of the use up of power.

Owing to these drawbacks, a device status-aware and stability-aware mobile device selection method for mobile cloud service providing is proposed in this paper. Some devices may join the same mobile device cloud many times. The students have the same class every day, and go to the same library every night. People are used to going to the same coffee shop. The mobile device cloud controller can save the information for future use. The mobile device clouds that belong to the organization can also share the device historical information with each other. Based on the historical information, our method includes three steps. In step 1, we design a storage model to store the status and the historical characteristics of each mobile device. In step 2, we extend the cloud model to a weighted cloud model and employ the weighted cloud model to evaluate the stability of each mobile device. Based on the evaluation results from step 2, an optimal mobile device searching algorithm is presented in step 3 to select the optimal mobile device. We provide an extensive evaluation of our approach. The results show that our method can increase the quality of cloud mobile service compared with the traditional methods.

The rest of this paper is organized as follows. Section 2 presents the related work, followed by the system model in Sect. 3. Section 4 illustrates the detail of our method. The simulation results are provided in Sect. 5. Finally, we conclude the paper in Sect. 6.

2 Related work

There are plenty of researches on mobile device cloud computing in literature. Only some notable works are reviewed due to space limitation.

To save the execution time and the energy of mobile devices, several works studied how to off-load parts of mobile applications to the cloud. Off-loading to cloud is illustrated in Fig. 1.

The number of mobile services has grown rapidly and the execution of mobile applications consume too much time and energy. Therefore, CloneCloud is proposed [1]. In CloneCloud, a mobile application is partitioned at a fine granularity by adopting a combination of static analysis and dynamic profiling. When the application is executing, some partitions are migrated as a thread to the cloud. When the partitions have finished, the thread is migrated back to the mobile device.

Mobile services usually are real time and require quick response. However, too much time is lost because of long setup time and data transfer latency. To attack the problem, COSMOS [12] was proposed. COSMOS can reduce the lost time by allocating and scheduling the off-loading requests effectively. In addition, to overcome the network connectivity problem, the off-loading decision is made in a risk-controlled manner.

Although the above-mentioned works can effectively save the execution time and the energy of mobile devices, these methods migrate a large amount of execution contexts to the cloud. To solve the problem effectively, a novel method was proposed by [13]. The method tries to migrate the least execution context to remote cloud. The target is achieved by predicting the memory contexts that the application may access. Only the relevant contexts are migrated to the cloud.

Different from the above-mentioned works, other researchers studied how to off-load parts of mobile applications to the other local mobile devices. Off-loading to other mobile devices is illustrated in Fig. 2.

Recently, some types of mobile devices are increasingly becoming more and more powerful, while others have very limited processing capabilities. For example, the processing capabilities of the wearable computing devices are very limited. This condition allows the mobile devices to save time and energy by off-loading to local mobile

