IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 3, May 2014.
www.ijiset.com
ISSN 2348 - 7968

Performance Improvement of a Mobile Device by Selecting between Offloading and Local Computation

Darshan Thoria[1], Prof. Dhaval Nimavat[2]
[1] ME-CE Student, Atmiya Institute of Technology & Science,Rajkot, Gujarat-360005, India
[2] CE-Department, Atmiya Institute of Technology & Science,Rajkot, Gujarat-360005, India

Abstract
Energy is a primary constraint for mobile systems. Smartphones are started also being used for voice communication; instead, they are used for acquiring and watching videos online, online gaming, web surfing, and many other purposes. As a result, these systems will likely consume more power. And usage of network is the first parameter to lower the performance of device. Algorithm given contains taking a meaningful decision between offloading and Local Computation. And if offloading is needed, it also provides the cheapest way of it. So with the increasingly use of mobile cloud computing we can make a device performance better in Dynamic Environment.

Keywords: *Darshan Thoria Dhaval Nimavat, Performance, Improvement, Offloading.*

1. Introduction

Mobile systems have limited resources, such as battery life, network bandwidth, storage capacity, and processor performance. [1]

The consistent trend of the 21st century is the move to mobile devices as a primary computing platform for many users: laptops have replaced desktops, netbooks have replaced laptops, tablet PCs have replaced netbooks and increasingly, smartphones are replacing other devices.

Unfortunately, the desire for rich, powerful applications on mobile devices reduces the performance. One emerging approach to dealing with the resource limitations of mobile devices is to leverage computation offloading, where parts of an application's execution are performed on a remote server, with results communicated back to the local device. [2] And another way is to do Local Computation. In today's world, mobile devices are having rich set of hardware components. So it is not always cheaper to offload data in usage of modern mobile devices. It is necessary to reduce the energy consumption to improve the performance. The algorithm helps in selection of a cheaper option so that the performance can be improved.

2. Introduction to Mobile Data Offloading Technique

Mobile Data Offloading is a solution to augment these mobile systems' capabilities by migrating computation to more resourceful computers (i.e., servers). This is different from the traditional client-server architecture, where a thin client always migrates computation to a server. Computation offloading is also different from the migration model used in multiprocessor systems and grid computing, where a process may be migrated for load balancing.

3. Analysis

3.1 Study of the Current System

A significant amount of research has been performed on computation offloading: making it feasible. Researchers mostly focused on making offloading feasible. This was primarily due to limitations in wireless networks, such as low bandwidths. Their focus didn't moved to developing algorithms for making offloading decisions i.e., decide whether offloading would benefit mobile users.

The decisions are usually made by analyzing parameters including bandwidths, server speeds, available memory, server loads, and the amounts of data exchanged between servers and mobile systems. The solutions include partitioning programs and predicting parametric variations in application behavior and execution environment. Offloading requires access to resourceful computers for short durations through networks, wired or wireless.

3.2 Problem of Current System

First and foremost problem of current system is a lake of decision making for offloading itself. These are more problems exists in to the Mobile Cloud Computing, for e.g.

Inter-operability: Different types of resource constrained devices may interact and connect across different types of networks to one or many servers.

Mobility and Fault Tolerance: Offloading relies on wireless networks and servers; thus it is important to handle failures and to focus on reliable services.

Privacy and Security: Privacy is a concern because users' programs and data are sent to servers that are not under the users' control.

And several other problems, but one of them clearly effects on performance and that is Battery Performance.

Battery Performance: According to a new survey from ChangeWave, owners of Apple's new iPhone 3GS are quite happy with the device. Asked what they dislike most about the iPhone, 41 percent of respondents said the device's short battery life. [3]

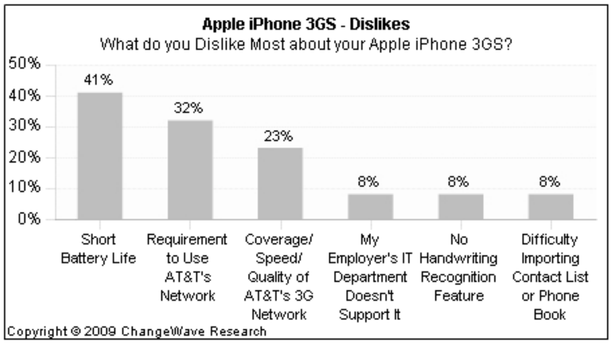

Fig. 1 Short Battery Life Problem.

