A metaheuristic approach for application partitioning in Mobile System

S. M. Ferdous
Department of CSE
AUST,Dhaka-1208,Bangladesh

M. Sohel Rahman*
Department of CSE
BUET,Dhaka-1000,Bangladesh
Department of Informatics
King's College London

***Abstract*—Mobile devices such as smartphones are extremely popular now. In spite of their huge popularity, the computational ability of mobile devices is still low. Computational offloading is a way to transfer some of the heavy computational tasks to server(cloud) so that the efficiency and usability of the system increases. In this paper, we have developed a metaheuristic approach for application partitioning to maximize throughput and performance. Preliminary experiment suggest that our approach is better than the traditional all cloud and all mobile approach.**

I. Introduction

Computing technology is undergoing major advancement. The popularity of the mobile devices are increasing drastically. The usage of computers are not now limited to only desktops and Mainframes. With the advances in technologies of wireless communications and portable devices, mobile computing has become integrated into the fabric of our daily life. The application provided in this mobile devices ranges from surveillance, sensing to social networking. Many of these applications run on systems with limited resources. For example,mobile phones are battery powered. Environmental sensors have small physical sizes, slow processors, and small amounts of storage. Most of these applications use wireless networks and their bandwidths are orders-of-magnitude lower than wired networks. The gap between the demand of the complex programs or application and the supply of the computational power is increasing day by day.

The application of cloud services in the mobile ecosystem enables a newly emerging mobile computing paradigm, namely *Mobile Cloud Computing* (MCC). MCC offers great opportunities for mobile service industry, allowing mobile devices to utilize the elastic resources offered by the cloud. There are three MCC approaches [1]: 1) extending the access to cloud services to mobile devices; 2) enabling mobile devices to work collaboratively as cloud resource providers; 3) augmenting the execution of mobile applications using cloud resources, e.g. by offloading resource intensive computing tasks on mobile devices to the cloud. This will allow us to create applications that far exceed traditional mobile device's processing capabilities.

Of these three MMC approaches, the third approach, offloading can be a solution to augment mobile systems capabilities by migrating computation to more resourceful computers (i.e., servers/cloud). In this paper we have developed three optimization models for analyzing partitioning algorithm. The first optimization objective is to maximize throughput, the second one is to minimize the total execution time and the third one optimizes the energy consumption. Then we have developed a Greedy Randomized Adaptive Search (GRASP) metaheuristic algorithm to optimize the objective functions.

*On a Sabbatical leave from BUET.
978-1-4799-8126-7/15/$31.00 ©2015 IEEE

II. Related Works

Computational offloading is the most widely used technique to solve the resource poverty problem of mobile devices [2]. In [3], the author argued that cloud computing could potentially save energy for mobile users, but not all applications are energy efficient when migrated to cloud. Compared with offloading a whole application into cloud a partitioning scheme is able to achieve a fine granularity for application partitioning problem. In [4], the author provides a comprehensive list of the algorithms for mobile computational offloading. But these algorithms do not utilize application partitioning. The application is not partitioned. Either it is run on mobile device or in cloud. Application partitioning algorithms are studied early in [5]. More application partitioning algorithm is proposed and analyzed in [6][7] [8]. Recently in [1], the author proposes a framework for data stream application partitioning. They have used genetic algorithm to find (so-called) optimal partition. The objective of their work is to maximize throughput.

III. Preliminaries

In this section we will describe the terms and notations used throughout the paper.

A. Application Model

We have adopted the application model proposed by [1]. The application is modeled as a specific data flow graph $G = (V, E)$, where the vertices $V = \{i| = 1, 2, \dots v\}$ represents its components and the edge set $E = \{(i, j) : i, j \in V\}$ represents the dependency between the components. s_i is the average number of CPU instructions required by component i to process one unit of data. $d_{i,j}$ presents the amount of data required to be transmitted on the channel (i, j) for one unit of data. The weight on a node i, denoted as w_i, represents the computational cost (time). The weight on an edge denoted as $c_{i,j}$ is the communication cost (time). Both wi and $c_{i,j}$ are measured by one unit of data.

Two dummy edges are added in the graph denoting $(0, 1)$ and $(v, v + 1)$ that represent the input amount of data from and the output amount of data delivered to the mobile device.

