ISSN:0975-9646

An Effective Computation Offloading from Mobile Devices to Cloud

Jaya Ashok Suradkar[#1], Prof. R. D. Bharati[*2]

[#]*Department of Computer Engineering,*
Savitribai Phule Pune University,
DYPIET, Pimpri, Pune, India.

***Abstract*—Energy and time effectiveness is a primary consideration for smartphones or mobile devices. Cloud computing has the ability to conserve mobile device's energy by using the concept of computation offloading. Existing studies focus on offloading computation by assuming the relations among the factors like wireless bandwidth, the amount of computation to be performed, and the amount of data to be transmitted. Objective of Computation Offloading Decision Maker Framework (CODMF) is to reduce energy consumption and response time for a mobile device simultaneously. Execution of computation intensive or resource intensive task to be done locally on CPU and remotely on cloud, which provide a highly versatile execution platform for mobile or android applications. We conduct a real world application for finding road roughness, in order to measure the performance of CODMF. When the task demands maximum energy consumption and time, task will be shifted to the cloud. In addition, we proposed application partitioning for a CODMF which gives a smaller amount of decision rate for false offloading than previous methods. By offloading modules of an application, proposed system can achieve significant savings in battery consumption and in execution time.**

Keywords—computation offloading; cloud computing; Energy; Battery; response time; framework.

I. INTRODUCTION

In our daily life, mobile devices have become common entity. Energy or Battery is the only resource in mobile devices that cannot be restored immediately and needs external resources to be renewed. Computation offloading is a way to improve performance and save energy. Cloud computing provide services based on pay-as-you-use principle like water, electricity, telephone etc. and provides a model for "enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction", as NIST describes. Mobile cloud computing is a combination of cloud computing, wireless communication technologies, mobile web, location based services and computing devices. There are many mobile cloud applications such as android applications, biometric application, social media applications, mobile commerce application, mobile healthcare applications etc. are using software as a service model.

Several research efforts devoted towards computation offloading but most of the works have not considered the impact of computation offloading on response time, which leads in degradation of performance. Also, investigation needs to be done on energy consumption of retrieving and uploading data which may results in a high battery usage and can significantly reduce battery lifetime. Decision accuracy depends on various factors like memory bandwidth, bandwidth required for network transmission and CPU speed of a mobile device are to be considered in computation offloading for building a cost functions. Adaptability, Portability, Accuracy and Offload targets are important factors in computation offloading. Proposed CODMF framework with partitioning algorithm can achieve better performance in terms of execution time and battery usage.

In this paper we study about related work done on energy and time saving in section II. Implementation details for computation offloading framework in section III gives overview of a CODMF framework, proposed system, mathematical model, algorithm and experimental setup. In section IV we discuss expected results and finally conclude a paper in section V.

II. RELATED WORK

Optimizing energy and response time for mobile devices simultaneously are difficult design goals and very few research efforts have adopted on energy and time saving simultaneously. The authors in [1] presented a simple model to evaluate energy cost of wireless data transmission. They formulated data offloading through several wireless network interfaces as a "statistical decision problem" and presented several solutions to evaluate wireless network states and solved its issues. Authors in [2] showed that the network characteristics calculated through network profiling. They also denoted partitioning granularity in terms of a service, segment and module. SALSA [3] is a near-optimal algorithm for performing the energy-delay tradeoff in bandwidth intensive delay-tolerant smartphone applications. CloneCloud [4] uses a combine static analysis and dynamic profiling for automatic application partitioning. Their System solves design and implementation issues to meet basic augmented execution of mobile applications on the cloud, representing the whole transfer of control from the device to the clone and back. Also, combine partitioning, migration with merging, and on-demand instantiation of