3.3 Features of New System

The primary avenues for work concern taking a meaningful decision between local execution and offloading. First, if the computation is less and can be computed on local device with comparatively lower power consumption than offloading, take it.

Otherwise select that cheapest node found by an efficient algorithm which returns a cheapest partition cost from Interaction Graph. We intend to build a system that can generate code and perform partitioning in a manner consistent with the scheme described.

There will be a gateway that decides whether to even check for the offloading or not. Because in today's world, mobile devices are having richer components by using those a Local Computation may clearly be a cheaper option.

For platform dependency may be removed by using Cloud APIs. Cloud APIs are application programming interfaces (APIs) used to build applications in the cloud computing market. [4]

4. Proposed System

4.1 Introduction to System

A system, if it has the amount of user data is reasonably smaller that can be computed on local device in a cheaper way rather than offloading then it will be the selection of the system. But the data and computations are found cheaper to be offloaded then the selection will be of offloading.

Accomplishing this type of data-centric offloading requires that several key challenges be addressed:

1) Algorithm must be developed having ability to take a decision about offloading.

2) If an offloading is decided then partition a program between multiple possible executions sites in a multi-site searching offloading scenario, the partitioning algorithm must account for differences in capabilities between remote sites.

4.2 Proposed Algorithm

To develop an algorithm that can calculate following main parameters on which it can take a decision about offloading. The parameters are,

I: Number of instructions.

S: The speed of cloud to compute C instructions.

M: The speed of mobile to compute C instructions.

D: The data needed to transmit.

B: The bandwidth of the Wireless Internet.

Cc: Energy usage when the mobile device is doing computing.

Ci: Energy usage when the mobile device is idle.

Ct: Energy usage when the mobile device is transmission the data.

Calculate Local computation Cost T1.

$$T1= (Cc*I/M) \quad (1)$$

Calculate Idle Time of device T2 when cloud is computing.

$$T2= (Ci*I/S) \quad (2)$$

Calculate time T3 for transmission of data.

$$T3= (Ct*D/B) \quad (3)$$

Calculate an equation to decide what will be cheaper, local execution or offloading.

$$T = Cc*I/M - (Ci*I/S + Ct*D/B) \quad (4)$$

If T is a positive digit, then offloading time is lower and we can choose offloading computation.

If T is a negative digit, then local execution is better, so don't offload.

If T is zero (in case), then also select local execution because of the better stability.

4.3 Algorithm

Input: I, M, D, B, u is the given values, v is the set of vertices

Output: Offload or not.

The best node to offload. (In case of offloading)

Step 1: If B (in KBpS) / I >= 1*, then go to step 16.

Step 2: Fetch all the vertices existing on the site.

Step 3: Fetch the vertex weight of that site.

Step 4: Select u as a selected vertex.

Adj[n] = Get vertices AdjTo(u)

Step 5: Make a group of vertex Array of allocation site of Partition P1.
Step 6: Repeat step 1 to 5 for to make all allocation site.
* Gateway, subject to change as per situation.

Step 7: Visit all the vertex in to allocation.
Step 8: When one method of S1 site invoke the allocation site S2 then Make Edge between them.
Step 9: If alias of S exist then make it as source vertex.
Step 10: Check all the methods accessed from that allocation site S1.
Step 11: Edge is formed between that two vertexes of allocation site.
Step 12: Repeat step 1 to 10.
Step 13: Evaluate the Graph find maximum vertex weight and assign it to S.
Step 14: Calculate T = T1 – (T2 + T3).
Step 15: If T is negative, go to step 15, otherwise 16.
Step 16: Perform Local Execution, END.
Step 17: Perform offloading, END.

