

Computation Offloading Strategy for Multi User Mobile Data Streaming Applications

Wei Liu, WanJia Gong, Wei Du, Chengming Zou

School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, China
Email: wliu@whut.edu.cn

Abstract— In the mobile cloud environment, computation offloading provides a feasible solution for mobile intelligent devices to run those complex applications that require a large amount of computing resources. But most of the existing computation offloading algorithms are based on a single user's point of view and assume that the cloud resources are sufficient at any time. The scenario considered in this paper is multi-user with limited cloud resources, when there are a lot of components to be migrated to the cloud. But the cloud will not be able to handle all components in real-time, hence the need to find a solution to effectively reduce the energy consumption of mobile devices while simultaneously ensuring the performance of each user's mobile application. As an effort to solve this problem, a four-stage multi-user offline computation offloading algorithm (FMOCO) is proposed in this paper. FMOCO takes into account the offloading requests submitted by all users over a specific period of time and derives the best possible offloading strategy based on each individual user's computing power, bandwidth, as well as available cloud resources. Experimental results show that FMOCO can consider the performance of the applications as well as power consumption of all users under the condition of limited cloud resources.

Keywords—Limited Cloud Resources, Multi User, Computation Offloading, Application Performance, Energy Consumption

I. INTRODUCTION

As of December 2015, the number of Internet users in China has reached 688 million, of which 90.1% use mobile phones and 31.5% use tablet computers [1]. Mobile devices, such as smartphone and tablets, are equipped with higher resolution camera, more types of sensors, better portability and richer set of functionality. As these devices become popular with a growing number of users, data streaming applications, such as face recognition [2], mobile augmented reality [3], mobile games [4] and mobile social network [4] have emerged. Such applications constantly obtain input data through sensors or cameras, process those data and then return the results to users. A streaming application typically has three characteristics: delay sensitive, computation intensive and high power consumption during execution [5]. The hardware resources of smartphones such as computing power, battery capacity, memory size and bandwidth are limited, and the increase in battery capacity is far less than the speed of power consumption [6], all combined to make it very challenging to meet the user's

QoS (Quality of Service) requirements when running increasingly sophisticated mobile streaming applications.

The cloud usually has enough computing and storage resources, and the current domestic 3G, 4G network has basically achieved full coverage, plus there are also a lot of WIFI hotspots, mobile devices can easily and quickly have access to cloud resources. With computation offloading technology, some or all components of data streaming applications can be migrated to the cloud, leveraging powerful computing power and storage capacity provided by cloud to reduce the execution time of components and computational load of mobile terminals, allowing mobile users to run those applications that need a large amount of computing resources which are otherwise unavailable in the mobile devices [7-8].

Although the computing power and storage capacity provided by cloud is more powerful than a mobile device, the number of cloud servers rented by mobile application providers is always limited for the sake of cost [9]. When the increase in the number of users outpaces the available cloud resources, the cloud will not be able to handle all offloaded components in time, resulting in competition for cloud resources among individual users, in which case a user's migration strategy will affect others, consequently some components may not be executed at the desired pace.

In this paper, Section 2 describes the current researches. Section 3 builds the system model, including the average response time model and average energy consumption model. Details of the algorithm are described in Section 4. Simulation experiment is carried out in Section 5. Section 6 discusses our conclusion and prospects for the future.

II. RELATED WORKS

Computation offloading [10-12] makes it possible for resource-constrained mobile devices to run computation intensive applications. It refers to the process of offloading a subset of components of a data streaming application from running locally to the cloud.

Computation offloading can usually save a mobile device's energy consumption and improve the performance of mobile applications. However, its effectiveness depends on a number of variables, such as the user's location, network bandwidth, available computing power and load of cloud servers. For instance, the location of the mobile device may change frequently, which can lead to the constant change of bandwidth.

Rich Wolski [13] studied the influence of bandwidth on the offloading decision, estimated the time for both offloading and no offloading by predicting the current bandwidth, and concluded that computation offloading occurs only when it can save response time. Another common scenario is that network connection may be unstable and in worst case the connection could be dropped. Liu Xing et al [14] put forward an energy-saving TOSOC algorithm aiming at service delay and connection disconnection problem in the process of computation offloading.

Users are often very sensitive to delays. Usually the distance between the mobile device and the cloud is far, therefore, offloading tasks to the cloud may cause a large transmission delay, which then negatively affects the user experience. Cloudlet is relatively close to the users and is often connected with mobile terminals through WIFI, hence migrating tasks to cloudlet rather than the cloud can greatly reduce the transmission delay. To improve the performance of interactive streaming applications in the presence of network latency and bandwidth, Weiqing Liu et al [15] used multiple cloudlet to balance throughput and energy consumption. M.Reza Rahimi et al [16] studied the three tier computation offloading architecture that consists of mobile device, cloudlet and cloud, and proposed a heuristic algorithm MuSIC that can meet the application requirements of QoS while taking into account of user's mobility.

Wireless network interface will consume tail energy during data transmission [17-19], which accounts for approximately 60% of the entire transmission energy consumption [20], so tail energy consumption can't be ignored. Guojun Wu [21] proposed a bundled migration strategy, which will simultaneously migrate multiple tasks that meet the specific constraints to the cloud, reducing tail energy by reducing the number of states of high energy consumption to low energy transitions. Zhang Di [22] used the tail time to deal with the requests to effectively reduce energy consumption, but its research objective focused on delay tolerant applications and hence is not suitable for data streaming applications with strict real-time requirement.

However, the approaches mentioned above are all from single user's point of view, with the assumption that the cloud resources are infinite and always be able to deal with the computation offloading requests submitted by all users in a timely manner. Yang Lei and Cao Jiannong et al [9] implemented a multi-user offloading algorithm called searchadjust based on limited cloud resources, the goal is to minimize the average delay of mobile applications for all users. To solve the problem that the computation offloading decisions of multiple users may produce interference under multi-channel condition, Xu Chen et al [5] adopted the method of game theory to make an optimal decision for each user, so as to achieve the balance of energy consumption and response time, but his scope is limited to independent tasks, therefore his approach does not apply to dependent tasks.

