International Journal of Computer Applications Technology and Research
Volume 6–Issue 12, 493-497, 2017, ISSN:-2319–8656

Energy Optimization using Cloud Offloading Algorithm

Jyothi T
Assistant Professor
Department of ISE
GSSSIETW, India

***Abstract*—Computational offloading is an effective method to address the limited battery power of a mobile device, by executing some components of a mobile application in the cloud. In this paper, a novel offloading algorithm called 'Dynamic Programming with Hamming Distance Termination' (denoted DPH) is presented. The algorithm uses randomization and a hamming distance termination criterion to find a nearly optimal offloading solution quickly. The algorithm will offload as many tasks as possible to the cloud when the network transmission bandwidth is high, thereby improving the total execution time of all tasks and minimizing the energy use of the mobile device. Furthermore, the algorithm is extensible to handle larger offloading problems without a loss of computational efficiency.**

Keywords —***Mobile Cloud Computing; Dynamic Programming; Computational Offloading; Randomization; Hamming Distance; Energy-Efficiency***

I. INTRODUCTION

Computation offloading is a method where some of the computational tasks of a mobile application can be offloaded to run on remote servers in the cloud, in order to save energy [1] [2]. However, the problem of partitioning the application tasks for offloading is NP-complete in general. The main goal of the offloading algorithm is to minimize the overall energy used by the mobile application, while meeting an execution time constraint.

A task to be offloaded must be transmitted over a wireless access network, and the time-varying wireless transmission bandwidth must be considered. An adaptive offloading algorithm can determine the offloading decisions dynamically according to a changing wireless environment.

Reference [3] presented a system that enables energy-aware offloading of mobile tasks to the

cloud called MAUI. Further improvements were proposed in CloneCloud [4] and Thinkair [5]. In all cases, the partitioning problem results in an integer programming problem which cannot be solved efficiently. A Dynamic Programming (DP) algorithm was proposed in [6], where a two dimensional DP table was used. However, this scheme did not consider an execution time constraint when computing the offloading decisions, although this time constraint is an important issue for many interactive applications . Furthermore, a backtracking algorithm was needed to find the final decisions, which was time consuming.

Reference [7] provided a dynamic programming approach which builds a three-dimensional programming table and requires pseudo polynomial time complexity [7]. However, it doesn't consider the energy consumed in the mobile device which is an important criteria for mobile devices.

In this paper, an innovative dynamic programming algorithm called DPH is proposed. Dynamic programming is an optimization approach that transforms a complex problem into a sequence of simpler problems. The DPH algorithm introduces randomization, i.e., we generate random bit strings of 0s and 1s periodically and utilize sub-strings when they improve the solution (similar to genetic optimization). We also fill the dynamic programming table in a creative way to avoid the extra computation for common sub-strings.

The algorithm can find a nearly optimal solution after several iterations. It uses a Hamming distance criterion to terminate the search to obtain the final decision quickly. The hamming distance termination criterion is met when a given fraction of tasks are uploaded. The final solution depends upon the wireless network transmission bandwidth and the computational power of the CAP and cloud servers.

The remainder of the paper is organized as follow: Section II provides the system model and problem formulation. Section III presents the proposed algorithm which is based on the dynamic programming table. The paper concludes in section IV.

II. SYSTEM MODEL AND PROBLEM FORMULATION

Consider an application consisting of some un-offloadable (i.e., local) tasks and N offloadable tasks. Normally, local tasks include those that directly handle user interaction, access local I/O devices or access specific information on the mobile device. Therefore, local tasks must be locally processed by the mobile user. We can merge all the local tasks into one task [13]. In [14], an example of a face recognition problem which is composed of eight offloadable tasks and one local task is presented.

A. Network Model

We consider a handheld mobile device with N independent tasks that can be executed locally or transferred to cloud for execution as shown in Fig. 1. We assume that a WiFi wireless network is available for the mobile device, but the network transmission bandwidth can change dynamically. Typically,wireless interference and network congestion will dynamically change the network transmission bandwidth. The mobile device needs to decide whether each task should be processed locally or offloaded, according to the current wireless network transmission bandwidth. The time taken to transfer a task between a mobile device and the cloud through a wireless link is an important issue since a total execution time constraint for all tasks exists. Therefore, the dynamic programming algorithm must consider the current wireless network bandwidth when computing a decision.