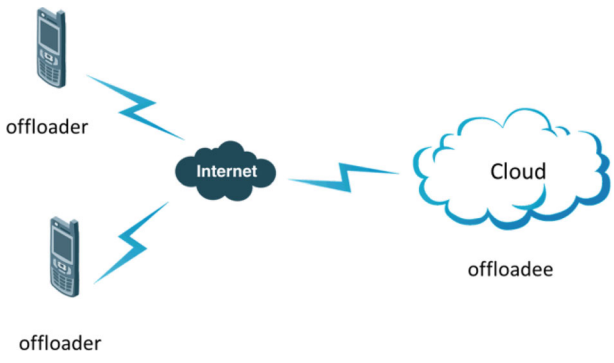

Fig. 1 Off-loading to cloud

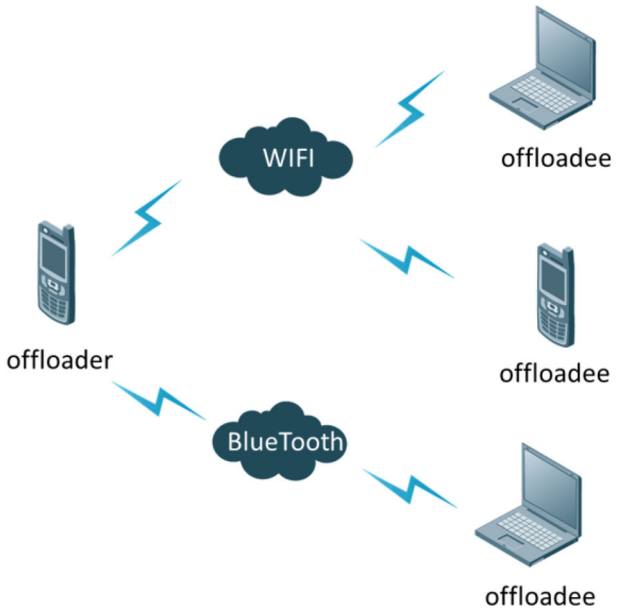

Fig. 2 Off-loading to other mobile devices

devices. Based on the condition, the authors in [14] conducted a set of experiments to investigate the gain in time and energy. They also carry out experiments to test what kind of services is more suitable to be off-loaded to the mobile device cloud. The work considered how to maximize the lifetime of the mobile device cloud. The target is obtained by proposing a method to balance the power across a set of collaborative mobile devices.

A mobile device cloud control system named FemtoClouds is presented by [11]. Basic functions, such as capability estimation, user profiling, execution prediction, presence time prediction and task scheduling, can be performed by the system.

However, the above works assume that the departure time of each device is known. They cannot achieve the optimal effects in mobile device selection for ignoring the status and the historical characteristics of each devices. For example, some devices are prone to leaving before declared departure time, or a device shows a decrease in credit. Each device has an available duration. If a device departs before completing a task, the task should be restarted from the beginning. The quality of service is negatively affected. Our method can solve the problem by taking the status and the historical characteristics of the mobile devices into consideration.

3 System model and expression

We consider the mobile device cloud which is similar to [11]. All mobile device owners have installed the mobile device cloud client service on their mobile devices. All mobile device owners are willing to share their computation, but they can receive

Table 1 List of symbols used

Symbol	Description
T	A task to be assigned to an off-loadee
D	A mobile device belonging to the mobile device cloud
ET(T, D)	The estimated execution time if T is assigned to D
WT(T, D)	The estimated waiting time if T is assigned to D
DL(T)	The remaining time to the deadline of T
DT(D)	The declared departure time of D
AT(D)	The actual departure time of D
t	A time point
n	The number of mobile devices in the mobile device cloud
ST(D)	The estimated shutdown time of D because the battery is flat. The value tends to infinity when the mobile device is in a charging state
Ex	Expected value
En	Entropy
Ne	Hyper-entropy
$\mathrm{WE}x_i$	Weighted expected value of x_i
$\mathrm{WE}x_{i-}$	Weighted expected value of x_{i-}
WS^2_{i-}	Weighted sample variance of x_{i-}
WEn_{i-}	Weighted entropy of x_{i-}
WHe_{i-}	Weighted hyper-entropy of x_{i-}

financial gains. All mobile devices that are willing to join a special mobile device cloud are managed by a mobile device cloud controller. In addition, many other functions are provided by the controller: mobile device discovery, task scheduling, user profiling, execution time estimation, presence time prediction, and so on. The task size refers to the length of the execution time in this paper. The task size can be determined by the presence time prediction module [15]. Every device owner declares the departure time after joining the mobile device cloud. However, some dishonest devices may leave before the declared departure time. When a device departs before the assigned task has been completed, the task should be rescheduled to another device and restarted from the beginning. "Depart" means that the device leaves the mobile device cloud and stops sharing its computation.

In addition, the symbols in Table 1 will be used throughout the paper.

4 Optimal mobile device selection

Figure 3 illustrates the framework of our method. Our optimal mobile device selection method (OMDS) is divided into three phases: mobile device information storage (in Sect. 4.1), mobile device stability evaluation (in Sect. 4.2), and optimal mobile device searching (in Sect. 4.3).

