Maximizing the User's Benefit in the Mobile Cloud Computing

Ning Wang
Dept. of Computer and Information Sciences
Temple University, USA
ning.wang@temple.edu

Jie Wu
Dept. of Computer and Information Sciences
Temple University, USA
jiewu@temple.edu

ABSTRACT

The increasing task computation complexity and limited battery has become a serious concern for smartphones. To reduce the task computation delay and save the smartphone battery usage, there have been many efforts to offload the tasks from the mobile device to the remote cloud with a much higher computation ability. However, offloading tasks to cloud will cause extra transmission delays and energy consumption. In reality, timely task execution is very important because the task's utility decays with time. Therefore, there exists a delay-energy trade-off. This paper addresses the aforementioned challenge. The smartphone should take advantage of the cloud in high computation speed so that the smartphone can achieve the maximum utility with limited battery. We get a 2-approximation schedule on expectation by using a LP rounding algorithm. The real trace experiments show the effectiveness of the proposed algorithms.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies; D.4.1 [**Process Management**]: Scheduling; C.2.2 [**Network Protocols**]: Routing protocols

1. INTRODUCTION

Mobile cloud computing (MCC), which offloads the computation complexity tasks from mobile devices to the cloud, has received a substantial amount of research attention in recent literature. However, offloading tasks to the cloud is not free. Though offloading tasks to the cloud can reduce the computation delay of the task, it creates transmission delay and will consume transmission energy. When the data input/output size is large or the mobile signal is weak, cloud offloading will become energy-intensive. From the viewpoint of the user or the service provider, the task finishing time is an important metric [6]. It is because the usefulness of the data decays with time. Computation offloading may be instrumental in a wide variety of mobile applications, from natural language processing, e.g., Apple's Siri, face recognition applications, to augmented reality.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

S3'16, October 03-07, 2016, New York City, NY, USA
© 2016 ACM. ISBN 978-1-4503-4255-1/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2987354.2987368

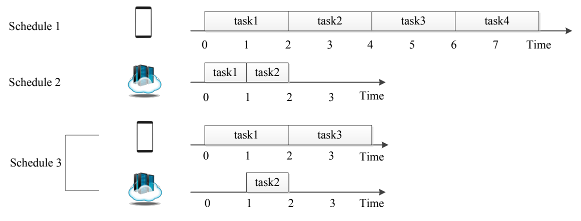


Figure 1: An illustration of the problem

As long as a task arrives, the smartphone has to either execute it in the local smartphone or transmit it into the remote cloud. The task utility decays along with the finishing time. A fundamental problem is to find a strategy which can fully use the limited battery of smartphone, so that the user's benefit, i.e., utility, can be maximized. Tasks arrive along with the time.

The proposed problem is non-trivial. An example is shown in Fig. 1. In this example, we assume that at the beginning of every second, a new task arrives. The tasks are identical. Each task has an initial utility of 10, and its utility decays 2 units per second. The overall energy budget of the smartphone is 4 units. If the smartphone executes a task locally, it consumes 1 energy unit, and the computation time is 2s. If the smartphone sends a task to the cloud, it consumes 2 energy units, and the transmission time is 1s. There are three scheduling strategies. Schedule (1) only uses the smartphone to execute the tasks and the overall utility that the user can get is 12. Schedule (2) only transmits the tasks to the cloud and the overall utility that the user can get is 16. Schedule (3) jointly uses the cloud and smartphone to execute the tasks and the overall utility that the user can get is 20. However, if the task utility decays 1 per second, these three schedules will earn $26, 18, 25$, respectively. The optimal schedule changes.

The contributions of this paper are twofold:

- To the best of our knowledge, we are the first to consider the user's benefit maximization in MCC with energy constraints and task utility decays.
- An LP rouding algorithm with a 2-approximation performance ratio is proposed.

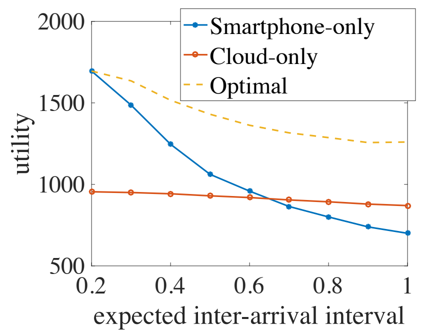


Figure 2: An example to illustrate the energy delay trade-off

2. PROBLEM FORMULATION

2.1 Model

In this paper, we assume that a smartphone has a certain battery budget B, e.g., 1000 mAh. Due to the limited bandwidth between the smartphone and the cloud, computation offloading (i.e., transmitting data to the cloud) will cause a certain data transmission delay. In reality, the cloud's computation ability is much higher than that of the smartphone's. Therefore, we ignore the task processing time in the cloud.

