Single and Double-Threaded Offloading Strategies and Computational Analysis in Mobile Cloud Computing

Siyu Xia\textsuperscript{1, a}, Mingyuan Jiu\textsuperscript{2, b, *}, Chenglong Fan\textsuperscript{1, c}

\textsuperscript{1}International College, Zhengzhou University, #75, Daxue Avenue, Zhengzhou, Henan, China
\textsuperscript{2}School of Information Engineering, Zhengzhou University, #100, Kexue Avenue, Zhengzhou, Henan, China
\textsuperscript{a}15225758959@163.com, \textsuperscript{b}iemyjiu@zzu.edu.cn, \textsuperscript{c}fancl@zzu.edu.cn

ABSTRACT. At present, mobile devices are hard to support the increasing number of intensive applications due to their limited power and fixed storage size, therefore, how to improve the computing power and also reduce the energy consumption is a crucial problem in the development of mobile terminals. In order to solve this problem, a new offloading scheme is proposed based on the combination of double-thread computing scheme and offloading strategy in this paper. First, we combine single and double-threaded computing with mobile cloud computing and obtain the optimal offloading strategy through component partitioning and the single-threaded application theorem. Then the energy consumption is analyzed on the basis of double-thread calculation. Finally, we compare the performance of single-thread and double-thread offloading strategy in terms of energy consumption and delay of mobile cloud computing. The results show that the double-thread offloading strategy has the least energy consumption and has better characteristics.

1. INTRODUCTION
With the rapid development of electronics technology, mobile devices have been widely used in the real-life. However, there exists some weakness such as low battery and storage capacity, small computing power to limit their applications. Many skills have been proposed to solve these bottlenecks. Among them, mobile cloud computing (MCC)—the combination of clouding and mobile wireless technologies, is one successful skill, which enables mobile devices to capture the results of heavy computing in a lower configuration.

Research has shown that choosing some tasks to offload to the cloud servers can save energy for mobile devices [1]. Offloading means that data storage and computation are completed on remote cloud servers. The problem then becomes an optimization problem of choosing the computing scheme with low energy consumption, which is one of the main research directions. There are two important stages to process before the offloading calculation is performed: code partition and component migration. Several works have been conducted to decide on the optimal offloading strategy can be obtained by linear complexity algorithm and heuristic algorithm respectively. We consider that it is difficult to calculate the best strategy in general applications, that’s why we think it is advisable to convert it into a chain application. Combined with multi-threaded computing (MTC), we can get a new optimal offloading strategy. This paper mainly involves the following aspects: Firstly, this paper analyses the offloading strategy in mobile cloud computing. On the basis of the existing offloading...
strategy, we consider the combination of MTC and propose an optimized offloading strategy for general applications. Then according to the idea of multi-threaded offloading, we find out the best offloading scheme of mobile cloud computing, and finally we also conduct several simulation experiments and the results validate our proposed method.

Recently, there are many published relevant works on mobile cloud computing. In [4], in order to maximize the speed of data processing, partition processing is carried out. The search strategy is obtained to achieve the optimal flow control through the study of energy consumption, delay, access cost and other factors [5]. Cheng et al. proposed to represent linear topologies with directed graphs and a decision algorithm—genetic algorithm, for loading between multiple devices [6]. However, multi-thread computing is not considered in the above works. In the spirit of these works, we propose a double-thread computing scheme combined with offloading using a directed graph, including network arcs and connection components in this work.

The rest of the paper is organized as followed: The offloading strategy is described in Section 2. Section 3 gives the system model and mathematical formulation. The solution to the problem is optimized according to the above models in Section 4. In Section 5, we show the results of the specific analysis for different schemes, as followed by the conclusion in Section 6.

2. OFFLOADING CLASSIFICATION
Depending on which components are migrated, the offloading strategy can be divided into local offloading and complete offloading, which is briefly introduced as follows:

**Local offloading:** We divide the data and only transmit the data with high energy consumption in order to reduce the amount of data transmitted. There are several parts under the offloading client-server architecture: monitor, offloading engine, wireless bandwidth of mobile devices, module, CPU utilization [7]. We can obtain communication costs and computing time before performing offloading.

**Complete offloading:** We migrate the entire program to the cloud for calculation if the mobile device cannot complete the calculation in a specific short period of time. The disadvantage is that it can cause extra energy consumption in the transmission process.

3. SYSTEM AND COMPUTATION MODEL
In MCC, we need to consider the following problems: mobile communication problem and data migration problem. Based on these issues, we briefly present three models: system model, application model and mathematical model.

3.1 System Model
MCC system is composed of three parts: mobile terminal equipment, communication network and cloud servers.