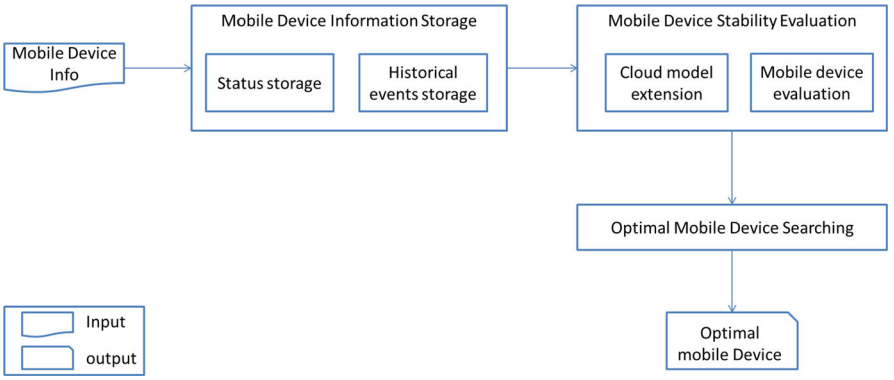

Fig. 3 Framework of our method

4.1 Mobile device information storage

Mobile devices belonging to a mobile device cloud need to exchange application data and other information with the controller. The controller stores the information of each mobile device on a local server. In the mobile device selection stage, the controller can evaluate the mobile devices based on the information. The format of the status data is as follows:

$$\mathrm{SD} = \{\mathrm{eID}, \mathrm{Status}, \mathrm{Time}\}, \tag{1}$$

where eID denotes the ID of the mobile device. Status denotes the status of the mobile device. When the value of Status is 0, the mobile device is in a charging state. Otherwise, the value of Status is 1. Time denotes current time.

The format of the historical event data is as follows:

$$\mathrm{ED} = \{\mathrm{eID}, \mathrm{DT}, \mathrm{AT}, \mathrm{Source}, \mathrm{Time}\}, \tag{2}$$

where eID denotes the ID of the mobile device, DT the declared departure time of the mobile device, AT the actual departure time of the mobile device and Source the cause of the departure. When the value of Source is 0, the mobile device departs because the power is used up. When the value of Source is 1, the mobile device departs because network connection. When the value of Source is 2, the mobile device departed for the departure of its owner. Time denotes the current time.

4.2 Mobile device stability evaluation

We evaluate the stability of the mobile device based on the mobile device information. In this paper, stability denotes that the actual departure time is the same as or even longer than the declared departure time.

We adopt the cloud model to [16–18] evaluate the stability of each mobile device. In the cloud model, there is a group of cloud drops in the universe, and each cloud

drop has a numerical value. The overall distribution characteristics of the cloud drops are reflected by three numerical characteristics: expected value (Ex), entropy (En) and hyper-entropy (He). Ex denotes the distance to the center of the cloud gravity. En is used to measure the coverage of the cloud drops, and He is used to measure the dispersion of the cloud drops. In other words, He can be considered as the entropy of En. Ex, En and He are calculated by the following:

$$\mathrm{Ex} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3}$$

$$S^2 = \frac{1}{m-1}\sum_{i=1}^{m} (x_i - \mathrm{Ex})^2 \tag{4}$$

$$\mathrm{En} = \frac{\sqrt{\pi/2}}{m}\sum_{i=1}^{m} |x_i - \mathrm{Ex}| \tag{5}$$

$$\mathrm{He} = \sqrt{S^2 - \mathrm{En}^2}. \tag{6}$$

In our context, each record denotes a cloud drop. In mobile device selection, only the mobile device that leaves before the declared departure time will affect the quality of the mobile service. The two types of numerical value of a cloud drop are calculated by the following:

$$x_i^k = \mathrm{DT}_i^k - \mathrm{AT}_i^k \tag{7}$$

$$x_{i-}^k = \begin{cases} \mathrm{DT}_i^k - \mathrm{AT}_i^k, & \mathrm{DT}_i^k - \mathrm{AT}_i^k > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

where k denotes the kth record of the device D_i. When there is only one drop, we think the value of En and He is 0.

Based on the cloud model, the group with smaller En and He is more stable. However, a dishonest owner may start to keep his promise because of a good incentive strategy. Therefore, the recent characteristic data are more important than that saved a long time ago. To attack this problem, we design a weighted cloud model. The three numerical characteristics in our weighted cloud model are calculated by the following:

$$\mathrm{WE}x_i = \frac{1}{m}\sum_{k=1}^{m} w_i^k x_i^k \tag{9}$$

$$\mathrm{WS}_i^2 = \frac{1}{m-1}\sum_{k=1}^{m} \left(w_i^k \left(x_i^k - \mathrm{Ex}_i\right)\right)^2 \tag{10}$$

$$\mathrm{WEn}_i = \frac{\sqrt{\pi/2}}{m}\sum_{k=1}^{m} w_i^k \left|x_i^k - \mathrm{Ex}_i\right| \tag{11}$$