Once a task has arrived, the smartphone has to either execute it in the local smartphone or transmit it into the remote cloud until the battery of the smartphone is exhausted. The smartphone can only execute or transmit a task one at a time. If the computation or the transmission interface is busy, the task is put into the waiting queue. We assume all tasks are non-preemptive, i.e., once a task is assigned, it cannot be interrupted.

Let us use J to denote all the tasks that are executed, a_j and f_j denote the arrival time and the finishing time of task j, respectively. Task j's utility decays w_j per second until it reaches the zero. Initially, task i's utility is U_j. For each task, there are two execution options: (1) local execution or (2) cloud execution. Let use x_{ij} to denote task j's execution option, where $i = 1$ and $i = 2$ means that the task is executed at the local smartphone and the cloud device, respectively. In addition, let us denote p_{ij} as the processing time of a task (execution time and transmission time) at the smartphone and the cloud, respectively. e_{ij} is the corresponding energy consumption that is used for task j to be executed on device i, respectively. Note that the waiting time for a task is not included in the processing time.

2.2 Problem Formulation

Based on the model in the Section 2.1, the proposed problem can be formulated in the following:

$$\begin{aligned} &\max \quad \sum_{j\in J} U_j - \sum_{i\in J} w_j(f_j - a_j) \\ &\sum_{i=1}^{2}\sum_{j\in J} x_{ij} = 1, \qquad\qquad (1) \\ &\sum_{i=1}^{2}\sum_{j\in J} x_{ij}\cdot e_{ij} \le B, \quad x_{ij}\in\{0,1\}, \end{aligned}$$

where the first constraint means that each arrival task must be executed and the second constraint means that the overall energy cannot exceed the energy constraint. There is a delay-cost trade-off. Offloading tasks to the cloud might lead to a smaller delay but will cost more energy. An illustration of the trade-off in the proposed problem is shown in Fig. 2. When the task arrival rate is low, the task can be executed in the local smartphone and it will not lead to a large utility decay, since there are not many queued tasks. The extra energy can be used to execute more tasks. However, when the task arrival rate is high, this schedule executes the maximal number of tasks, but the overall utility is smaller than transmitting all the tasks to the cloud. The target of this paper is to find a proper scheduling method which can always achieve high overall utility in different scenarios.

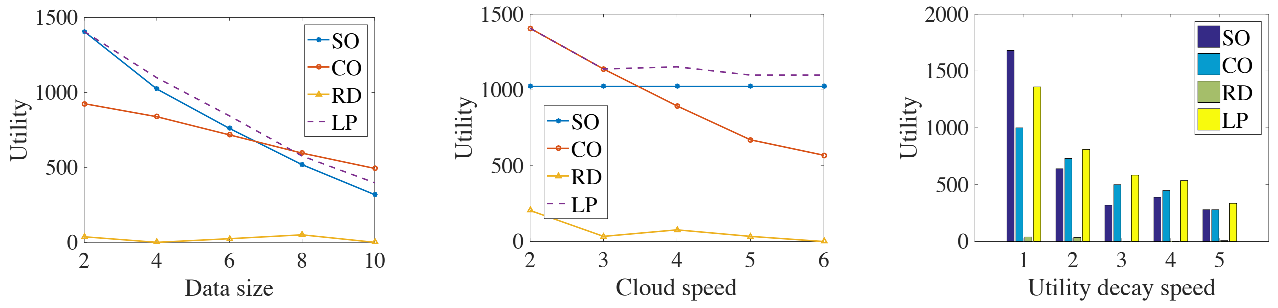
3. SOLUTION

The proposed problem is NP-hard, since it can be reduced into a classical scheduling problem. Therefore, instead of solving the original problem in Eq. 1, we can relax the proposed problem into the following linear programming problem. In the relaxed linear programming problem, tasks are preemptible, and a task may use the capacity of more than one device at a time. To illustrate the preemption task scheduling, let us use a new variable y_{ijt} that represents the amount of time task j that is processed on device i with the time interval $(t, t+1]$. Therefore, it is clear that $\frac{y_{ijt}}{p_{ij}}$ fraction of the task is being processed on device i within the time interval $(t, t+1]$. The LP relaxation is as follows:

$$\begin{aligned} \max \quad & \sum_{j\in J} U_j - \sum_{j\in J} w_j(f_j - a_j) \\ \text{s.t.} & \sum_{i=1}^{2}\sum_{t=a_i}^{T} \frac{y_{ijt}}{p_{ij}} = 1, && \forall j, \\ & \sum_{j\in J} y_{ijt} \le 1, && \forall i\&t, \\ & f_j \ge \sum_{i=1}^{2}\sum_{t=a_i}^{T} \left(\frac{y_{ijt}}{p_{ij}}\left(\frac{2t+1+p_{ij}}{p_{ij}}\right)\right) && \forall j, \qquad (2) \\ & f_j \ge \sum_{i=1}^{2}\sum_{t=a_i}^{T} y_{ijt} && \forall j, \\ & \sum_{i=1}^{2}\sum_{j\in J} e_{ij}\frac{y_{ijt}}{p_{ij}} \le B \\ & y_{ijt} \ge 0, && \forall i, j, \&t. \end{aligned}$$