TABLE I. CONTROLLING PARAMETERS

Symbol	Explanation
p	CPU's capability of the mobile device
η	percentage of the ideal CPU on mobile
s_m	available CPU resource of mobile (ηp)
s_i	speed of the server CPU
B	Bandwidth of the wireless network

B. Model Parameters and Assumptions

The offloading decision is taken considering local computing resources and the wireless network quality. A list of parameters should be introduced to model the offloading decision. The parameters are listed on the Table I

The assumptions of our system are following:

- CPU resource is allocated equally to all running processes in mobile
- If a component is offloaded to cloud then the other components speed up in the mobile device. The speed up factor is $N/(N+1)$ where N is the number of components running before the offloading occur.
- The cloud does not have abundant resources. We define cloud speed s_s, where s_s is significantly larger than the mobile speed s_m.
- The total bandwidth B is shared by all crossing channels. A crossing channel is the edge on the graph whose one end point is in mobile and the other end point is in cloud.
- The critical component of the system are the crossing channels and the nodes running in the mobile device.
- Input data of the application acquired from the sensor on the mobile device and the output data should also be delivered to the mobile device.

A partition of the application graph is represented by a binary vector, X. The elements of X can either be 0 or 1 representing the component running on cloud or mobile respectively.

IV. OFFLOADING DECISION MODEL

In this section we will develop the offloading model. We will formulate the offloading problem as an optimization problem. We have developed three optimization model namely, maximization of the throughput, minimizing execution time and minimizing energy consumption.

A. Throughput Maximization Model

The throughput of a system the amount of data transferred from one place to another or processed in a specified amount of time. Like [1], we have defined critical component/channel, which represents the component/channel that has the greatest weight among all the components/channels. Assuming that all the channels' capacity is unlimited and whatever level of pipeline parallelism is allowed, the throughput of the data flow application is determined by the critical component/channel, which have the slowest speed to compute/transfer the data. So the throughput is defined as $Tp = 1/t_p$

$$t_p = max\{max_{i\in V}\{w_i\}, max_{(i,j)\in E}\{c_{i,j}\}\}$$

We have slightly modified the optimization model proposed in [1]. The modified model incorporates the server speed. The model is:

Let $X = \{x_i | x = 1, 2, \dots v\}$ where $x_i = \{0, 1\}$, denote the partition. The throughput is defined as,

$$max_X\ Tp = \frac{1}{maxtcomp_m(X), tcomp_s(X), tcomm(X)} \quad (1)$$

where,

$$tcomm(X) = \frac{1}{B}[(1-x_1)d_{0,1} + (1-x_v)d_{v,v+1} + \sum_{(i,j)\in E}(x_i - x_j)^2 d_{i,j}]$$

and

$$tcomp_m(X) = max_{x_i\in X}\{x_i \frac{s_i}{s_m}\sum_{i=1}^{v} x_i\}$$

$$tcomp_s(X) = max_{x_i\in X}\{x_i \frac{s_i}{s_s}\sum_{i=1}^{v} x_i\}$$

here $tcomp_m(X)$, $tcomp_s(X)$ and $tcomm(X)$ are the critical computational cost on the mobile, critical computation cost in the server and critical communication cost over the crossing channels.

B. Execution Time Minimization Model

The total execution time is a measure of performance in mobile device. The aim is to minimize the total execution time. To find the total execution time of a partition we have to consider two things. The execution time of the components those run in the mobile device and the execution time of the components running on server. Given a partition X, we define TE_m, the total execution time on mobile device as the,

$$TE_m = \sum_{i=1}^{v}\{x_i \frac{s_i}{s_m}\sum_{i=1}^{v} x_i\} \quad (2)$$

The total execution time in server TE_s is defined as follows,

$$TE_s = \tfrac{1}{B}[(1-x_1)d_{0,1} + (1-x_v)d_{v,v+1} + \sum_{(i,j)\in E}(x_i - x_j)^2 d_{i,j}] + \sum_{i=1}^{v}\{x_i \frac{s_i}{s_s}\sum_{i=1}^{v} x_i\} \quad (3)$$

The objective function to minimize total execution time of the system can be formulated as,

$$min_x\ TE = max\{TE_m, TE_s\} \quad (4)$$

C. Energy Minimization Model

Energy is a primary constraint for mobile systems. Now-a-days the use of smartphones are not limited in only communication device. They are used in various purposes like watching videos, taking photographs and so on. As a result, these systems will likely consume more power and shorten the battery life. Offloading can also increase battery life of the devices. Let p_m is the power consumption of mobile system in unit time. Given a partition X, the total power needed to execute the components in mobile can be adjusted from equation 2 as,

$$TP_m = p_m \sum_{i=1}^{v} \{x_i \frac{s_i}{s_m} \sum_{i=1}^{v} x_i\} \tag{5}$$

Now suppose p_c be the power required to send data from the mobile system over the network. After sending the data, the system needs to poll the network interface while waiting for the result of the offloaded computation. During this time, the power consumption is p_k. The total power consumed int the process is,

$$\begin{aligned} TP_s = p_c \frac{1}{B}[(1-x_1)d_{0,1} + (1-x_v)d_{v,v+1} \\ + p_k \sum_{(i,j)\in E} (x_i - x_j)^2 d_{i,j}] + \sum_{i=1}^{v}\{x_i \frac{s_i}{s_s} \sum_{i=1}^{v} x_i\} \end{aligned} \tag{6}$$