$$\mathrm{WHe}_i = \sqrt{\mathrm{WS}_i^2 - \mathrm{WEn}_i^2} \tag{12}$$

Algorithm 1: Backward cloud generator

Input: all ED record of D_i

Output: the numerical characteristics of each mobile device

1 Calculate the weighted expected value of x_i by using:

2 $$WEx_i = \frac{1}{m}\sum_{k=1}^{m} w_i^k x_i^k ;$$

3 Calculate the weighted expected value of x_{i-} by using:

4 $$WEx_{i-} = \frac{1}{m}\sum_{k=1}^{m} w_{i-}^k x_{i-}^k ;$$

5 Calculate the sample variance:

6 $$WS_{i-}^2 = \frac{1}{m-1}\sum_{k=1}^{m} (w_i^k (x_{i-}^k - Ex_{i-}))^2;$$

7 The Entropy of the mobile devices can be calculated by:

8 $$WEn_{i-} = \frac{\sqrt{\pi/2}}{m}\sum_{k=1}^{m} w_i^k |(x_{i-}^k - Ex_{i-})|;$$

9 Finally, the Hyper-Entropy of each mobile device can be obtained by:

10 $$WHe_{i-} = \sqrt{WS_{i-}^2 - WEn_{i-}^2};$$

11 **return** the numerical characteristics;

$$w_i^k = \frac{k}{1+2+\cdots+m} = \frac{k}{m(m+1)/2}, \tag{13}$$

where m denotes the record number of the device D_i, and k is the sequence number of a record. The sequence number of the oldest record is 1.

We adopt the backward cloud generator to evaluate the stability of each mobile device. The detail of the backward cloud generator is illustrated in algorithm 1.

The time complexity of the backward cloud generator is $O(n * m)$. n denotes the number of devices, and m denotes the record number of the device. We design an optimal mobile device searching algorithm based on the numerical characteristics of the cloud model. The details of our optimal mobile device searching algorithm is illustrated in the next section.

4.3 Optimal mobile device searching

Algorithm 2 describes our optimal mobile device searching algorithm. The algorithm traverses the mobile device list and makes schedule decision for the submitted task using the following steps:

Step 1 Calculate the difference between the presence time and the task size for each device. Sort the devices based on the results (line 1).

Step 2 Remove all improper devices from the list. If the difference between the presence time and the task size is smaller than 0, the device cannot complete the task on time. If the difference between the remaining time and the estimated execution time is smaller than 0, the device cannot complete the task before the deadline. If the weight expected value is smaller than 0, there is a very large chance that the owner will depart before the declared departure time (lines 2 to 7).

Algorithm 2: Optimal mobile device searching

Input: all mobile devices DVs

Output: all good virtual machines vector<vms> $DVms$

```
1  sort DVs by min(ST,DT) asc ;
2  for each element DVs[i] in DVs do
3      t = min(ST-currentTime -ET-WT,DT-currentTime
       -ET-WT);
4      if t< 0 || DL < ET || WEx_i < 0 then
5          remove DVs[i];
6      end
7  end
8  for j = 0;j < DVS.size() j = j+Δ do
9      clear tmpList;
10     for k = j;k ≤ j+Δ;k = k+1 do
11         add DVs[k] to tmpList ;
12     end
13     if tmpList.size() = 0 then
14         continue;
15     end
16     for k = 1;k < tmpList.size();k = k+1 do
17         for i = k;i < tmpList.size();i = i+1 do
18             if tmpList[i].WEx_i− < tmpList[i+1].WEx_(i+1)−
               then
19                 continue;
20             end
21             else if tmpList[i].WEx_i− =
               tmpList[i+1].WEx_(i+1)− then
22                 if tmpList[i].WEn_i− < tmpList[i+1].WEn_(i+1)−
                   then
23                     continue;
24                 end
25                 else if tmpList[i].WEn_i− =
                   tmpList[i+1].WEn_(i+1)− then
26                     if tmpList[i].WHe_i− <
                       tmpList[i+1].WHe_(i+1)− then
27                         continue;
28                     end
29                 end
30             end
31             exchange(tmpList[i], tmpList[i+1]);
32         end
33     end
34     break ;
35 end
36 return tmpList[0];
```

Step 3 Select a small group of candidates from the list. We should make full use of the time pieces (lines 8 to 12).

Step 4 If there is no candidate in the current interval, we increase the searching range and go back to step 3 (lines 13 to 15).