partitioning for the same purpose. ThinkAir [5], is nothing but a computation offloading framework for offloading computation from mobile side to the cloud side. It adapts to varying bandwidth and connectivity changes during runtime. Application's source code needs simple modifications by using ThinkAir framework. Some authors presented eTime [6], to manage data for delay-tolerant applications which achieves energy efficient data migration or transmission by using cloud resources in mobile cloud computing. In [7], experimental driven approach to computation offloading tradeoffs is given. They presented that instead of offloading tasks to a cloud side, running particular tasks at local side is better choice. So authors proposed a new generic architecture for any mobile cloud computing application to automate the offloading decision and help these applications to optimize energy and time for the mobile device. Static Partitioning and adaption to varying bandwidth implemented by [8] and object-level is the choice of the fine grained offloading and its granularity. The first task scheduling work [9] focused on reducing energy consumption under a difficult completion time constraint for the graph of the task in the MCC environment. A novel algorithm proposed by authors to achieve energy savings by shifting tasks in between the local cores and the cloud while maintaining delay scheduling. A linear time rescheduling algorithm for the task migration used to minimize overall computation complexity effectively. In [10], ternary decision maker framework called TDM for computation offloading is designed and implemented. Survey of different computation offloading framework is given in [12]. The above work summarized as:

TABLE I. WORKS ON ENERGY AND TIME SAVINGS

Year	Related Work on both Energy and Time Savings		
	Contribution	***Decision***	***Partition***
2007	Context-sensitive energy-efficient offloading[1]	Static	No
2010	Dynamic partition between devices and clouds[2]	Dynamic	Yes
2010	Stable and adaptive link selection algoithm[3]	Dynamic	No
2011	Elastic execution between devices and clouds[4]	Static	Yes
2012	Dynamic resource allocation and parallel execution[5]	Dynamic	No
2013	Energy efficient data transmission strategy[6]	Dynamic	No
2013	Feasibility of mobile cloud systems in a real setting[7]	Dynamic	No
2014	Partition scheme taking the bandwidth as a variable[8]	Dynamic	Yes
2014	Energy and performance aware task scheduling[9]	Dynamic	Yes
2015	Offloading framework TDM[10]	Dynamic	No
2016	Proposed Work (Partitioning Algorithm [11] in TDM Framework [10])	Dynamic	Yes

Existing frameworks for cost models of computation offloading are shown in Table II.

TABLE II. EXISTING OFFLOADING FRAMEWORKS ON COST MODEL

Sr. No	Related Work		
	Contribution	***Observation***	***Limitation***
1	Computation Offloading in Handheld Devices to Coprocessors and Clouds[10]	High Accuracy, Shorter Response Time, Less Energy Consumption	Does not focuses on experimental analysis of different Mobile devices, apps and network interfaces
2	CloneCloud[4]	History-based Profiling, Implementation on Dalvik VM(Android)	Does not provide generalized calculations for real networks and device conditions, All offloading scenarios need to considered, Pre-calculated Partitioning
3	MAUI [13]	History-based Profiling, Implementation on Microsoft .NET Framework	Platform Dependent, Less Scalable, Less Adoptability

III. IMPLEMENTATION DETAILS

A. Computation Offloading Framework (TDM)

TDM is nothing but a daemon, similar to a process that execute in a backend. It has two modules: factor measurement and ternary decision making. Before startup of a system, static decision factors are measured. These factors are determined at static time so these are independent of a module and deterministic. As shown in Fig. 1, at a dynamic time, when a module is invoked, TDM first looks for the presence of the corresponding factor table of the application's module, which collects required parameters or device information to evaluate energy savings and execution time. If the factor table is not present, the module of mobile application executed locally on CPU, and a creation of corresponding factor table process completed.

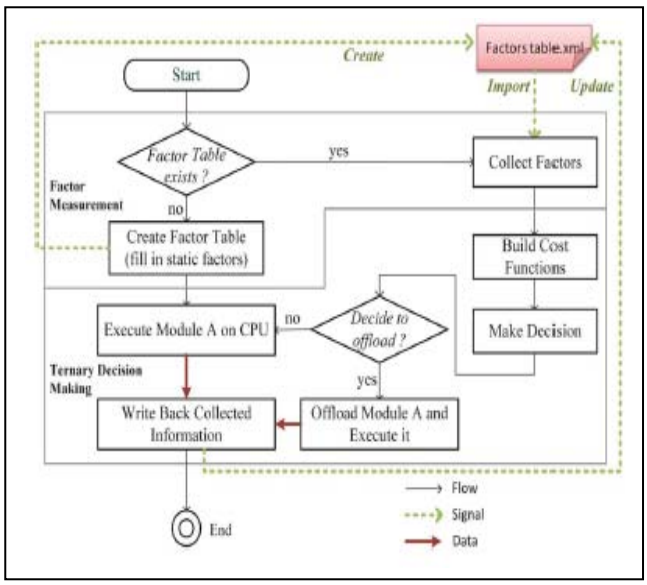

Fig. 1. Flow of a TDM [10].