III. SYSTEM MODEL

Data streaming applications usually consist of several components, and there are dependences among components, that is, for components that have data transmission relationship, the successor component must wait until its predecessor components have been executed and all of their outputs can be obtained. So it is usually represented by a DAG graph (directed acyclic graph, $G = (V, E)$), in which the vertex represents the component, the edge represents the predecessor successor relationship. Each component C_j in V can be represented as a three tuple- $C_j = \{s_j, x_j, w_j\}$, s_j represents the component number, x_j represents the execution location, and w_j represents the execution time.

The following table lists the parameters of the system model used in this section and the corresponding meaning.

TABLE 1. DEFINITION OF RELEVANT PARAMETERS

Parameter	Definition
(i, j)	component j of user i
t_i	offloading request submission time of user i
n	number of component
$w_{i,j}$	execution time of component (i, j) at the mobile device
$\Gamma_{i,j}$	execution time of component (i, j) in the cloud
λ	number of users
$ST_{i,j}$	start execution time of component (i, j)
$ET_{i,j}$	completion time of component (i, j)
P_{i,cpu_comp}	CPU power of user i in running state
P_{i,cpu_idle}	CPU power of user i in idle state
$ResponseTime_i$	response time of user i
$T_{i,j,k}$	transmission time between component (i, j) and (i, k)
$P_{i,DCH}$	network interface of user i in high power consumption state
$P_{i,FACH}$	network interface power of user i in low power consumption state
$P_{i,network_idle}$	network interface power of user i in idle state
$E_{i,comp}$	computational energy of user i
$E_{i,tran}$	transmission energy consumption of user i
$E_{i,totaltail}$	tail energy of user i
$E_{i,network_idle}$	the energy consumption of the wireless network interface of user i in idle state
$ReponseTime_{avg}$	the average response time for all users
E_{avg}	the average energy consumption for all users

A. Average Response Time Model

Response time [23] is an important indicator for evaluating the performance of a mobile data streaming application. It refers to the time required to process input data by all components, and represents the total application latency. Users tend to be very sensitive to delays, so low response time can give users a higher degree of satisfaction. When there are multiple users, the average response time for all users is used as a measure of delay performance.

For user i , a migration strategy is defined as $X_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$, the value of $x_{i,j}$ is 0 or 1, 0 indicates that the component (i, j) is executed in the remote cloud, while 1 indicates local execution. The start time and completion time of a component can be calculated by formula (1) and (2).

$$ST_{i,k} =$$

$$\begin{cases} t_i & \text{pred}((i,k)) = \emptyset \\ \max_{(i,k) \in \text{succ}(i,j)} \{ET_{i,j} + T_{i,j,k} \cdot |x_{i,j} - x_{i,k}|\} & x_{i,k} = 0 \\ \max_{(i,k) \in \text{succ}(i,j)} \{ET_{i,j} + T_{i,j,k} \cdot |x_{i,j} - x_{i,k}|\} + \text{wait}_{i,k} & x_{i,k} = 1 \end{cases} \quad (1)$$

$$ET_{i,k} = ST_{i,k} + w_{i,k} \cdot x_{i,k} + \Gamma_{i,k} \cdot (1 - x_{i,k}) \quad i \in [1, \lambda] \quad (2)$$

The average response time of all users can be expressed as:

$$\begin{aligned} \text{ReponseTime}_{avg} &= \frac{1}{\lambda} \sum_{i=1}^{\lambda} (\text{ResponseTime}_i) \\ &= \frac{1}{\lambda} \sum_{i=1}^{\lambda} (ET_{i,j} - ST_{i,k}) \end{aligned} \quad (3)$$

Where (i,j) is the exit node, (i,k) is the entry node.

B. Average Energy Consumption Model

The power consumption of a mobile device is the sum of power consumed by many components, such as LCD, Bluetooth, GPS, CPU, wireless networks (e.g. WIFI, 3G) and so on [6]. In this paper, we are mainly concerned with the energy consumption of CPU and wireless network interface.

1) Computational Energy Consumption for Each User:

The CPU of the mobile device is typically switched between running and idle states. If there is a task that has not been completed or a new task is ready to become active, the CPU will always be in active running state. For a period of time when there is no task to be executed, CPU slowly transforms into a low power idle state. The computational energy of each user is divided into two parts: one is consumed in running state and the other consumed in idle state.

(1) Energy consumption during component execution

$$\begin{aligned} E_{i,cpu_comp} &= P_{i,cpu_comp} \cdot T_{i,comp} \\ &= P_{i,cpu_comp} \cdot \sum_{j \in V_i} (w_{i,j} \cdot x_{i,j}) \quad i \in [1, \lambda] \end{aligned} \quad (4)$$

(2) Energy consumption of the CPU in idle state

CPU time in idle state can be expressed as:

$$T_{i,cpu_idle} = \text{ResponseTime}_i - \sum_{j \in V_i} (w_{i,j} \cdot x_{i,j}) \quad (5)$$

Energy consumption of the CPU in idle state is:

$$E_{i,cpu_idle} = P_{i,cpu_idle} \cdot T_{i,cpu_idle} \quad (6)$$

Thus, the total computational energy consumption is:

$$E_{i,comp} = E_{i,cpu_comp} + E_{i,cpu_idle} \quad (7)$$

2) Transmission Energy Consumption for Each User:

Transmission energy consumption is generated by the wireless network interface during data transmission, which is usually divided into three states: high energy consumption, low energy consumption and idle [17].

The interface is in a state of high energy consumption when data needs to be sent or received. Even when there is no data transmission, the network interface will first be maintained at high energy consumption for some time, and then transferred to the state of low energy consumption. After staying in the low power state for some time, the wireless interface will eventually change into idle state.

In order to reduce the time of state switching and speed up the data transmission, the wireless interface would stay in high or low energy consumption state for some duration - energy generated during this period is called tail energy [17-19].

Therefore the transmission energy consumption is actually composed of three parts: data communication energy consumption, tail energy and the energy consumption in idle state.