So the objective is to minimize total power of the system:

$$min_x\ TP = max\{TP_m, TP_s\} \tag{7}$$

V. GRASP ALGORITHM FOR COMPUTATION OFFLOADING

In this section we will describe a simple metaheuristic algorithm to generate partition. First we will develop the algorithm. Then we will explain the pseudocode in detail.

The main motivation of our partition algorithm is to favor in offloading the components those have heavy computation and low communication cost. We consider a high computational node in data flow graph which is currently running in mobile. We offload the node in server. The question is which neighbor of this node should be offloaded? We favor the neighbor whose communication cost and computation cost is high.

Given a data flow graph G, we randomly select a high computational node a which is currently in mobile. Let $N(a)$ be the neighbor of the node. We can assign probability of offloading to each of the neighbor node. The probability to select a node $i \in N(a), i \neq 0$ as an offloading component is,

$$p(i) = \frac{w_i c_{a,i}}{\sum_{j\in N(a), j\neq 0} w_j c_{a,j}} \tag{8}$$

The detailed pseudocode is provided in Algorithm 1

A. Pseudocode

A GRASP solution can be divided into two parts. The first part is a greedy randomized solution and the second part is the local search on the solution. In our approach we have first constructed a profile of solution. Each solution is constructed by transferring a node from mobile to server. From this solution profile we choose the solution as a greedy output of the algorithm whose throughput(execution time/energy) is maximum(minimum).

Algorithm 1 GRASP

```
1: set parameters $v$,$s_m$,$s_s$,$m$
2: Generate a random application graph, $G$
3: $X \leftarrow$ vector of all ones
4: while $itCounter < itLimit$ do
5:     $rcl \leftarrow$ Sorted(descendent) computational cost vector of length $m$(components currently at mobile).
6:     $a \leftarrow$ randomly select a node from $rcl$
7:     $X(a) \leftarrow 0$
8:     $Profile(counter) = X$
9:     while $counter < v$ do
10:        if $N(a)$ is empty then
11:            $rcl \leftarrow$ Sorted(descendent) computational cost vector of length $m$ (components currently at mobile).
12:            $a \leftarrow$ randomly select a node from $rcl$
13:        else
14:            construct pdf for $N(a)$ by equation 8
15:            $a \leftarrow$ randomly select a node from pdf
16:        end if
17:        $X(a) \leftarrow 0$
18:        $Profile(counter) = X$
19:    end while
20:    $bestSol \leftarrow$ select the configuration from $Profile$ that maximize(minimize) throughput(execution time/energy consumption) according to equation (1 - 7)
21:    localsearch on $bestSol$
22: end while
```

At first (Line 1 and 2) we have set the parameters of the graph and construct the random graph. Here v is the number of nodes in the application graph, and m is the length of *rcl*. The construction procedure of the random graph will be discussed in next section. The GRASP algorithm will run for a certain iteration. We have to first find the available nodes (the nodes are in mobile). It is to be noted that initially all the nodes are in mobile. The available nodes are sorted descending order of their computational cost. We define *Restricted candidate list*(rcl) as the first m nodes on the sorted list. Next a node from the *rcl* is selected randomly.

This node("a") is then transferred to cloud (setting $X(a) = 0$). Now a neighbor of this node is to be transferred to cloud. Two possible events can occur. If there are not any available neighbor of "a" (all of the neighbor nodes are in cloud), then we will again generate the *rcl* from the available nodes and select a random node. Otherwise we assign probability to the available nodes on the neighbor list of "a" by equation 8. Next a node is selected from this probability distribution function. The algorithm continued to run until the iteration counter reached its limit.

VI. PRELIMINARY EXPERIMENT

Due to shortage of time in the experiment, we are presenting only the preliminary experiment result on throughput and execution time metric.