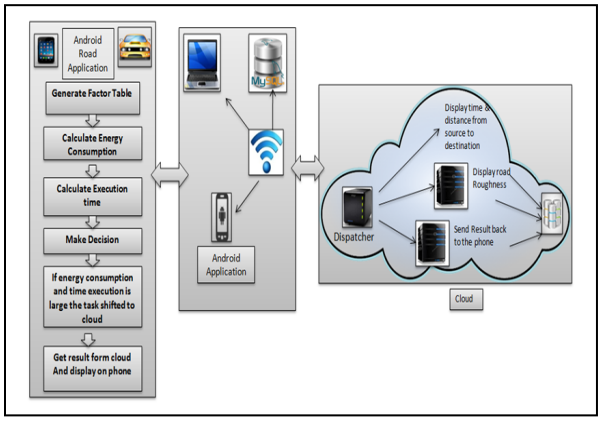

Fig. 2. System Architecture.

TDM insert static decision factors and module-dependent decision factors to the factor table, which are provided by the module to the factor table. After completion of a module, factor table records execution time for future use. But, if the corresponding table of factors is present, TDM obtains the decision factors from the decision table. TDM then request cloud to give speed of execution and calculates network transmission bandwidth. By passing all the decision factors to the cost functions, the decision maker takes the offloading decision. Offloading decision is of two types: one to execute locally and second to execute on cloud. Here, a module is a mobile application's function for finding road roughness. Workloads for frequently used functions are deterministic in nature. For a cloud execution, module will be offloaded [10].

Computation Offloading Decision Making Framework (CODMF), which intends to reduce execution time as well as conserve energy simultaneously. While running for the first time application create factor table for calculating the energy consumption and time consumption. Each device possesses a factor table. A table of factors collects required device information to evaluate battery usage and execution time of the corresponding application for finding roughness of a road. Execution of computation intensive or resource intensive task will be done at mobile side and at a cloud side, which results in a highly flexible framework for execution of android applications.

Optimizing energy consumption and response time are two divergent goals. Firstly, design a cost function which is customizable and helps end users to adjust the weight of energy consumption and response time. By developing a lightweight profiling method, evaluate enhancement in a performance and energy usage obtain from offloading. To make right decisions, important factors like network transmission bandwidth, Speed of mobile CPU, and memory bandwidth etc., plays a key role and need to consider when building cost function. To make offloading decision based on the user specified and customizable cost function, energy usage and response time evaluated. Computation Offloading Decision Maker Framework (CODMF), which is used to minimize energy usage and response time the same time. The CODMF can be explained with the following modules:

1) Creation of factor table: When runing an application for the first time create factor table for calculating the energy consumption and time consumption. Each device has its own factor table. A table of factors table collects required device information to evaluate execution time and energy usage of the corresponding application for finding road roughness. For the Measurement of decision factors like wireless bandwidth, component speed and execution time; Calculate the local CPU speed and at runtime sent request to the cloud to obtain cloud speed. Also, measure memory bandwidth at runtime by measuring the access time for a large amount of data saved in the memory.

2) Calculate Energy Consumption and Execution time(Ddecision making): Calculate energy consumption at mobile side and at cloud side. Also, execution time at moble side and at cloud side.

3) Task offloading: If the energy consumed on location is maximum the and also determine the time required for execution is large then task will be offloaded to the cloud.

4) Apply on Application (Finding Road Roughness): The energy consumption and time execution is applied on android application for finding road roughness.

In addition, MCOP (min-cost offloading paritioning) algorithm [11] to be implemented in a TDM framework to have more accuracy in energy and time saving estimation.