(1) Data transmission energy consumption

For a particular user, the transmission energy consumption between two components that have dependencies is 0 when they are all executed locally or in the remote cloud, otherwise, the communication overhead will be generated. $P_{i,DCH}$ and $T_{i,tran}$ respectively indicate the power of high energy consumption and the total communication time, then the communication energy consumption is:

$$E_{i,tran} = P_{i,DCH} \cdot T_{i,tran} \quad (8)$$

$$T_{i,tran} = \sum_{(j,k) \in E_i} (T_{i,j,k} \cdot |x_{i,j} - x_{i,k}|) \quad i \in [1, \lambda] \quad (9)$$

(2) Tail energy and the energy consumption of wireless network interface in idle state

Tail energy is determined by the state of the current wireless network interface and the time interval between the two adjacent transmissions. $T_{i,1}$ and $T_{i,2}$ respectively indicate the time that network interface is kept in a state of high energy consumption and low energy consumption. Suppose that the time interval between the two adjacent data transmission is Δt , the number of time interval is n_i , the tail energy of two adjacent transmission can be expressed as:

$$E_{i,tail} = \begin{cases} P_{i,DCH} \cdot T_{i,1} + P_{i,FACH} \cdot T_{i,2} & \Delta t \geq T_{i,1} + T_{i,2} \\ P_{i,DCH} \cdot T_{i,1} + P_{i,FACH} \cdot (\Delta t - T_{i,1}) & T_{i,1} \leq \Delta t < T_{i,1} + T_{i,2} \\ P_{i,DCH} \cdot \Delta t & 0 < \Delta t < T_{i,1} \\ 0 & \Delta t = 0 \end{cases} \quad (10)$$

Where $1 \leq i \leq \lambda$, then the total tail energy of user i can be expressed as:

$$E_{i,totaltail} = \sum_{k=1}^{n_i} E_{i,tail} \quad (11)$$

The tail time generated by the two adjacent data transmission is:

$$T_{i,tail} = \begin{cases} T_{i,1} + T_{i,2} & \text{if } \Delta t \geq T_{i,1} + T_{i,2} \\ \Delta t & \text{if } 0 < \Delta t < T_{i,1} + T_{i,2} \\ 0 & \text{if } \Delta t = 0 \end{cases} \quad (12)$$

The total tail time and the time of the wireless interface in the idle state can be calculated by formula (13) and (14).

$$T_{i,totaltail} = \sum_{k=1}^{n_i} T_{i,tail} \quad (13)$$

$$T_{i,network_idle} = \text{ResponseTime}_i - T_{i,tran} - T_{i,totaltail} \quad (14)$$

The energy consumption of the wireless network interface in the idle state is:

$$E_{i,network_idle} = P_{i,network_idle} \cdot T_{i,network_idle} \quad (15)$$

The average energy consumption of all users can be expressed as:

$$E_{avg} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} (E_{i,comp} + E_{i,tran} + E_{i,totaltail} + E_{i,network_idle}) \quad (16)$$

IV. ALGORITHM DESIGN

A. Problem Description

Suppose that all users using mobile devices are running the same data streaming application. During a period of time, each user may submit a request to the cloud asynchronously for computation offloading, while at the same time their hardware information (such as computing power, memory size, available power) and network bandwidth are also sent to the cloud. Each user's request can be represented by a 6 tuple- $Request_i = \{i, RT_i, CP_i, Memory_i, Power_i, B_i\}$, where RT_i represents the request's submission time, and the total number of requests λ can be considered as subject to Poisson's distribution.

The cloud resources are represented by the number of virtual machines N . When the number of users is very large, the cloud resources will become in short supply, leading to partial components of each user can't be fully executed as expected. In the cloud there is a request receive processor which is responsible for receiving user information, and then based on the current battery capacity of mobile devices, available computing power, bandwidth and the usage of cloud resources, it will try to get a relatively optimal computation offloading results for all users, so that the overall performance and energy consumption of all users can achieve the best result under current condition.

B. Chromosome Encoding

A chromosome in the genetic algorithm corresponds to a specific migration strategy. Such as $chroms(X) = \{x_1, x_2, \dots, x_n\}$, where n represents the number of components, $x_j = 1$ represents component j is executed locally, while $x_j = 0$ represents it is executed in the remote cloud.

C. Fitness Function

The goal of the offloading algorithm is to minimize the average execution time of data streaming applications and the average power consumption of the mobile devices. The fitness of chromosome X can be calculated as follows:

$$fitness(X) = e^{\alpha \ln\left(\frac{ResponseTime(X)}{ResponseTime(all-in-local)}\right) + \beta \ln\left(\frac{Energy(X)}{Energy(all-in-local)}\right)} \quad (17)$$

For any user, $ResponseTime(X)$ and $Energy(X)$ denote the response time and energy consumption when each component of the application is executed in accordance with the offloading strategy X . $ResponseTime(all-in-local)$ and $Energy(all-in-local)$ respectively represent the response time and energy consumption when all components are executed in mobile device, α and β are the proportion of response time speedup ratio and energy consumption speedup ratio, and $\alpha + \beta = 1$.

D. Chromosome Manipulation

1) Selection Operator: The selection operation is to choose some relatively good chromosomes from the parent generation according to a certain proportion and the fitness function, so that the current computation offloading strategy moves one step closer towards the best approach. The purpose of this operation is to speed up the process of convergence. The selection process usually adopts the roulette method, assuming the total number of offloading strategies is M , the probability selecting the scheduling scheme X can be calculated as follows:

$$p(X) = \frac{fitness(X)}{\sum_{X=1}^M fitness(X)} \quad (18)$$

2) Crossover Operator: Crossover operation is to partially exchange the values of two chromosomes to form a new migration strategy. This operation also expands the solution space.

3) Mutation Operator: Mutation means changing the execution location of some components of certain users according to a certain proportion, so as to form a new offloading strategy.