TABLE II. VALUES OF THE CONTROLLING PARAMETERS

No.	$G(v, d_{out}, CCR)$	B	ηp
1.	$G(*, 3, 1)$	1	1
2.	$G(30, 3, *)$	1	1
3.	$G(30, 3, 1)$	*	1
4.	$G(30, 3, 1)$	1	*

A. *Methodology*

The algorithm is evaluated on the various combination of the controlling parameters. The input of the algorithm is a random application graph and a set of parameters. The random application graph is generated using the technique discussed in [9]. The graph is controlled by the following parameters: 1) number of nodes; 2) average out-degree; and 3) communication to computation ratio(CCR). CCR is defined as the ratio of the average communication time to the average computation time which is shown in equation. High CCR value indicates communication intensive application.

$$CCR = \frac{[d_{0,1} + d_{v,v+1} + \sum_{(i,j)\in E} d_{i,j}]/[(e+2)\times B]}{(\sum_{i\in V} s_i)/(v \times s_m)} \quad (9)$$

here e is the number of edges. A group of simulation has been done to test the effect of the controlling parameters. Table II listed the parameters. In each simulation, we choose one parameter as the variable, which is indicated by '*', while assigning other parameters as constant values. we have set the length of the rcl, m as 5 and the iteration limit set to 400.

B. *Results*

In our preliminary experiment, we have done a group of simulations keeping constant all other parameters except the application graph size (v), the bandwidth(B), the resource of mobile device($s_m(=\eta p)$) and the CCR respectively. Figure (1 - 4) represent the simulation result to maximize the throughput where as the figure (5 - 8) show the simulation result on the second objective function that is to minimize the total execution time.

We compare our partition result with the other approach namely keeping all the components in the mobile (all mobile) and offloading all the component into cloud (all cloud). From our analysis it is seen that our partition approach provides better throughput (more than 2X than the extreme two approaches). The figure (1 - 4) show that the partition line is always above the other two lines depicting better throughput in any configuration. On the other hand figure (5 - 8) represent partition line always below to the other two lines which shows the prove better performance.

VII. CONCLUSION

In this paper we have developed a metaheuristic approach to increase throughput and performance of mobile system by partitioning application. The preliminary experiments suggest that our approach is better than the no-partition approach. Future work includes a more robust experiment. The analysis on energy consumption model is needed to be done also. Real life mobile application can also be tested using this algorithm in future.