Step 5 Traverse the *tmpList* that stores all candidates. If all nodes in the list have been visited, go to step 10 (lines 16 to 17).

Step 6 If the weighted Ex of a device is larger than the device behind it, exchange the place of the two devices. Go to step 5 (lines 18 to 20).

Step 7 Otherwise, if the weighted En of a device is larger than the device behind it, exchange the place of the two devices. Go to step 5 (lines 22 to 24).

Step 8 Otherwise, if the weighted He of a device is larger than the device behind it, exchange the place of the two devices. Go to step 5 (lines 26 to 28).

Step 9: Exchange the content of *tmpList[l]* and *tmpList[l+1]* (line 31).

Step 10: Return the head of the *tmpList* (line 36).

Therefore, based on the preceding steps, we obtain an optimal mobile device that is honest and prone to leaving after the declared time. The time complexity of the algorithm is $O(n^2)$. n denotes the number of devices.

5 Simulation results

To verify the effectiveness of our optimal mobile device selection method, we implement our method in JAVA and conduct experiments on it. Section 5.1 illustrates our experimental setting. We then discuss the experimental results and present the advantages of our method in Sect. 5.2.

5.1 Experimental setting

We conduct our experimental results from a PC with an Intel Core 2 2.0GHz processor and 4.0GB of RAM. The machine is run on Windows 7, Matlab R2012a and Java 1.6.0. We compare our approach with random device selection RDS [11]. In RDS, the stability of the mobile devices is ignored. RDS considers whether the mobile device can complete the task before the declared departure time. The task is assigned to the mobile device that enables completing the task earlier. The methods are evaluated by using the following metrics:

1. Total execution time, which can be calculated as follows:

$$t_{\text{total}} = \sum_{i=1}^{n} (t_{\text{finish}}(T_i) - t_{\text{submitted}}(T_i)), \tag{14}$$

where t_{finish} denotes the completion time of T_i, and $t_{\text{submitted}}$ denotes the submission time of T_i.

2. Total number of reassignment times, which can be calculated as follows:

$$N_{\text{total}} = \sum_{i=1}^{n} (N_{\text{reassignment}}(T_i)), \tag{15}$$

where $N_{\text{reassignment}}$ denotes the reassignment frequency of T_i for the departure of the mobile devices.

5.2 Experimental results

5.2.1 Results on total execution time

In this section, we study the experimental results on total execution time. There are 1000 mobile devices in the mobile device cloud, and the presence time of each device is between 30 and 100 min. The task size is between 20 to 60 min. Figure 4 illustrates the total execution time when the rate of dishonest mobile device increases from 0.02 to 0.1. Figure 5 illustrates the total execution time when the number of tasks increases from 600 to 900. The figures show that:

- The total execution time of OMDS is less than RDS. Our method shows an average 1.2 % decrease in the total execution time. Because we take the status and stability of the mobile devices into consideration, we can avoid selecting the dishonest devices. There is a lower chance that the task execution process is interrupted. Therefore, our method consumes less total execution time.
- In RDS, total execution time is affected when the rate of dishonest device varies. Total execution time increases with an increase in the rate of dishonest devices. But our method is almost not affected by the rate of dishonest devices. When the rate of dishonest devices increases, there is a higher chance that a task is assigned to a dishonest device.