E. Algorithm Description

The FMOCO algorithm proposed in this paper is an improvement over the searchadjust algorithm proposed in [9], it mainly includes four stages. First of all, the genetic algorithm is used to get a best result for all users according to the fixed target such as the energy consumption of mobile terminals and the response time of applications. Secondly, the number of the cloud virtual machine is added into the constraint condition, and the corresponding time slot is adjusted accordingly, adjustment is divided into delaying execution or returning components to mobile devices for execution, so as to get a relative optimal scheduling result for each user. Thirdly, selectively replicating some components to cloud based on application structure, component execution, and cloud resource usage. However, it should be noted that the replication will lead to additional energy consumption, while there may be more than the limit of cloud resources, it requires consideration of multiple factors to jointly determine whether it needs to replicate. As a result, when multiple users have multiple components to be replicated, the replication sequence is determined by (17). Lastly, the dynamic voltage frequency adjustment is carried out for the components which are executed in the mobile terminal and the latest completion time is greater than the earliest completion time, so as to save energy consumption of mobile devices.

Algorithm 1 Initial optimal computation offloading strategy for each user

Input: Mobile device configuration, application structure, Pop_size, Selection_rate, Mutation_rate, Maximum_iterations

Output: the initial computation offloading results for each user

1. initialize all chromosomes $chroms = \{\}$
 2. $chroms \leftarrow fitnessfunction(chroms)$
 3. initialize global optimal chromosome $gbest$
 4. $k \leftarrow 0$
 5. while($k < Maximum_iterations$) do
 6. $selchrom \leftarrow roulette_wheel(chroms)$
 7. $selchrom \leftarrow Crossover_op(selectchrom)$
 8. $selchrom \leftarrow Mutation_op(selectchrom)$
 9. $selchrom \leftarrow fitnessfunction(selectchrom)$
 10. $chroms \leftarrow update_chrom(chroms, selchrom)$
 11. $index \leftarrow indexofbestchrome()$
 12. if($chroms[index].fitness < gbest.fitness$)
 13. $gbest \leftarrow chroms[index]$
 14. end if
-

```

15.  k ← k + 1
16.  end while

```

As the first step of FMOCO, with the assumption that the cloud resource is unlimited, the initial optimal migration strategy is obtained for each user.

To start with, the population is initialized to generate Pop_size chromosomes, with each chromosome maps to an offloading strategy (line 1). Next, the fitness value of each chromosome is calculated (line 2), and the initial global optimal scheduling strategy is obtained (line 3). Then the iterative process starts: select a group of chromosomes based on the roulette wheel method and the selection ratio; apply crossover and mutation operation to selected chromosomes; find out the optimal solution in this generation; update the global optimal migration strategy (line 4-16). When the number of iterations is reached, the global optimal solution thus far achieved would correspond to the initial optimal migration strategy for each user.

Algorithm 2 The final computation offloading strategy for each user

Input: The initial computation offloading result for each user, cloudservernumber

Output: The final computation offloading result for each user

```

1.  REPEAT /* begin adjust */
2.  initialize the resource occupation list
   ResourceList
3.  FOR each timeslot in ResourceList DO
4.  IF occupiedservernum > cloudservernum THEN
5.  candidate ← find out user to be adjusted and
   corresponding component
6.  delay the execution of the candidate or migrate
   candidate back to the mobile terminal
7.  update_execetime(candidate)
8.  break
9.  END IF
10. END FOR
11. UNTIL the number of virtual machines in all
   timeslots < cloudservernumber
12. WHILE(1) DO /* begin duplicate */
13. FOR each component (i,j) DO
14. candidate ← find component that should be
   duplicated
15. END FOR
16. IF (candidate != ∅) THEN
17. update_execetime(candidate)
18. initialize ResourceList
19. END IF
20. ELSE
21. break
22. END WHILE
23. calculate the latest finish time for each component
   (i,j)
   /* begin dvfs */
24. FOR each component (i,j) DO
25. IF exec_inlocal(i,j) and latest_finish_time[i][j] >
   earliest_finish_time[i][j] THEN

```

```

26. dvfs(i,j)
27. END IF
28. END FOR

```

According to the initial optimal migration strategy, the implementation of cloud components is obtained, and then the start time, the end time and the number of occupied virtual machine of each time slot are also obtained (line 2). Next, it traverses all timeslots in case there are some time periods exceeding resource constraints, it tries to locate the components to be adjusted from all users' components that are executed in the cloud and then calculate the reward of these components individually. The adjustment can be either migrating components back to mobile terminals or delaying the execution of components. The strategy is to postpone the execution of the adjusted components as long as possible because the added response time is relatively small at this time. Any change in the execution time of a component will affect the execution of the subsequent components, therefore, it is necessary to update the execution time of the corresponding components of the user, and then re-divide the time slot as well as recount resource usage. Repeat this process until the resource constraints can be met at each time period (line 3-11). In addition, some components on the critical path of some users are selectively replicated to reduce the response time. Replication can cause changes in the execution time of the components, so it is also necessary to update the execution time of the corresponding components (line 12-22). In the end, the latest completion time of each component of each user is calculated (line 23). If the latest completion time is greater than the earliest completion time, DVFS for these components will be activated in the mobile devices to save energy by increasing the processing time (line 24-28). In the end, the final computation offloading decisions for all users are obtained.

F. Time Complexity Analysis

Algorithm 1 achieves the first phase of the FMOCO algorithm. First, initialize the population, calculate the fitness of each strategy, update gbest, the time complexity is $O(\text{Pop_size} \times n)$. The time complexity of the iterative process is $O(\text{Pop_size}^2 + \text{Pop_size} \times n)$. The time complexity of algorithm 1 is $O((\text{Pop_size}^2 + \text{Pop_size} \times n) \times \text{Maximum_iterations})$.

Algorithm 2 consists of three stages: adjustment, replication and DVFS. Adjustment includes initializing the cloud resource usage list, finding components that are executed in cloud and do not meet the number of VMs in the cloud, as well as calculating the effectiveness of each candidate component, the time complexity is $O(\lambda^2 \times n^3)$. Replication requires finding the critical path and traversing all components that can be replicated. The time complexity is also $O(\lambda^2 \times n^3)$. The time complexity of DVFS is $O(\lambda \times n)$.