The mobile terminal device is the data source of the application which processes the application accordingly. On mobile terminal devices, we need to conduct perceptual analysis of the application to obtain the computational offloading data, parameters and complexity of the application model, and then we divide the application into a manageable set of components. It is used to provide the information which is needed for the offloading decision. We use communication network to connect mobile terminal devices and cloud servers. Communication network influences users’ application experience through its bandwidth, interference and other characteristics. In brief, the transmission of data through communication network brings certain time delay and energy consumption. Cloud servers have powerful computing power and storage capacity. When offloading data to the cloud, it consumes energy of mobile terminals. At this point, we do not consider the energy consumed by cloud computing, but the time delay on cloud computing. Therefore, we need to consider the calculation amount and frequency of cloud computing.
3.2 Application Model
In this section, we introduce the application model, including model based on delay constraint and model based on energy.

**Model based on delay constraint:** Time delay can be divided into two parts: calculation latency and transmission latency. Computing latency refers to the time taken by cloud servers and mobile devices to process offloaded data. Transmission latency refers to the time taken by the offloaded data to be transmitted in the channel, which can be classified into two types—the uplink delay and the downlink delay. The uplink delay happens when data are offloaded from the mobile device to the cloud service while the other one happens when the calculation results are returned from the cloud to the mobile device. The constraint is that the offloading strategy we used must be completed within a specific time delay.

**Model based on energy:** We use directed graph \( O(D,R) \) to represent components and their connections, \( D = \{ n | n = 0,1,2,...,l+1 \} \) , \( R = (n,m), n,m \in D \). The energy consumption on the cloud equipment is out of our consideration, we only calculate the energy consumption of the mobile equipment in the cases of [8]. The energy of mobile devices mainly includes the energy of component partition and data encoding, the energy used to calculate the data locally, the energy used in the channel to transfer data from mobile devices to cloud servers and the energy of data decoding after the mobile device receives the calculated results. What’s more, the energy consumption of the mobile devices and cloud servers are related to the calculation amount of data. Therefore, we use \( C_n \) to represent the calculation amount of a group of components, when the input data amount is \( M_n \) and the corresponding complexity is \( V_n \), we can get the specific value of the calculation amount from the following equation:

\[
C_n = M_n V_n
\] (1)

3.3 Mathematical Model
In this section, we will introduce the data communication model and the calculation model of energy consumption.

Channel model of data communication: First of all, we assume that the noise in the wireless communication channel is Gaussian white noise, whose mean value is zero and variance is \( \sigma^2 \). According to the Shannon’s Formula, the channel capacity is:

\[
C_t = B \log_2 \left( 1 + \frac{P_n G_n}{\sigma^2} \right)
\] (2)

\[
G_n = \frac{E_n}{H}
\] (3)

where \( B \) (Hz) is the transmission bandwidth, \( P_n \) (Hz) is the transmission power of the nth component, \( G_n \) is the gain of signal to noise ratio, \( E_n \) is the transmission power gain, and \( H \) is the encoding gap [9]. The transfer time \( T_n \) is calculated as:

\[
T_n = M_n \frac{P_n E_n}{\sigma^2 H}
\] (4)

The energy consumption of data transmission is related to data bit size \( q \), channel gain \( c \), energy coefficient \( \gamma \) and encoding mode \( m \). The specific forms are as follows:

\[
E_n = \gamma c q^m
\] (5)

Calculation model: Previous research results show that CPU power consumption is a super linear function of frequency, which can be expressed as follows:

\[
P_n = \beta (f_{ci})^r
\] (6)
where \( f^l_n \) is defined as the calculate frequency of \( n \)th component on the mobile device, \( \beta \) is the CPU power loss coefficient and we choose \( \tau = 2 \) empirically in the following simulation experiments [8]. The time spent in calculating the \( n \)th component is expressed as:

\[
T_n = \frac{C_n}{f^l_n}
\]  

(7)

Then combining (6) with (7), we can get the local energy formula:

\[
E^l_n = \beta f^l_n T_n = \beta C_n \frac{C_n^2}{T_n}
\]

(8)

Since it is widely believed that cloud computing does not consume the energy of mobile devices, we do not take into account the energy consumption of cloud server:

\[
E^c_n = 0
\]

(9)

4. PROBLEM FORMULATION

In this section, we determine the single-thread, double-thread offloading scheme and component partition in consideration of specific situations. As shown in the Fig. 1. We use \( U = \{u_n| n = 0, 1, 2, \ldots, l, l + 1 \} \) to represent offloading strategy. When deciding to offload \( n \)th component, we set \( u_n = 1 \), otherwise, \( u_n = 0 \). Besides, we assume \( u_0 = u_{l+1} = 0 \).