5. Result and Comparative Analysis

5.1 Result

As a result of decision, as per different scenarios, we are getting different results by the algorithm. Major deciding factors seen affected to this algorithm are,

- Computation Speed of Mobile Device. (M)
- Bandwidth of wireless internet. (B)

Here is a comparative table showing different scenarios. Rest of the parameters is kept constant for comparison.
Here in Table 6.1, we can see that the variation in bandwidth (B) and Computation Speed of Mobile device (M) make changes in decision of Local Computation or Offloading.

Table 1: Runtime Decision taken with variable B and M

M (GHz)	S (GHz)	B (KBpS)	T	Selection
1	4	5	-1650	Local Computation
		10	-450	Local Computation
		50	510	Offload
		500	726	Offload
2	4	5	-2100	Local Computation
		10	-900	Local Computation
		50	60	Offload
		500	276	Offload
3	4	5	-2250	Local Computation
		10	-1050	Local Computation
		50	-90	Local Computation
		500	126	Offload

Each time we are having four variations of Bandwidth (B), a static Computation speed of Cloud, and variable Computation speed of Mobile Device. By changing these two we get a different T and according to the value of T, the selection of method is done.
Here is the graph shown having variable B and M and depends on T, the selection is done of Local Computation or Offload.

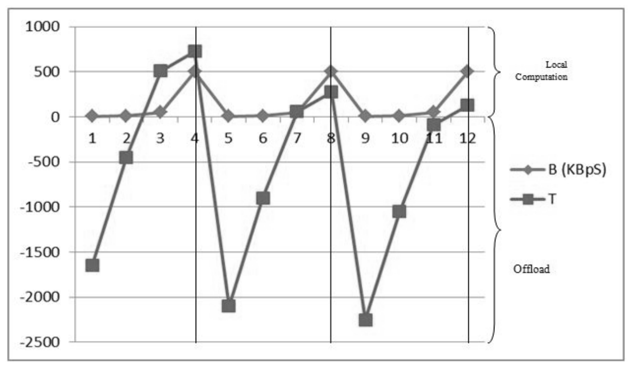

Fig. 2 Graph showing decision depends on T.

5.2 Comparative Analysis

Here is the comparison table of response time of proposed algorithm to existing techniques. As a result we are getting quick response.

Table 2: Response time comparison

Nodes	HELVM	Labelling Technique	Proposed
10	1496	799	764
15	3331	1639	1498
30	14048	5570	5375
50	38577	14596	14481

The graph shows the pictorial view of the above table of comparison.

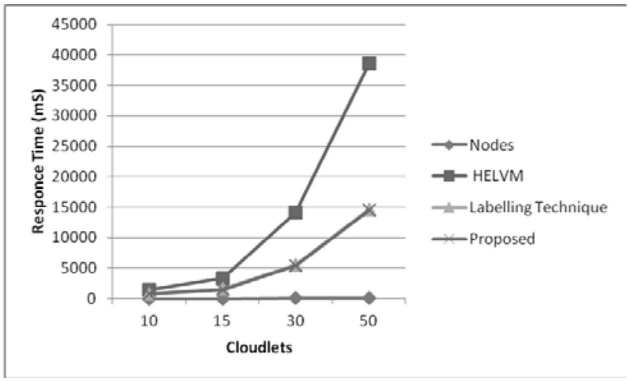

Fig. 3 Response time comparison.

Here showing one case in which we are comparing following two existing techniques with proposed algorithm.

- Never Offload (Local Computation)
- Always Offload.

In that for getting better comparative results, we are keeping bandwidth as a constant parameter. Here is the table given below.

Table 3: Fixed Bandwidth Performance Comparison

B	I	Local Computation	Offloading	Proposed Algorithm Selection
50	3000	270	330	270
50	3500	315	345	315
50	4500	405	375	375
50	5000	450	390	390
Total		**1440**	**1440**	**1350**

Here is the graph shown of all three techniques. We can see that the proposed algorithm selected the cheapest option all the time.