Thus, the total time complexity of FMOCO is $O((\text{Pop_size}^2 + \text{Pop_size} \times n) \times \text{Maximum_iterations} + \lambda^2 \times n^3)$.

V. EVALUATION

A. Experimental Testbed

1) Cloud and Mobile Device Configuration: Our experiment uses a multi node Grizzly OpenStack cloud platform consisting of 4 servers as the cloud and a number of laptop computers as mobile devices. For the cloud, the 4 servers are all Lenovo Thinkserver RD330, with Intel Xeon E5 2407 CPU running at 2.2GHz, 16G memory and CentOS 6.5.

In order to analyse the relationship between migration strategy and mobile terminal, we tested with three different smart phones, namely the ZTE N900, Samsung N9002 and Smartisan U1. The computing power and the power of the wireless network interface in different states are listed in Table 2 below (unit: mw).

TABLE 2. MOBILE DEVICE CONFIGURATION

State	Smart phone		
	ZTE N900	Samsung N9002	Smartisan U1
CPU	running	577	1476.3
	idle	6.29	14.8
3G	high	582	573.5
	low	398	414.57
	idle	15	19.98

2) Network Configuration: In the cellular network, tail energy is more dominant, so we mainly use 3G network in our experiments. Also to demonstrate how offloading is related to network bandwidth, we used three different bandwidth, 240kbps, 480kbps and 960kbps.

3) Mobile Data Streaming Application Configuration: Data streaming applications can be divided into two categories, one is any one component has one successor component at most such as face recognition, and the other is a component can have a number of subsequent components. Without loss of generality, in this paper, we use QR code recognition application [24-25] in which components can be executed in parallel.

B. Experimental Results and Analysis

In this section, we compare FMOCO with three existing strategies: searchadjust, All-in-local, and All-in-cloud. searchadjust is an offline multi user computation offloading algorithm, whose goal is to minimize the average response time of all users. All-in-local means that all components are executed locally, hence no additional communication cost. All-in-cloud means that all components are executed in a remote cloud— this strategy can greatly reduce the computational load of mobile terminals, but it will cause excessive transmission overhead.

1) The Impact of the Number of Cloud Virtual Machines: This group of experiments studies the impact of the number of cloud virtual machine on different strategies. The specific configuration is shown in table 3 below.

TABLE 3. ENVIRONMENT CONFIGURATION OF EXPERIMENT 1

Experimental environment	Parameter
Number of users	3
Network bandwidth	480kbps
Mobile device	ZTE N900
Number of virtual machines	2, 4, 6

Detailed results of the experiments are shown in Figure 1(a) and 1(b) below.

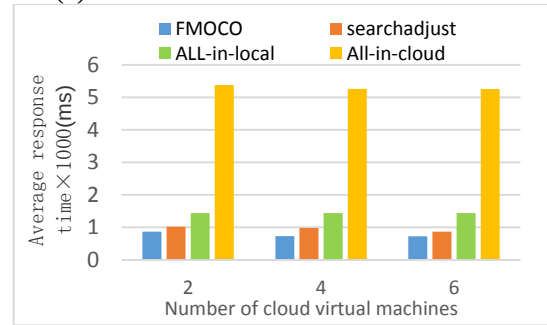


Figure 1(a). The impact of the number of virtual machines on the average response time

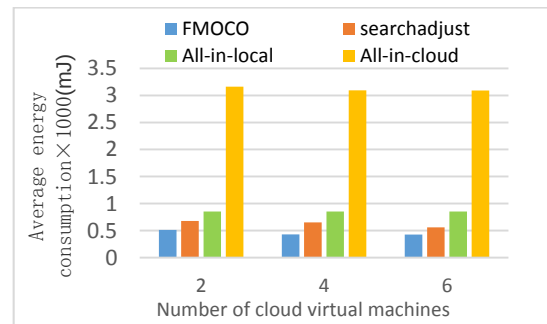


Figure 1(b). The impact of the number of virtual machines on the average energy consumption

As shown in Figure 1(a), with the increase in the number of cloud virtual machine, the average response time for all users' computation offloading strategy calculated by FMOCO decreases. This is because when the number of users is fixed, as more virtual machines become available, fewer components would require delayed execution or to be migrated back to the mobile terminal, therefore the results of each user are closer to the original optimal scheduling results. All-in-cloud shows similar results, but its response time remains the same as there is no component in the cloud. On the other hand, the average response time of FMOCO is clearly less than those of the other three strategies.

As shown in Figure 1(b), when fewer cloud virtual machines are available, the average energy consumption of FMOCO becomes higher. This is because the user's response time increases as the number of virtual machines decreases, more components have to be delayed or returned to the mobile terminal because they do not meet the constraints, consequently computational energy, energy consumption of CPU in idle state and tail energy will all increase to some degrees, contributing to higher overall energy consumption.

2) The Impact of the Number of Users: This group of experiments studies the impact of the number of users on different strategies.

TABLE 4. ENVIRONMENT CONFIGURATION OF EXPERIMENT 2

Experimental environment	Parameter
Number of virtual machines	4
Network bandwidth	480kbps

Mobile device	ZTE N900
Number of users	2, 4, 6

All users in this experiment use the same phone, but with variable number of users. Detailed results of these experiments are shown in Figure 2(a) and 2(b).