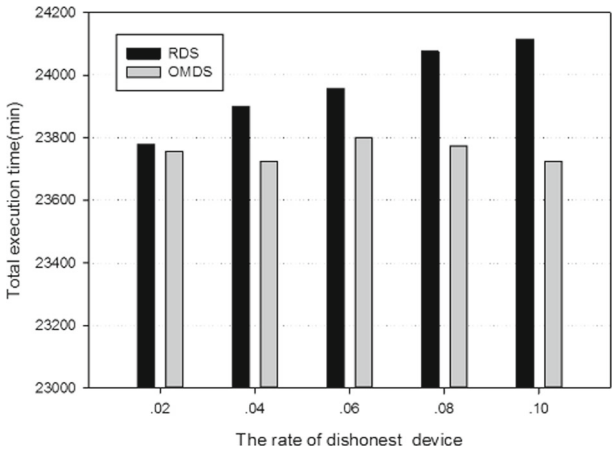

Fig. 4 Total execution time under different rates of dishonest devices

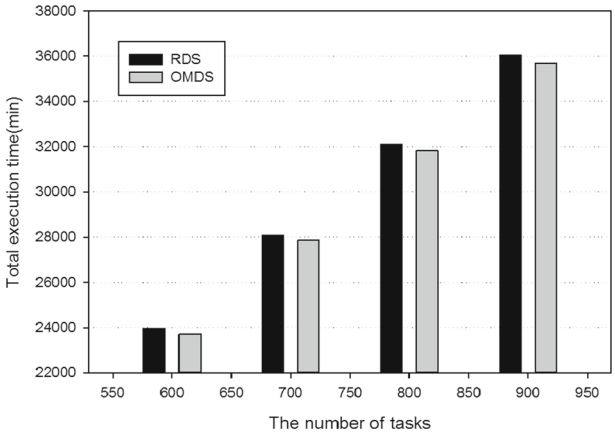

Fig. 5 Total execution time under different number of tasks

- In RDS, the total execution time is affected when the number of tasks varies. The total execution time increases with an increase in the number of tasks. But our method is almost not affected by the number of tasks. There is a higher chance that a task is assigned to a dishonest device when the number of tasks increases.

We will show the results of the total number of reassignment times in the next section.

5.2.2 Results on frequency of reassignment

In this section, we study the experimental results on frequency of reassignment. There are 1000 mobile devices in the mobile device cloud, and the presence time of each device is between 30 and 100 min. The task size is between 20 and 60 min. Figure 6 illustrates the frequency of reassignment when the rate of dishonest mobile device increases from 0.02 to 0.1. Figure 7 illustrates the frequency of reassignment when the number of tasks increases from 600 to 900. The figures show that:

- Reassignment frequency of OMDS is less than RDS. Our method shows an average 80 % decrease in the reassignment frequency. Considering the status and the stability of the mobile devices, we can avoid selecting dishonest devices. There is a lower chance that the task execution is interrupted because of the departure of mobile devices.
- In RDS, the reassignment frequency is affected when the rate of dishonest devices varies. The reassignment frequency increases with an increase in the rate of dishonest devices. But our method is almost not affected by the rate of dishonest devices. When the rate of dishonest devices increases, there is a higher chance that a task is assigned to a dishonest device.
- In RDS, the reassignment frequency is affected when the number of tasks varies. The total number of reassignment times increases with an increase in the number of tasks. But the total number of reassignment times of our method is almost not affected by the number of tasks. When the number of tasks increases, there is a

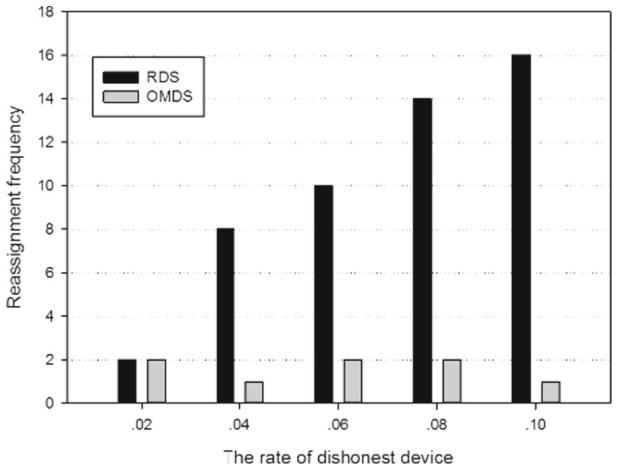

Fig. 6 Reassignment frequency under different rate of dishonest device

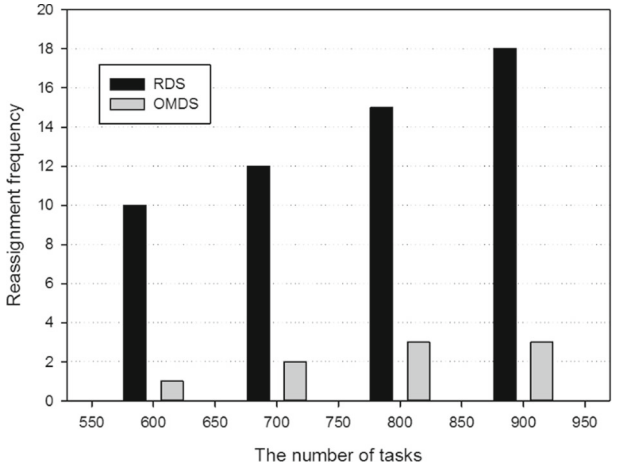

Fig. 7 Reassignment frequency under different number of tasks

higher chance that a task is assigned to a dishonest device. The departure of the dishonest device may result in the interruption of tasks.