B. Offloading Partitioning Algorithm

The MCOP algorithm [11] aims at finding the optimal partitioning cost that minimizes response time, energy consumption or the weighted sum of energy and time. The MCOP algorithm includes two steps as follows:

1) Unoffloadable Vertices Merging: An unoffloadable module in our application (for example, sensor data collection for finding road roughness) is the one which is unable to be migrated outside of the mobile device and thus is located only in the unoffloadable partition. Apart from this, choose any module to be executed at mobile side (local execution). Then all modules that are not going to be migrated to the cloud are grouped or merged into one that is selected as the source vertex. 'Merging' means application's opeartions are coalesced into one, whose 'communication cost to the offloaded computation' is the sum of 'communication cost to the offloaded computation' of all merged application's operations. Let G represent the original graph after all the unoffloadable modules are merged.

2) Coarse Partitioning: This step of coarse partitioning is to coarsen G to the coarsest graph $G_{|V|}$. To merge two application's operations or calculations and reduce their count by one is refer to as "coarsen". Therefore, the algorithm has $|V| - 1$ phases. In each phase i (for $1 \leq i \leq |V| - 1$), the cut value i.e., the partitioning cost in a graph $G_i = (V_i , E_i)$ can be calculated. G_i+1 arises from G_i by merging "suitable modules", where $G_1 = G$. In an idividual phase i, the partitioning results are the minimum partitioning cost among all the costs and the corresponding group lists for mobile side (local execution) and cloud side

execution. In each phase i of the coarse partitioning, there are five steps:

a) Start with A={a}, where a is usually an unffloadable application's operation in G_i.

b) Iteratively add the module to A that is the most tightly connected to A.

c) Let s, t be the last two modules (in order) added to A.

d) The graph cut or cost of the phase i is ($V_i \setminus \{t\}$, {t}).

e) G_{i+1} arises from G_i by merging modules s and t.

C. Mathematical Model

Application first invoke a module with the size of N_{input} to process data saved in memory. Process this data locally or remotely. Processed data is of size N_{output} are stored in a memory again. CODMF involves two parts:

1) Factor Table Creation

While running an application for the first time, create factor table to calculate energy consumption and time consumption. Each device possess its respective table of a factors. A table of a factors collects device information which contains required specifications to evaluate energy usage and execution time of the associated application. Table of factors involves measurement of decision factors:

a) Wireless Bandwidth (B) = (data sent in KB)/ (time in seconds) = KB/s

b) Component Speed: μ_{cpu}, μ_{cld}, μ_{mem}

Calculate the speed of local CPU μ_{cpu} and at run time sent query to the cloud to obtain speed of a cloud μ_{cld}. Also, measure memory bandwidth μ_{mem} at run time. Time required to access a huge amount of data termed as memory bandwidth and estimated as,

μ_{mem} = (amount of data access in MB)/(time in sec(20s)) MB/s

c) Execution time: t_{comp}, t_{trans}

$$t_{comp} = T_{cpu} - t_{trans}$$

Where T_{cpu} is the overall execution time and t_{trans} is a transmission time.

2) Decision Making

Ternary decision making involves:

a) Energy Consumption and Execution Time

i) Execution time for application to execute on local CPU can be calculated as,

$$T_{cpu} = t_{trans} + t_{comp} \quad (1)$$

Where t_{trans} is a transmission time. t_{comp} is a time consumed by CPU for a code execution termed as "computation time".

$$t_{trans} = (N_{input} + N_{output})/\mu_{mem} \quad (2)$$

ii) Based on equation (1) Energy consumption for local CPU execution can be estimated as,

$$E_{cpu} = (P_{basic} + P_{cpu}) * T_{cpu}$$

To calculate T_{cld}, determine data transfer rate as,

$\sigma = ((N_{input} + N_{output})/MTU) *$ (size of DATA packet)

Where MTU is "maximum transmission unit".

ACK packets handled in the time of transmission are calculated by,

$\sigma^{ack} = ((N_{input} + N_{output})/MTU) *$ (size of ACK packet)

iii) Execution time for cloud execution can be calculated as,

$$T_{cld} = ((\sigma + \sigma^{ack})/\mu_{mem}) + ((\sigma + \sigma^{ack})/B) + ((t_{comp} * \mu_{cpu})/\mu_{cld}) \quad (3)$$

iv) Energy consumption for Cloud execution is estimated by,

$$E_{cld} = P_{basic} * T_{cld} + P_{nic} * ((\sigma + \sigma^{ack}/\mu_{mem}) + (\sigma + \sigma^{ack}/B)) \quad (4)$$

Where P_{nic} is the power usage of the network interface.

b) Make a Decision:

At runtime, measure t_{comp}, N_{input}, N_{output}, and B dynamically. By using these parameters, determine T_{cpu}, E_{cpu}, T_{cld}, and E_{cld}.