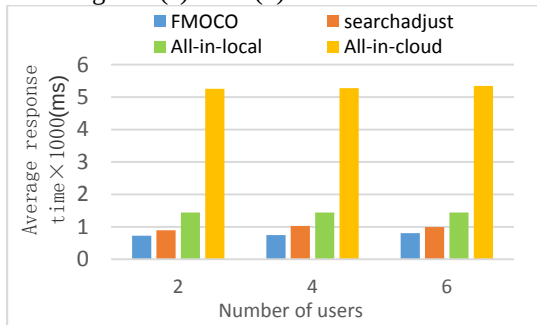


Figure 2(a). The impact of the number of users on the average response time

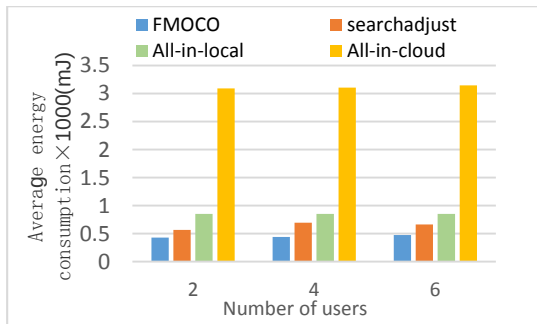


Figure 2(b). The impact of the number of users on the average energy consumption

As shown in Figure 2(a), with the increase of the number of users, the average delay of migration strategy gradually increases for both FMOCO and searchadjust algorithms. The more the number of users, the more shortage cloud resources becomes, and more components will have to be delayed or returned to the mobile devices.

As expected the change in the number of users has no effect on the All-in-local strategy. However due to the limited computing power of mobile terminals, its average performance is lower than FMOCO and searchadjust. Furthermore, the performance of FMOCO is better than searchadjust, because the latter is simply to migrate components to mobile devices, which will greatly increase the response time of the application.

As shown in Figure 2(b), apart from the All-in-local strategy, the average energy consumption of mobile devices for the other three strategies is heavily affected by the number of users – basically it trends higher with the increase of the number of users. As explained earlier, the competition for resources among multiple users results in the actual migrating strategy deviating significantly from the optimal migration results without resource constraints. But it can be seen clearly that the average energy consumption of FMOCO for the same number of users is less than the other three strategies. While searchadjust is only concerned with the average response time, FMOCO can achieve the lowest average energy consumption as well as minimizes the average response time of all users. The average energy consumption of All-in-cloud is the largest,

which is mainly caused by transmission energy consumption and tail energy.

3) The Impact of Network Bandwidth: This group of experiments studies the impact of network bandwidth on different strategies. The specific environment configuration is shown in table 5 and the detailed results are presented in Figure 3(a) and 3(b).

TABLE 5. ENVIRONMENT CONFIGURATION OF EXPERIMENT 3

Experimental environment	Parameter
Number of users	6
Number of virtual machines	5
Mobile device	N900、N9002、U1
Network bandwidth	240、480、960(unit: kbps)

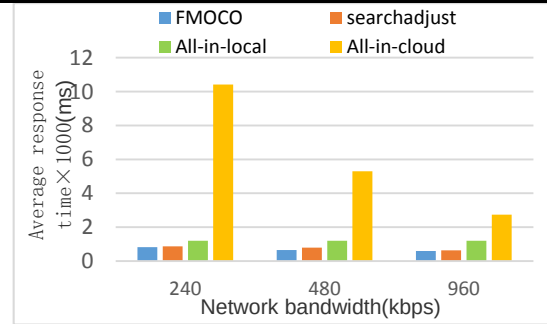


Figure 3(a). The impact of network bandwidth on average response time

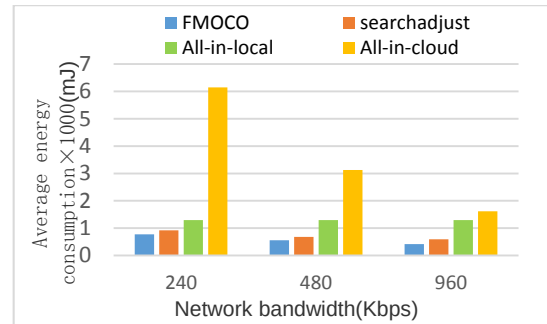


Figure 3(b). The impact of network bandwidth on average energy consumption

As can be seen from Figure 3(a), the average response time of FMOCO, searchadjust and All-in-cloud strategy is inversely related to the bandwidth. This is because with the increase in bandwidth, transmission time is reduced, and more components will be offloaded to the cloud, resulting in faster execution. However, the average response time of All-in-local policy is not affected by bandwidth, because no component is executed in the cloud. As a comparison, FMOCO can delay the execution of components, so the average response time of FMOCO is smaller than searchadjust under the same conditions.

From Figure 3(b), it can be seen that the average energy consumption of FMOCO and searchadjust are not necessarily related to the network bandwidth. As the bandwidth increases, the number of components that are migrated to the cloud will increase. On the one hand, this can effectively reduce the computational energy, but on the other hand it also increases transmission energy consumption and tail energy, so the end effect as determined by these two components is not

deterministic. The average response time and average energy consumption under All-in-local strategy is not related to the change of bandwidth, because the components are executed locally, and do not need to send or receive data through the wireless network. From the results it is clear that the average energy consumption under the FMOCO strategy is the least among these strategies.

4) The Impact of Mobile Device's Available Battery Power:

This group of experiments studies the effects of available battery power on migrating strategies.

TABLE 6. ENVIRONMENT CONFIGURATION OF EXPERIMENT 4

Experimental environment	Parameter
Number of users	6
Number of virtual machines	5
Network bandwidth	240kbps
Mobile device	N900、N9002、U1
Available battery power	saturation、inadequate

The available battery power can be divided into two levels: A, saturation (residual power $\geq 30\%$) and B, inadequate ($< 30\%$). The detailed experimental results are shown in Figure 4(a) and 4(b).