![Fig. 1 The specific double-threaded calculation scheme.](image)

4.1 Section division

First, we discuss the problem of energy consumption in offloading single-threaded application. How to minimize offloading energy for single-threaded application? According to [10], we can obtain the offloading theorem for single-threaded application:

**Theorem 1:** for single-chained applications, data is migrated from the mobile device to the cloud server at most once.

According to the analysis, the energy consumption of mobile devices is correspondingly reduced if the number of segments is reduced. Finally, we find that there is only one segment of data offloaded to the cloud to minimize the energy consumption.

Section division is to save energy through parallel computing. Within a given time, we appropriately reduce the frequency of the thread which completes the calculation first. It can be seen from [11] that energy can be saved by reducing the calculation frequency.

4.2 Energy Consumption and Latency

After component division, the energy consumed by component local calculation is:

\[
E^l_{\text{sum}} = \beta \sum_{0}^{j+1} \frac{(1 - u_n) f^l_n}{l^l_n}
\]

(10)

The CPU frequency of the selected mobile device is 2.7GHz, \( \beta = 3.7 \times 10^{-21} \). The corresponding local computing time is:

\[
T^l_{\text{sum}} = \frac{\sum_{0}^{l+1}(1 - u_n) C_n}{f^l_n}
\]

(11)
Encoding data locally is to prevent unauthorized users’ access. And sending pointers instead of data saves a lot of energy [12]. We use the following formulas to represent the energy consumption of the data encoding and decoding:

\[
E_{\text{sum}}^{\text{en}} = \beta \frac{\sum_{i=0}^{i+1} (1 - u_n)^2}{T_{lo}} 
\]

\[
E_{\text{sum}}^{\text{de}} = \beta \frac{\sum_{i=0}^{i+1} u_n^2}{T_{de}} 
\]

\[
T_{\text{sum}}^{\text{en}} = \frac{\sum_{i=0}^{i+1} (1 - u_n)}{f_n^l} 
\]

\[
T_{\text{sum}}^{\text{de}} = \frac{\sum_{i=0}^{i+1} u_n}{f_n^l} 
\]

\[C_{\text{en}} \text{ and } C_{\text{de}} \text{ represent the amount of computation of the uplink and downlink data separately. The energy consumed in the channel during data transmission can be described as:}
\]

\[
E_{\text{sum}}^{\text{off}} = c Y_{\text{off}} \sum_{i=0}^{i+1} (1 - u_n) u_m q_{\text{off}}^{m_{\text{off}}} 
\]

\[
E_{\text{sum}}^{\text{down}} = c Y_{\text{down}} \sum_{i=0}^{i+1} u_n (1 - u_m) q_{\text{down}}^{m_{\text{down}}} 
\]

The time of data transmission in the channel is:

\[
T_{\text{ch}}^{\text{off}} = \sum_{i=0}^{i+1} (1 - u_n) u_m \frac{M_{\text{off}}^{\text{off}}}{B log_2 \left( 1 + \frac{p_{\text{off}} E_{\text{off}}^{\text{off}}}{\sigma^2 H_{\text{off}}^{\text{off}}} \right)} 
\]

\[
T_{\text{ch}}^{\text{down}} = \sum_{i=0}^{i+1} u_n (1 - u_m) \frac{M_{\text{down}}^{\text{down}}}{B log_2 \left( 1 + \frac{p_{\text{down}} E_{\text{down}}^{\text{down}}}{\sigma^2 H_{\text{down}}^{\text{down}}} \right)} 
\]

\[M_{\text{off}} \text{ and } M_{\text{down}} \text{ represent the data bits transmitted on the uplink and downlink respectively. We only consider the time delay occupied by cloud computing:}
\]

\[
T_{\text{sum}}^{\text{cl}} = \sum_{i=0}^{i+1} u_n \frac{C_{\text{en}}}{f_n^{\text{cl}}} 
\]
5. ANALYSIS OF OFFLOADING STRATEGY

We use $E_{sc}$, $E_{dc}$, $E_{soc}$, $E_{doc}$ to represent calculation energy consumption by single and double-thread, energy consumption of single and double-thread offloading strategy respectively.

(1) At a given time delay, $E_{dc}$ and $E_{sc}$ are shown in Fig. 2. With the latency constraint, we can find out that the decrease of energy consumption results from the decrease of frequency of double-thread. From the above analysis, we obtain the following conclusion: The energy consumption of double-thread is relatively small compared with single-thread.