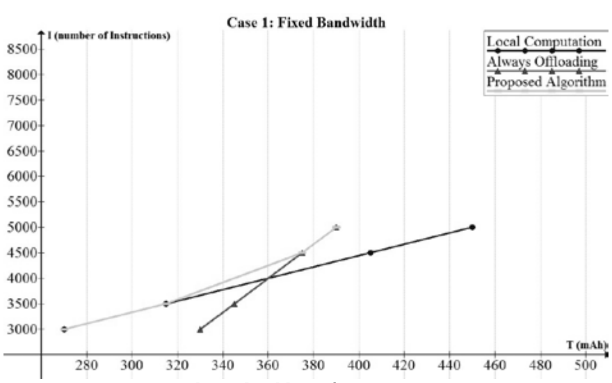

Fig.4 Fixed Bandwidth Performance Comparison

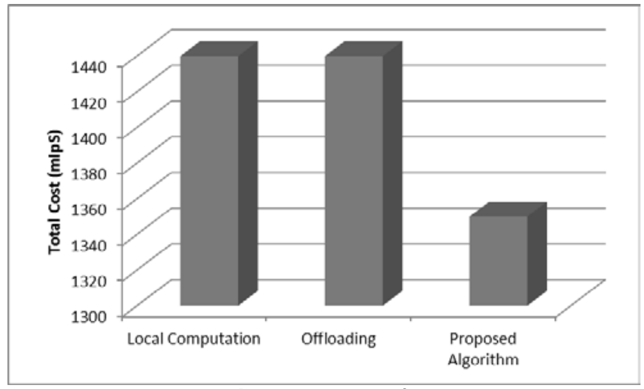

Fig. 5 Total Cost Comparison for case 1

Here, total cost comparison of proposed algorithm is shown with both the methods. We can see that, it is beneficial to take decision between Local Computation and Offloading.

6. Conclusions

If we calculate the total cost of proposed algorithm to both existing techniques, we are getting 6 to 7% of improvement in device performance in above "regular situation" example in which local computation is cheaper and in another two, offloading is cheaper.
In worst case scenario, 10 to 12% improvement in performance may be seen.

References

[1] Karthik Kumar, Jibang Liu, Yung Hsiang Lu, Bharat Bhargava, "A Survey of Computation Offloading for Mobile Systems", Springer, April-2012.
[2] Ahumao Ou, Kun Yang, Antonio Liotta, "An adaptive Multi - Constraint Partitioning Algorithm for Offloading in Pervasive Systems", IEEE-2006.
[3] Kanad Sinha, Milind Kulkarni, "Techniques for fine-grained, multi-site Computation Offloading", IEEE-2011.
[4] Juntunen Antero, Kemppainen Matti, Luukkainen Sakari, "Mobile Computation Offloading-Factors Affecting Technology Evolution", ICMB-2012.
[5] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, Xinwen Zhang, "Think Air: Dynamic Resource Allocation and Parallel Execution in cloud for Mobile Code Offloading", IEEE-2012.
[6] Huber Flores, Satish Srirama, "Adaptive Code Offloading and Resource-intensive Task Delegation for Mobile Cloud Applications".
[7] Antonis Hadjiantonis, Georgios Ellinas, "Converged Network and Device Management for Data Offloading", IEEE-2012.
[8] Eemil Lagerspetz, Sasu Tarkoma, "Mobile Search and the Cloud: The Benefits of Offloading", IEEE-2011.

[9] Antti P. Miettinen, Jukka K. Nurminen, "Energy Efficiency Of Mobile Clients In Cloud Computing", 2nd USENIX conference, CA, USA 2010.
[10] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya, "CloudSim: A Toolkit for the Modeling and Simulationof Cloud Resource Management and Application Provisioning Techniques", http://dx.doi.org/10.1002/spe.995, January 2011.
[11]Fotis Gonidis, Iraklis Paraskakis, Dimitrios Kourtesis, "Addressing the Challenge of Application Portability in Cloud Platforms", Open Excellence Workshop – Rennes, 6-7 September 2012.

Darshan Thoria is a B.E. (C.E.) from AITS, Saurashtra University, Rajkot-Gujarat, India, in 2009; perusing M.E. (C.E.) from AITS, Gujarat Technological University, Gujarat-India.

Dhaval Nimavat is having total 7 years of experience as an Asst. Prof., currently he is currently working as same at Atmiya Institute of Technology and Science, Rajkot-Gujarat, India.