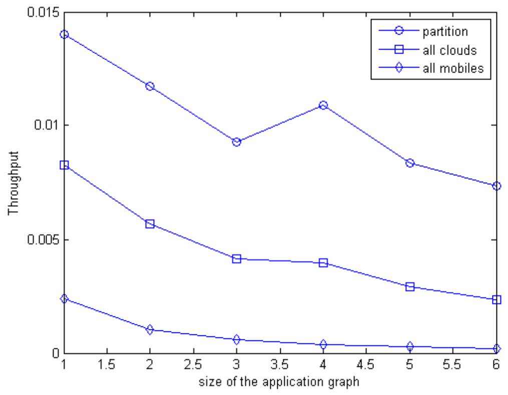

Fig. 1. Throughput Vs. application graph size

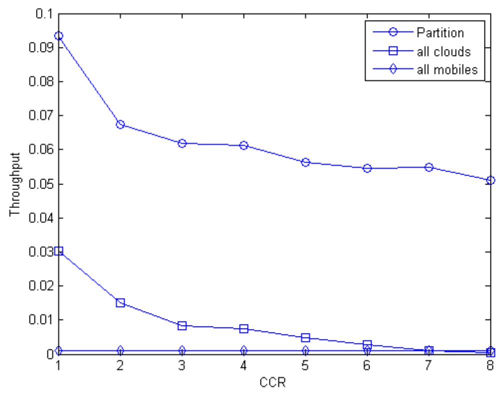

Fig. 2. Throughput Vs. CCR

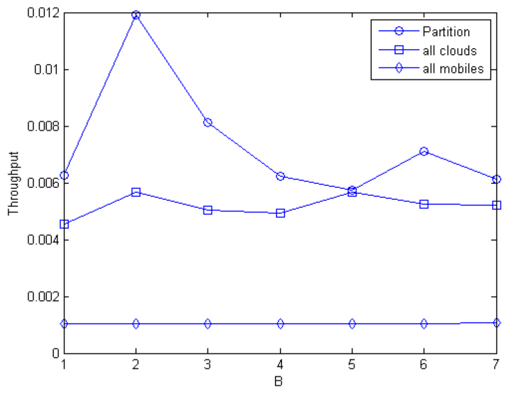

Fig. 3. Throughput Vs. Bandwidth

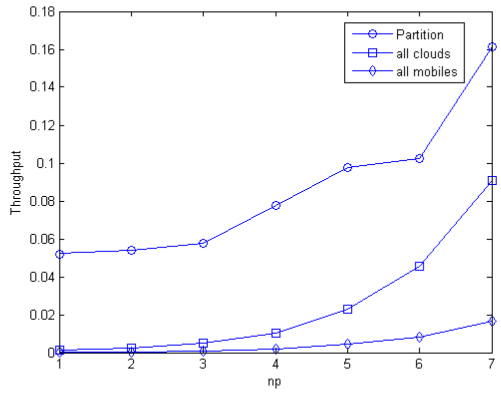

Fig. 4. Throughput Vs. available resource

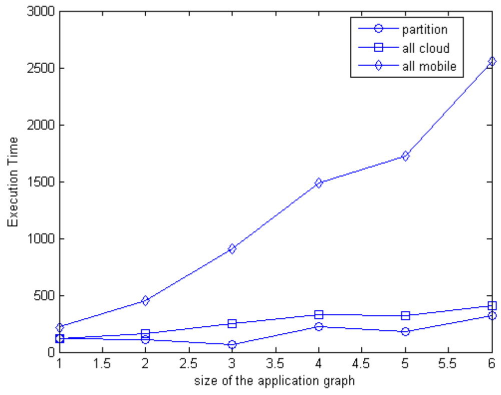

Fig. 5. Execution Time Vs. application graph size

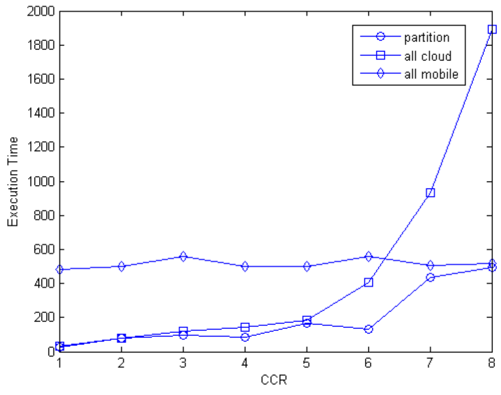

Fig. 6. Execution Time Vs. CCR

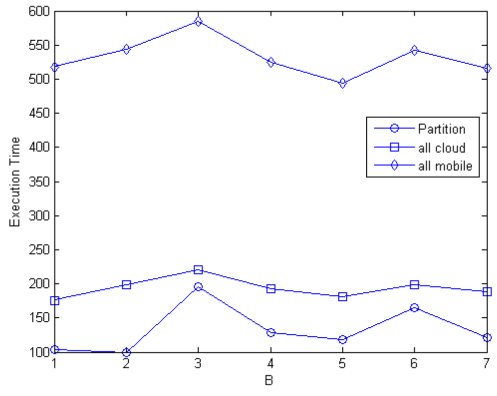

Fig. 7. Execution Time Vs. bandwidth

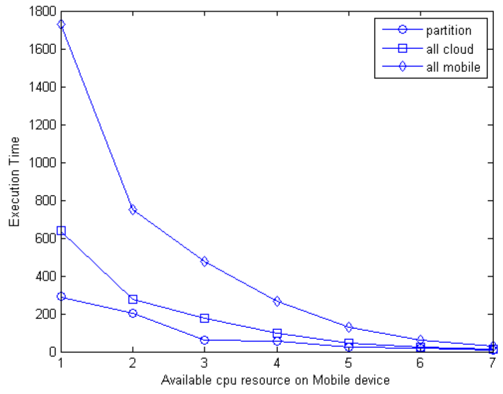

Fig. 8. Execution Time Vs. available resource

Acknowledgment

References

[1] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Apr. 2013. [Online]. Available: http://doi.acm.org/10.1145/2479942.2479946

[2] K. Yang, S. Ou, and H.-H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications," *Comm. Mag.*, vol. 46, no. 1, pp. 56–63, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2008.4427231

[3] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1109/MC.2010.98

[4] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1007/s11036-012-0368-0

[5] "Task allocation for distributed multimedia processing on wirelessly networked handheld devices," in *Proceedings of the 16th International Symposium on Parallel and Distributed Processing*, ser. IPDPS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 79–. [Online]. Available: http://dl.acm.org/citation.cfm?id=876874.878647

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966473

[7] 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 *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’10. New York, NY, USA: ACM, 2010, pp. 49–62. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814441

[8] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, “Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing,” *Mob. Netw. Appl.*, vol. 16, no. 3, pp. 270–284, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1007/s11036-011-0305-7

[9] H. K. T. Tobita, “A standard task graph set for fair evaluation of multiprocessor scheduling algorithms,” *Journal of Scheduling*, vol. 5, no. 5, pp. 379–394, 2002.