(2) Now we consider the energy consumption under the same thread calculated locally and adopted by offloading strategy respectively, as shown in Fig. 3. This figure tells us that $E_{soc}$ is higher than $E_{sc}$ with the delay is small. However, energy consumption of offloading is lower than the calculated as the delay increases. Offloading all computing to the cloud for execution sometimes increases the energy consumption of mobile devices when having poor channel transmission conditions or high communication costs. We’d better use local computation to minimize the energy consumption if the amount of computation is
small and less energy is used to transmit data wirelessly. In this example, the energy consumption used in offloading will increase due to the communication cost is high when the time delay is small. Offloading is the optimal scheme when the time delay increases and communication cost decreases. The offload calculation costs least when the time delay of a single-thread exceeds a certain threshold. Fig. 4 shows a comparison of $E_{d \text{off}}$ and $E_{d \text{ca}}$. It can be seen from the figure that under the same delay, the energy cost under offloading scheme is lower. As the delay changes, $E_{d \text{off}}$ is still lower than $E_{d \text{ca}}$. From the above analysis, the calculation frequency of double-thread as well as the corresponding power decreases correspondingly which result in the decrease of the energy consumption.

(3) Then we compared $E_{s \text{off}}$ and $E_{d \text{off}}$. Fig. 5 shows the contrast of the cost in two cases under specific circumstances. The figure demonstrates that under the same time delay, using double-thread offloading strategy greatly reduces the energy consumption of mobile devices. Combined with the analysis of the above two, the increase of computing threads reduces the running frequency and energy consumption.

In conclusion, combining with the chart Fig. 6, it can be concluded the analysis that $E_{d \text{off}} < E_{d \text{ca}} < E_{s \text{off}}$ when the time delay is 158ms–278ms, and $E_{d \text{off}} < E_{d \text{ca}} < E_{s \text{off}} < E_{s \text{ca}}$ when the delay more than 278ms. At this point, the most energy-saving scheme is double-thread offloading scheme.

6. CONCLUSION
In this paper, we mainly study the difference between single-thread offloading and double-thread offloading. Section division, local offloading and linear time searching algorithm are employed to achieve the optimal offloading scheme, and the energy consumption under local computation and offloading of single-thread and double-thread are compared. From the results, we can have a conclusion that the double-thread offloading strategy is the best scheme. However, this paper has not yet discussed the analysis of collaborative computing in MTC and mobile terminal devices, optimization algorithm and the use of MTC to optimize delay when the energy consumption is certain, which is remained as our future work.

ACKNOWLEDGMENTS
This work was supported by the Key Research Projects of Henan Higher Education Institutions in China under Grant No. 19A520037.

REFERENCES
[1] Kumar, K., Lu, Y.H. 2010. Cloud computing for mobile users: can offloading computation save energy? Computer, 43, (4), pp. 51–56
[2] M. Bolat, K. Kelsey, X. Li, and G. Gao. 2011. Source Code Partitioning in Program Optimization. IEEE 17th International Conf. on Parallel and Distributed Systems (ICPADS), pp. 56-63, Tainan.
[3] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden. 2009. Wishbone: Profile-based Partitioning for Sensor net Applications. 6th USENIX Symposium on Networked Systems Design and Implementation, pp. 395-408.
[4] L. Yang, J. Cao, S. Tang, T. Li, and A. Chan. 2012. A framework for partitioning and execution of data stream applications in mobile cloud computing. In Proc. 5th Int. Conf. CLOUD, Jun., pp. 794–802.
[5] E. Hyytia, T. Spyropoulos, and J. Ott. 2013. Optimizing Offloading Strategies in Mobile Cloud Computing. Cryptanalyst.
[6] Z. Cheng, P. Li, J. Wang, and S. Guo. 2015. Just-in-time code offloading for wearable computing. IEEE Trans. Emerg. Topics Comput., vol. 3, no. 1, pp. 74–83.
[7] X. Ma, Y. Cui, L. Wang and I. Stojmenovic. 2012. Energy optimizations for mobile terminals via computation offloading. Proceedings of 2nd IEEE International Conference on Parallel Distributed and Grid Computing, Solan, India. pp. 236-241.

[8] X. Chen, L. Jiao, W. Li, and X. Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw., vol. 24, no. 5, pp. 2795–2808.

[9] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li. 2019. Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing. IEEE Trans. Mobile computing., vol. 18, no. 2.

[10] Y. Tao, Y. Zhang and Y. Ji. 2015. Efficient computation offloading strategies for mobile cloud computing. 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, pp.626-633.

[11] S. Yang, Z. Wang, Z. Han and L. Yan. 2019. Cooperative scheduling of multi-core and cloud resources: Multi-thread based mobile cloud computing offloading strategy. IET Communications.DOI=https://digital-library.theiet.org/content/journals/10.1049/iet-com.2019.0270

[12] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan. 2014. A survey of mobile cloud computing application models,” IEEE Commun. Surveys Tut., vol. 16, no. 1, pp. 393–413, 1st Quart.