For task i, let Mi ∑ {0, 1} be an execution indicator variable. Let Mi = 1 if task i is executed at the mobile device and 0 otherwise. If it is executed locally, the energy consumption is Eli. Eri is the energy consumption of the mobile device when the task i is executed on the cloud, and Eti is the energy-cost of transmitting task i to the cloud server.

Variable Tli is the local execution time to process task i, and Tri is the remote execution time when task i is executed in remote cloud server. Variable Tti is the transmission time to transfer task i to the cloud server.

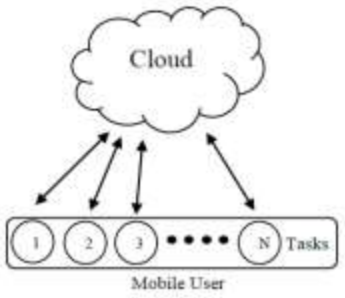


Fig 1: Network Model

It is clear that the transmission energy used to upload each task will depend on the network transmission bandwidth. Therefore, changes in wireless network bandwidth will affect the offloading decision. For example, if we assume that transmission time of each task is equal to the size of each task divided by the network transmission rate, then any variation in the transmission rate will affect the final decision of whether to offload this task or not.

The energy consumption function and its corresponding execution time are defined in (1) and (2):

$$E = \sum_{i \in N} (M_i El_i + (1 - M_i) Er_i + (1 - M_i) Et_i) \qquad (1)$$

$$T = \sum_{i \in N} (M_i Tl_i + (1 - M_i) Tr_i + (1 - M_i) Tt_i) \qquad (2)$$

The execution time T of all tasks must satisfy the following condition, where Tconstraint is the execution time requirement

$$T \le T_{cnostraint}$$

For simplicity, we use M = [M1, M2, MN] to denote a vector of binary offloading decisions. The problem that we want to solve is as follow:

$$\min \quad E$$
$$\text{Subject to:} \quad T \leq T_{constraint}$$

The number of combinations of binary values Mi to search for the optimal solution grows exponentially with the number of tasks. Our goal is to determine which tasks should be offloaded to the cloud server to minimize energy while meeting a mobile application's execution time constraint.

III PROPOSED ALGORITHM BASED ON THE DYNAMIC PROGRAMING TABLE

A. Innovative Way of Filling the Table

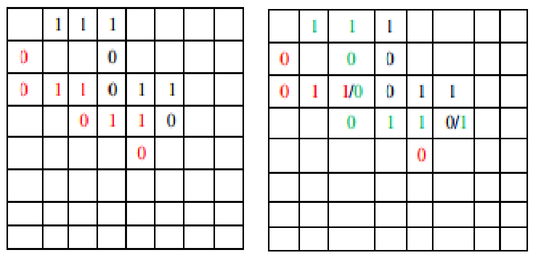
The proposed algorithm is called 'Dynamic Programming with Hamming Distance Termination' (DPH). In this scheme, we use an N*N table to store the bit-streams that show which tasks should be offloaded (where N is the number of tasks). For the first step, a random bit stream is generated that determines a first solution.

This stream is assigned to the table such that 1s are assigned to the next horizontal cell, and the 0s are assigned to the next vertical cell. If the first bit of the stream is 1, the starting cell is (1, 2) and if the first bit of the stream is 0, the starting cell is (2, 1).

This approach will avoid extra computations for common bit strings. A 2D 8*8 table is shown in table I. To clarify, assume that N = 8 and the first random stream is 11100110 (black numbers) or 00110110 (red numbers), (2 examples are given). Assume that the second random bit stream in each case is 11000111.

The starting cell of the second stream is (1, 2) since the first bit is 1. By following the aforementioned rules to fill the table, the resulting green stream is shown in table I.

Table I. How to fill the table for 2 examples.



	1	1	1				
0			0				
0	1	1	0	1	1		
		0	1	1	0		
				0			

	1	1	1				
0		0	0				
0	1	1/0	0	1	1		
		0	1	1	0/1		
				0			

Whenever a bit stream is generated randomly, we calculate the consumed energy and time of each cell (i.e., each task) in the table, and also at the same time calculate the total energy and execution time of this bit stream. However, if a random bit stream is generated which has some common cells with an existing string in the table, we only calculate the total energy of new string until the first common cell and then compare this new total energy with the existing total energy at this cell.

If the new total energy at this specific cell is less than the previous one, we keep the new sub-string and delete the old sub-string, and replace the total-energy and cell-energy of this cell with new amounts. We then update the energy and execution time of the remaining cells for the existing bit stream, based on the new values at this common cell. Otherwise, if the total energy of the existing bit stream is less than that of the new bit stream at the common cell, we will perform the same procedure while keeping the existing stream.