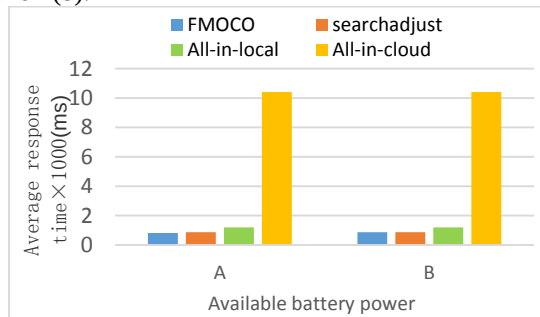


Figure 4(a). The impact of available battery power on average response time

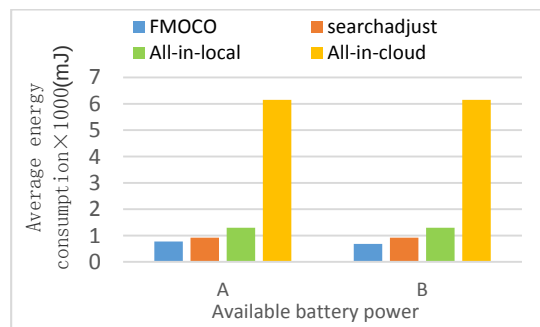


Figure 4(b). The impact of available battery power on average energy consumption

In Figure 4(a), the average response time of the other three strategies has nothing to do with the battery power, while FMOCO proposed in this paper will change accordingly based on the remaining battery power. When the battery power is sufficient, the performance is the first goal, and energy consumption is a secondary goal, while the battery power is low, the energy consumption is transformed into the main goal, thus more components will be transferred to the cloud execution to reduce the computing load of the mobile devices, so the average response time will increase with the decrease of the battery's

available power, essentially it is to increase the execution time in exchange for low energy consumption.

Since the average energy consumption under FMOCO strategy can be adjusted based on the current available power of mobile devices, the mobile device can automatically save power when its battery power is insufficient. As shown in Figure 4(b), compared with searchadjust, All-in-local and All-in-cloud, the average energy consumption saving ratio of FMOCO strategy is 20.6%, 43.8%, 88.2%, respectively for the two different battery power levels.

5) The Influence of Computing Power: This group of experiments studies the impact of computing power on migration strategies.

TABLE 7. ENVIRONMENT CONFIGURATION OF EXPERIMENT 5

Experimental environment	Parameter
Number of users	6
Number of virtual machines	5
Network bandwidth	480kbps
Mobile device	N900、N9002、U1

The computing power of the three mobile devices selected for testing is different, with N900 ZTE the weakest, Smartisan U1 in the middle, and Samsung N9002 the strongest. Detailed results of the experiments are shown in Figure 5(a) and 5(b).

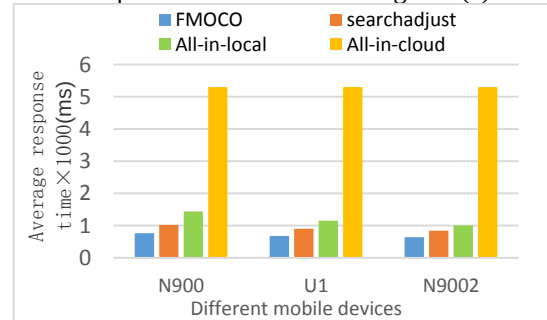


Figure 5(a). The impact of computing power on the average response time

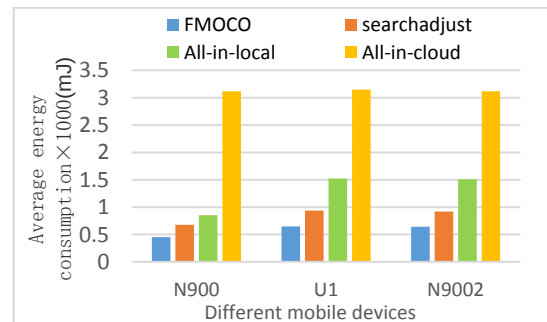


Figure 5(b). The impact of computing power on the average energy consumption

As can be seen from Figure 5(a), All-in-cloud strategy has nothing to do with the local computing power, because all components of all users are executed in the cloud. The performance of other strategies is positively correlated with computing power: on the one hand, the stronger the computing power of mobile phone is, the less time it takes to execute the same components, on the other hand, more components tend to be executed locally, which can effectively reduce the

transmission time. FMOCO is more inclined to delay the execution of the components, which means the increase in response time is smaller, so that its performance is better than searchadjust.

As can be seen from Figure 5(b), increasing computing power can reduce the response time and to some extent the tail energy as well, but it also means that computing power will be higher, so the final result is determined by these factors. Nonetheless, it is not difficult to see that under the same conditions, the FMOCO strategy of this paper has an obvious advantage over other strategies in improving the battery life.

VI. CONCLUSIONS

In this paper, we proposed a four-stage multi-user offline computation offloading algorithm (FMOCO) based on searchadjust algorithm, but motivated by the fact that cloud resources are not unlimited. Firstly, the initial computation offloading result of each user is obtained by genetic algorithm. Then adjustments are made to time periods during which the number of virtual machines occupied exceeds the capacity provided by the cloud. If there are components that can be replicated, some components on the critical path for some users will be replicated to reduce the response time and improve the utilization of virtual machines. Finally, the voltage and frequency can be reduced to a suitable level by DVFS technology, which can reduce the energy consumption by extending the execution time of components. The experimental results show FMOCO algorithm can achieve a good balance between performance and energy consumption of mobile devices, and generally provides better migration strategy than the other three existing strategies studied as references.

Future research work will focus on two new areas. The first one is to study the multi-user online migration strategy under cloud resource constraints. Online algorithm not only needs to know the number of current offloading requests, but also needs to forecast the number of possible incoming requests in the next period of time, so as to balance the performance of each time period. The second one is to study the case when the virtual machines provided by cloud are heterogeneous which obviously requires a strategy different from that of isomorphism.

ACKNOWLEDGMENT

The paper is supported by the Ministry of Education Project of Humanities and Social Sciences (Project Name, Research on fuzzy QoE aware and trustworthy cloud services selection in mobile commerce, Project No.16YJCZH014), the Natural Science Foundation of Hubei Province of China (Project Name, Research on the replica-aware data acquiring mechanism for mobile cloud computing, Project No. 2016CFB466), the Fundamental Research Funds for the Central Universities (WUT:2016III028).