Define β_x as, $\beta_x = (T_{cpu} - T_x)/T_{cpu}$

Here, x is "cld" for Cloud and "cpu" for mobile CPU. β_{cld} is the percentage of degradation in execution time by offloading the module to the cloud. $\beta_{cpu} = 0$, if offloading of the module does not takes place.

To represents the energy degradation, Calculate γ_x as,

$$\gamma_x = (E_{cpu} - E_x)/E_{cpu}$$

Here, x is "cld" for Cloud and "cpu" for mobile CPU. To recognize the importance of battery usage and execution time, State a combine cost function as,

$$f(\alpha, x) = \alpha \cdot \beta_x + (1 - \alpha) \cdot x \quad (5)$$

where α is a user specified variable in the range [0,1]. If α=0, battery usage is the choice to determine offload target and if α=1, execution time is the choice to determine offload target. To determine an offload target of the application, obtain composite cost function $f(\alpha, x)$ of each possible offload target and select the one with a minimum value as the offload target. In other words, given the user defined α, we offload the module to target y, which is evaluated by,

$$y = \arg\min f(\alpha, x), \text{ for } x \in N. \text{ where N is the set of } \{cpu, cld\}$$

D. Experimental Setup

The system is built using Java framework on Windows platform. The Eclipse is used as a development tool along with Android SDK. Lenovo K3 Note with Android Ver. 4.2.2 used for the analysis. The system analysis is supported out on android application.

IV. Results and Discussion

In this work we have used application for finding road roughness and partition algorithm to analyze the performance of CODMF. Existing system does not use partitioning but proposed system make use of partition algorithm to achieve more accuracy in offload decision making.

Factor Table in Fig.3 clearly shows that how offloading decision is takes place. Cost comparison at mobile side and at cloud side help to get accurate offloading decision.

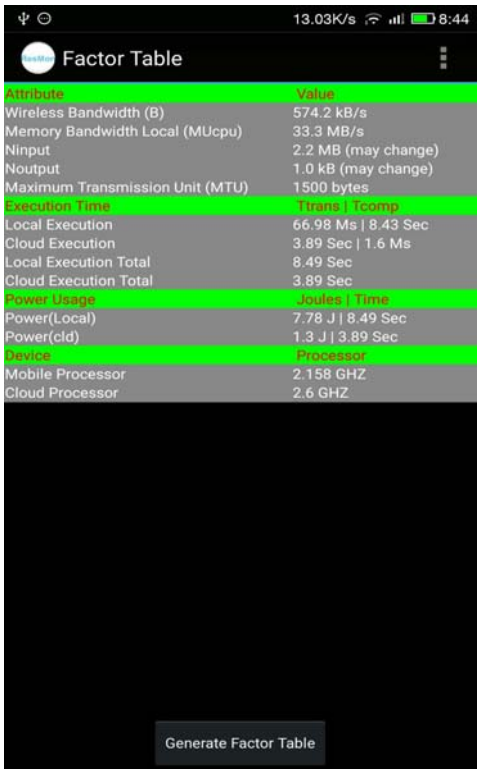

Fig. 3. Factor Table.

Following Fig. 4 represents comparison of execution time at mobile side and at cloud side and Fig. 5 represents comparison of energy consumption at mobile side and at cloud side.

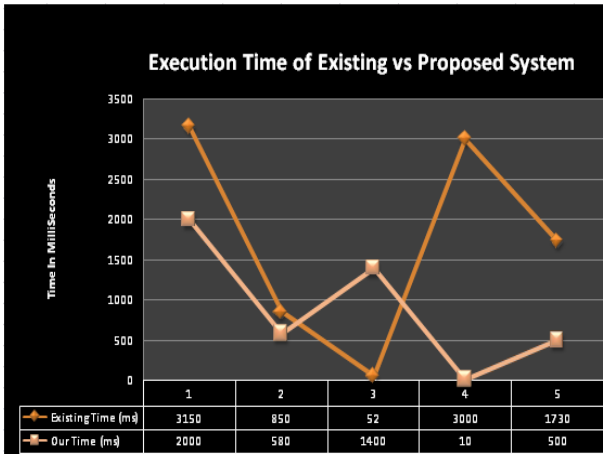

Fig. 4. Time Comparison.