6 Conclusions

To assign a task to the optimal mobile device, a status-aware and stability-aware mobile device selection method is proposed in this paper. Firstly, the status storage model and the historical characteristics storage model are designed. Secondly, we present a weighted cloud model and employ the weighted cloud model to evaluate the stability of each mobile device. Lastly, we propose an optimal mobile device searching algorithm

based on the evaluation results from the former step. The experimental results show the effectiveness of our optimal mobile device selection method.

It remains a task for future works to investigate how to save the network resource of a mobile device cloud. In addition, we will try to make time–energy tradeoff by transferring the context.

Acknowledgments This work was supported by NSFC (61272521), NSFC (61472047), NSFC (61571066), and "the Fundamental Research Funds for the Central Universities".

References

1. Chun B-G, Ihm S, Maniatis P, Naik M, Patti A (2011) Clonecloud: elastic execution between mobile device and cloud In: Proceedings of the sixth conference on computer systems, Salzburg, pp 301–314
2. Kao Y-H, Krishnamachari B, Ra M-R, Bai F (2015) Hermes: latency optimal task assignment for resource-constrained mobile computing. In: INFOCOM, Hong Kong, pp 1–9
3. Vouk MA (2008) Cloud computing-issues, research and implementations, CIT. J Comput Inf Technol 16(4):235–246
4. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I (2010) A view of cloud computing. Commun ACM 53(4):50–58
5. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
6. Dillon T, Wu C, Chang E (2010) Cloud computing: issues and challenges. In: 2010 24th IEEE international conference on advanced information networking and applications (AINA), Perth, pp 27–33
7. Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I (2009) Above the clouds: a Berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, vol 28, p 13
8. Li X, Wang X, Zhu C, Cai W, Leung V (2015) Caching-as-a-service: virtual caching framework in the cloud-based mobile networks. In: 2015 IEEE conference on computer communications workshops (INFOCOM WKSHPS), Hong Kong, pp 372–377
9. Shao J, Lu R, Lin X (2015) Fine-grained data sharing in cloud computing for mobile devices. In: 2015 IEEE conference on computer communications (INFOCOM), Hong Kong, pp 2677–2685
10. Cuervo E, Balasubramanian A, Cho D-k, Wolman A, Saroiu S, Chandra R, Bahl P (2010) MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th international conference on mobile systems, applications, and services, San Francisco, pp 49–62
11. Habak K, Ammar M, Harras KA, Zegura E (2015) FemtoClouds: leveraging mobile devices to provide cloud service at the edge. In: IEEE Cloud, New York, pp 1–8
12. Shi C, Habak K, Pandurangan P, Ammar M, Naik M, Zegura E (2014) COSMOS: computation offloading as a service for mobile devices. In: Proceedings of the 15th ACM international symposium on mobile ad hoc networking and computing, Philadelphia, pp 287–296
13. Li Y, Gao W (2015) Code offload with least context migration in the mobile cloud. In: 2015 IEEE conference on computer communications (INFOCOM), Hong Kong, pp 1876–1884
14. Mtibaa A, Harras K, Fahim A (2013) Towards computational offloading in mobile device clouds. In: 2013 IEEE 5th international conference on cloud computing technology and science (CloudCom), Bristol, pp 331–338
15. Kwon Y, Lee S, Yi H, Kwon D, Yang S, Chun B-G, Huang L, Maniatis P, Naik M, Paek Y (2013) Mantis: automatic performance prediction for smartphone applications. In: Proceedings of the 2013 USENIX conference on annual technical conference, SAN JOSE, pp 297–308
16. Li D, Liu C, Gan W (2009) A new cognitive model: cloud model. Int J Intell Syst 24(3):357–375
17. Wang S, Zheng Z, Sun Q, Zou H, Yang F (2011) Cloud model for service selection. In: 2011 IEEE conference on computer communications workshops (INFOCOM WKSHPS), Shanghai, pp 666–671
18. Wang S, Li D, Shi W, Li D, Wang X (2003) Cloud model-based spatial data mining. Geogr Inf Sci 9(1–2):60–70