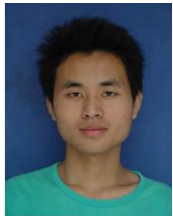
REFERENCES

- [1] CNNIC. Statistical Report on Internet Development in China, January 2016.
- [2] Qawlan Akariman, Agung Nugroho Jati, Astri Novianty. Face Recognition Based on the Android Device Using LBP Algorithm//Proceedings of International Conference on Control, Electronics, Renewable Energy and Communication, 2015: 166-170
- [3] Suzy Deffeyes, Neil Katz. Mobile augmented reality in the data center. IBM Journal of Research and Development, 2011, 55(5): 1-5
- [4] Yating Wang, Ing-Ray Chen, et al. A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges. Wireless Personal Communications, 2014, 80(4): 1607-1623
- [5] Xu Chen, Lei Jiao, et al. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. IEEE/ACM Transactions on Networking, 2015, pp(99): 1-14
- [6] G. P. Perrucci, F. H. P. Fitzek, et al. Survey on Energy Consumption Entities on the Smartphone Platform//Proceedings of the 73rd Vehicular Technology Conference (VTC Spring), 2011: 1-6
- [7] Niroshinie Fernando, Seng W. Loke. Mobile cloud computing: A survey. Future Generation Computer Systems, 2013, 29(1): 84-106
- [8] Kelie Zhan, Chung-Horng Lung, et al. A Comparative Evaluation of Computation Offloading for Mobile Applications//Proceedings of IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications(GreenCom) and IEEE Cyber, Physical and Social Computing(CPSCom), 2014: 518-525
- [9] Lei Yang, Jiannong Cao. Multi-user Computation Partitioning for Latency Sensitive Mobile Cloud Applications. IEEE Transactions on Computers, 2015, 64(8): 2253 - 2266
- [10] Xiaoqiang Ma, Yuan Zhao, Lei Zhan, et al. When mobile terminals meet the cloud: computation offloading as the bridge. IEEE Network, 2013, 27(5): 28-33
- [11] S. Deng, Huang L, Taheri J, et al. Computation Offloading for Service Workflow in Mobile Cloud Computing. IEEE Transactions on Parallel and Distributed Systems, 2016, 26(12): 1045-9219
- [12] Zhang Wen-li, Guo Bing, Shen Yan, et al. Computation offloading on intelligent mobile terminal. Chinese Journal of Computers, 2016, 39(5): 1021-1038(in Chinese)
- [13] Rich Wolski, Selim Gurun, Chandra Krintz, et al. Using Bandwidth Data To Make Computation Offloading Decisions//Proceedings of IEEE International Symposium on Parallel and Distributed Processing, 2008: 1 - 8
- [14] Xing Liu, Zhen Yang. An Energy Saving Algorithm based on User-Provided Resources in Mobile Cloud Computing//Proceedings of IEEE 78th Vehicular Technology Conference, 2013: 1-5
- [15] Weiqing Liu, Jiannong Cao, Xuanjia Qiu, Jing Li. Improving Performance of Mobile Interactive Data-Streaming Applications with Multiple Cloudlets// Proceedings of IEEE 6th International Conference on Cloud Computing Technology and Science, 2014: 46 - 53
- [16] M. Reza Rahimi, Nalini Venkatasubramanian, et al. MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing//Proceedings of IEEE Sixth International Conference on Cloud Computing, 2013: 75 - 82
- [17] Y. Geng, W. Hu. Energy-Efficient Computation Offloading in Cellular Networks//Proceedings of IEEE 23rd International Conference on Network Protocols (ICNP), 2015: 145-155
- [18] Yu Fei. Analysis of Energy Consumption on 3G Terminals [M. S. dissertation]. Shanghai Jiao Tong University, Shanghai, 2013(in Chinese)
- [19] Wu Yu. Adaptive offloading decision making model for mobile computing [M. S. dissertation]. Fudan University, Shanghai, 2012(in Chinese)
- [20] Balasubramanian N, Balasubramanian A, Venkataramani A. Energy consumption in mobile phones: a measurement study and implications for network applications//Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, 2009: 280-293
- [21] Guo Jun-Wu. Tasks Migration Technology for Computation-intensive Tasks in Mobile Cloud Computing [M. S. dissertation]. Harbin Institute of Technology, Harbin, 2015

- [22] Di Zhang, Yaoxue Zhang, et al. Leveraging the Tail Time for Saving Energy in Cellular Networks. IEEE Transactions on Mobile Computing, 2014, 13(7): 1536 – 1549
- [23] Moo-Ryong R, Anmol Sheth, Lily Mummert, et al. Odessa: Enabling Interactive Perception Applications on Mobile Devices//Proceedings of International Conference on Mobile Systems, 2011: 43-56
- [24] Tang Xiao. Research of Energy-efficient Computation Offloading and Scheduling Strategies in Mobile Cloud Computing [M. S. dissertation. Wuhan University of Technology, Wuhan, 2015(in Chinese)
- [25] L.Yang, J.Cao, S.Tang, T. Li, et al. A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing//Proceedings of 5th International Conference on Cloud Computing, 2012: 794-802



Wei Liu was born in 1978 and received his Ph.D in computer architecture from the Department of Computer Science and Engineering, Fudan University. Currently, he is working in Wuhan University of Technology as an associate professor of the college of the computer science and technology. His research interests include cloud computing and green computing. He has published more than 10 papers in national or international key journals.



WanJia Gong received his bachelor's degree from Jiangxi Agricultural University in 2014. He is currently a master candidate in School of Computer Science and Technology, Wuhan University of Technology.



Wei Du was born in 1978 and received her Ph.D. in computer science from the Department of Computer Science and Engineering, Huazhong University of Science and Technology. Currently, she is working in Wuhan University of Technology as a lecture of the college of the computer science and technology. Her research interests include information security and service computing. She has published more than 10 papers in national or international key journals.



Chengming Zou is a graduate tutor of Wuhan University of Technology. The main research directions of him are computer vision, embedded systems, software theory and methods (operating system). He is the member of China Computer Federation. Up to now, Prof.Zou has published more than 30 academic articles. More than 10 papers are indexed by SCI, EI and ISTP. He has written 2 monographs, and applied for 10 copies of software copyright. He has undertaken more than 10 scientific research projects. And he won the Hubei Provincial Science and Technology Progress Award.