The constraint (1) ensures that every task is fully processed. The constraint (2) expresses that each time the smartphone can execute or transfer at most one task at a time until time T. For (3), consider an arbitrary feasible schedule in which task j is being continuously processed between time $f_j - p_{ij}$ and f_j on device i, which is a lower bound. Then, the left-hand side of (3) corresponds to the real finishing time if we assign the values to the LP variables y_{ijt}; The right-hand side of (4) equals the processing time of task j in the schedule and is therefore a lower bound on its finishing time. The constrain (5) is the energy constraint.

Then, the LP rounding algorithm is as follow: first, we calculate the solution of the LP relaxation problem by Eq.



(a) data size (b) transmission cost (c) decay speed

Figure 3: Performance comparison of our algorithm in Sina Weibo Dataset.

2. Then, we assign task j to a device-time pair (i, t), where the device-time pair is chosen from the probability distribution that assigns task j to (i, t) with probability $\frac{y_{ijt}}{p_{ij}}$. We schedule each task on each device i that the tasks that were assigned to it as early as possible in order of nondecreasing t_j. The LP rounding algorithm has an expected 2-approximation ratio. The detailed proof is provided in [1].

4. PERFORMANCE EVALUATION

4.1 Experiments Setting

We use the real trace from the Sina Weibo [4] to demonstrate the effectiveness of the proposed algorithm. Based on the real situation, we set the following experimental parameters in our experiments. The average data size is 2 MB. The processing time for a task in smartphone is 1 MB per second, the corresponding energy consumption is 500 mW. The transmission bandwidth for smartphone is 2 MBps, and the corresponding energy consumption is 2000 mW.

4.2 Algorithm Comparison

We propose three comparison algorithms: the Cloud-only (CO) algorithm, which only utilizes the cloud for computation, referred the AllServer algorithm in [5]. Smartphone-only (SO) algorithm, which only utilizes the smartphone for computation, and is referred to as the AllMobile algorithm in [5]. We further propose a Random (RD) algorithm, which randomly assigns the new task to a device.

4.3 Experimental Results

The experimental results from the Weibo dataset are shown in Fig. 3, the LP algorithm can always achieve best performance in most scenarios, and it achieves more than 90% the performance in all the scenarios. However, the SO and SO algorithm can only achieve good performance in certain scenarios. That is, the SO algorithm achieves a good performance when the task utility decay speed is low. Conversely, the CO algorithm achieves a good performance when the task transmission cost is low. As for the random algorithm, its performance is really poor in most of case, since it cannot take advantage of smartphone or cloud execution.

5. RELATED WORKS

In the literature, there are many cloud offloading models used in different scenario [2, 3]. However, many of them are not general and the task finishing time matters in reality. A common assumption behind them is that the cloud offloading is always benefit, so that we should try to use as much as possible [8]. Another assumption is that the execution delay of a task is always assumed to be a constant value [7].

6. CONCLUSION

The limited battery and the timely execution requirement of the mobile device is an important requirement in the mobile cloud computing. In this paper, we consider the limited battery in the smartphone and the task utility decay scenario in reality. These two practical characters in our model distinguish our work from the existing works. We discuss the corresponding problems and propose a LP rounding solution. Real trace experiment results verify the effectiveness of the proposed algorithms.

7. REFERENCES

[1] https://www.dropbox.com/s/33ogl6ekeslawab/texstudio_proof.pdf?dl=0.

[2] E. Angel, E. Bampis, and F. Kacem. Energy aware scheduling for unrelated parallel machines. In *Proceedings of the IEEE GreenCom*, 2012.

[3] W. Chang and J. Wu. Progressive or conservative: Rationally allocate cooperative work in mobile social networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):2020–2035, 2015.

[4] K.-w. Fu, C.-h. Chan, and M. Chau. Assessing censorship on microblogs in china: Discriminatory keyword analysis and the real-name registration policy. *IEEE Internet Computing*.

[5] Y. Geng, W. Hu, Y. Yang, W. Gao, and G. Cao. Energy-efficient computation offloading in cellular networks. In *Proceedings of the IEEE ICNP*, 2015.

[6] D. E. Irwin, L. E. Grit, and J. S. Chase. Balancing risk and reward in a market-based task service. In *Proceedings of the IEEE HPDC*, 2004.

[7] X. Wang, J. Wang, X. Wang, and X. Chen. Energy and delay tradeoff for application offloading in mobile cloud computing. 2015.

[8] L. Xiang, S. Ye, Y. Feng, B. Li, and B. Li. Ready, set, go: Coalesced offloading from mobile devices to the cloud. In *Proceedings of the IEEE INFOCOM*, 2014.