Every time a new stream is generated, we keep tracking the arrangement of the stream in the table. We terminate and accept a solution which has Hamming distance larger than a given threshold from an all 1 stream. The all 1 stream denotes the case where all components are executed locally. For example, the algorithm can terminate after K=20 iterations, or when 70% of the tasks have been offloaded. This heuristic termination criterion yields good results.

B. Algorithm of the Proposed Scheme

The algorithm of our proposed scheme is shown in table II:

Table II. Algorithm of Proposed Method (DPH)

```
1.Initialize Energy and Time matrixes and set the time constraint
(Tconstratint) and Transmission Rate
2.for iteration = 1 to iteration_num
3.generate a random bit stream
4. check the first bit to specify the starting cell in the table
5.  for i = 1 to N-1
6.     Put each bit of the bit stream in the correct
       position in table
7.   Calculate the self-Energy and time of each cell
     and the total energy and time.
8.      if this specific cell in table is visited before
          compare the new Total Energy of this cell
         with the previous one
9.          if the new Total Energy of the cell is less
              than the previous one
10.           Replace the total energy and time of this
              Cell with the new calculated amounts.
11.           Update the remaining amounts in the
              Remaining cells of the previous bit
              stream based on the new amount of this
              common cell.
12.            Calculate the energy and time of the
               Remaining bits of the new bit stream.
13.          Track the position of all bits in the table in
              a  matrix
14.      else
15.          Keep the previous total energy and time in
             the cell.
16.         Calculate the Energy and time of the
             remaining cells of the new stream based
             on the existing amount of this cell.
17.          Track the position of all bits in the table in
              a  matrix
18.      End if
19.    End if
20. End for
21.
22. if  Number of bits in table = N & Etotal < Emin
         & Ttotal < Tconstraint & hamming distance
         criterion is  met
23. return Etotal, Ttotal
24. end if
25. End for
```

IV. CONCLUSION

A mobile device must decide which computational tasks of a mobile application should be offloaded in order to minimize energy consumption while satisfying an execution time constraint. An efficient heuristic algorithm called DPH to solve this optimization problem is proposed, which uses dynamic programming combined with randomization.

It also uses a hamming distance as a termination criterion. Simulation results show that the proposed DPH algorithm can find nearly optimal solutions and it can be easily handle larger problems without losing computational efficiency. The DPH algorithm can be used dynamically, to adapt to the changes in the network transmission rate. The algorithm will tend to offload as many tasks as possible when the network performance is good, resulting in a rapid convergence to a near optimal solution with a very fast execution time.

REFERENCES

[1] Z. Li, C. Wang, and R. Xu, “Computation offloading to save energy on handheld devices: a partition scheme,” in Proc. International Conf. Compilers, Architecture, Synthesis Embedded Syst., pp. 238–246, 2001.
[2] P. Rong and M. Pedram, “Extending the lifetime of a network of batterypowered mobile devices by remote processing: a Markovian decisionbased approach” , Design Automation Conf., pp. 906–911, 2003.
[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “MAUI: Making smartphones last longer with code offload,” in Proc. ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), pp. 49–62, 2010.
[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A Patti, “Clonecloud: Elastic execution between mobile device and cloud,” in Proc. ACM Conference on Computer Systems (EuroSys), pp. 301–314, 2011.
[5] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, “Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading,” in Proc. IEEE International Conference

on Computer Communications (INFOCOM), pp. 945–953, 2012.

[6] Y. Liu, M. J. Lee, “An Effective Dynamic Programming Offloading Algorithm in Mobile Cloud Computing System”, IEEE WCNC'14, pp.1868 – 1873, 2014.

[7] A. Toma, J. Chen, “Computation Offloading for Frame-Based Real- Time Tasks with Resource Reservation Servers”, IEEE Euromicro Conference on Real-Time Systems, pp. 103-112, 2013.

[8] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, “Adaptive offloading for pervasive computing,” IEEE Pervasive Comput., vol. 3, no. 3, pp. 66–73, 2004.

[9]Available: http://darnok.org/programming/face-recognition/.

[10] A. Kammerdiner, P A. Krokhmal, P. M. Pardalos, “On the Hamming distance in combinatorial optimization problems on hypergraph matchings”, Springer Optimization Letters, Vo. 4, pp. 609-617, 2010.