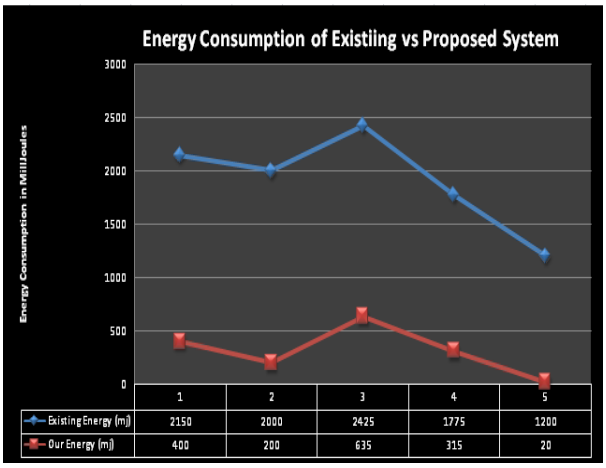

Fig. 5. Power Comparison.

V. Conclusion

To reduce response time and save energy for mobile devices at a time is a challenging task as unit of execution time and power consumption are different. System develops mechanism for profiling to evaluate the enhancement in a performance. Proposed computation offloading framework can attain significant improvements in battery savings and time. Also, reduce memory consumption at mobile side. Based on Partition algorithm and decision accuracy of offloading framework, resource intensive modules of application for finding road roughness can be offloaded to the cloud. By offloading modules to the cloud we attain maximum reduction in execution time as well as savings in battery usage.

References

[1] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In Proceedings of the 5th international conference on Mobile systems, applications and services, pages 165–178. ACM, 2007.

[2] B.-G. Chun and P. Maniatis. Dynamically partitioning applications between weak devices and clouds. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond. ACM, 2010.

[3] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely. Energy-delay tradeoffs in smartphone applications. In Proceedings of the 8th international conference on Mobile systems, applications, and services, pages 255–270. ACM, 2010.

[4] 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, pages 301–314. ACM, 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 INFOCOM, 2012 Proceedings IEEE, pages 945–953. IEEE, 2012.

[6] P. Shu, F. Liu, H. Jin, M. Chen, F.Wen, Y. Qu, and B. Li. "eTime: energy-efficient transmission between cloud and mobile devices" In INFOCOM, 2013 Proceedings IEEE, pages 195–199. IEEE, 2013.

[7] R. Beraldi, K. Massri, M. Abderrahmen, and H. Alnuweiri. "Towards automating mobile cloud computing offloading decisions: An experimental approach" In ICSNC 2013: The Eighth International Conference on Systems and Networks Communications, 2013.

[8] J. Niu, W. Song, and M. Atiquzzaman. Bandwidth-adaptive partitioning for distributed execution optimization of mobile

applications. Journal of Network and Computer Applications, 37:334–347, 2014.

[9] X. Lin, Y. Wang, Q. Xie, and M. Pedram. Energy and performance-aware task scheduling in a mobile cloud computing environment. In Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on, pages 192–199. IEEE, 2014.

[10] Ying-Dar Lin; Chu, E.T.-H.; Yuan-Cheng Lai; Ting-Jun Huang, "Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds," in *Systems Journal, IEEE* , vol.9, no.2, pp.393-405, June 2015.

[11] Wu, H., Seidenstücker, D., Sun, Y., Nieto, C.M., Knottenbelt, W. J., & Wolter, K. A Novel Offloading Partitioning Algorithm in Mobile Cloud Computing. Unpublished, 2015.

[12] Jaya Ashok Suradkar and R D Bharati. Article: Computation Offloading: Overview, Frameworks and Challenges. International Journal of Computer Applications 134(6):28-31, January 2016. Published by Foundation of Computer Science (FCS), NY, USA.

[13] 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 Int. Conf. on Mobile systems, applications, and services, pages 49–62. ACM, 2010.

.