ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

TASK OFFLOADING TO THE CLOUD BY USING CUCKOO MODEL FOR MINIMIZING ENERGY COST

ANJALI V, Dr. S. SRINIVASAN

Department of Computer Science and Engineering, Anna University Regional campus Madurai- 625 019 INDIA (e-mail: anjali.malathi@gmail.com)

Abstract: The current needs for handy devices to complete tasks, tending these devices to accommodate new changes in it. The increased usage of these devices caused them to face a large amount of resource, memory and processing speed scarcity. Of all other constraints, energy is the major problem for smartphones to carry out a task. The concept of offloading gets into play for mobile devices i.e., the task or the computation which needs to be performed involving more service in the android systems will be shifted to resourceful server (for ex cloud) and getting back the results done from the cloud. This practice has been introduced under a new field called Mobile Cloud Computing. There are different types of offloading in MCC depending on the application which involved. The decision of whether to offload a computation or not will depend on the task accounting to the energy spent by the device while working with the application versus the amount of energy spent by the same device for uploading the task to the cloud and getting the result back from the cloud. The types of applications will also decide upon offloading. As this concept depicts energy is the major constraint for whether to offload a task or not, there is a model called CUCKOO framework which acts as an interface between the cloud and the android environment to support for the task offloading to the cloud. Thus this framework bridges the gap between the smartphones as well as the cloud environment so that that computation intensive task can be performed with less amount of energy consumed. In this work two applications are used to detect the amount of energy consumed in the cloud as well as the smartphones namely eyedentify and Photoshoot.

KeyWords: Mobile Cloud Computing, Cuckoo Framework, Offloading, eyeDentify and Photoshoot.

1. INTRODUCTION

In recent years, mobile devices such as smart phones and tablets have been upgraded into more powerful terminals with faster CPU, substantial memory, and multiple sensors. Each of these devices is able to serve the users' request depending on individual applications. However, the battery lifetime is still a major concern of the modern mobile devices. From the users' perspective, they need better performance of their mobile devices, which reflects on longer battery life and shorter processing time of any kind of services. To overcome this obstacle, mobile cloud computing is introduced. Mobile Cloud Computing is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers. Each of the mobile devices is able to serve the users' request depending on individual applications. However, the battery lifetime is still a major concern of the modern mobile devices. To overcome this obstacle, mobile cloud computing is introduced. Mobile Cloud Computing is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers. The task offloading will offer a better service when it is coupled with a framework called CUCKOO model.

Task offloading is a critical technique because in some cases it increases the energy consumption of smartphones. This technique can be four variants depending on the task and data involved in the particular application involved. In the first case, The input data is available locally on the smartphone and task execution occurs on the smartphone as well. This is the normal case where no offloading occurs. The second case is where the task execution happens on the cloud but the task data exists locally on the smartphone. In this scenario, the smartphone has to upload the task data to the cloud and then download the task results. The third scenario is where the task execution is performed locally on the smartphone, but the task data exists on the cloud. In this scenario, the smartphone needs to download the task data and perform the task execution locally. In this scenario, the input data is available on

All Rights Reserved © 2016 IJARBEST

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

the cloud and task execution occurs in the cloud as well. Therefore, the smartphone just needs to download the task results.

The task offloading will offer a better service when it is coupled with a framework called CUCKOO model. The Cuckoo framework, which simplifies the development of smartphone applications that benefit from computation offloading and provides a dynamic runtime system, that can, at runtime, decide whether a part of an application will be executed locally or remotely.

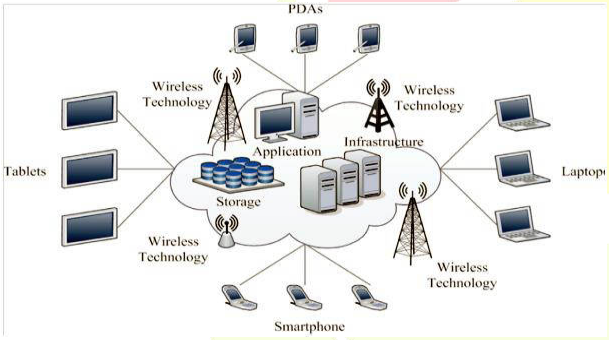

Fig 1. Overview of MCC

1.2 OBJECTIVE

Providing uninterrupted service to the smartphone is evitable because of its energy consumption and low resource. To enhance the handheld device to next dimension a hybrid platform called Mobile Cloud Computing is used to minimise energy consumption is introduced. The cuckoo framework is used to analyze the energy cost consumption for task offloading to the cloud. Dynamic offloading decisions based on cuckoo model are worked out. The energy consideration for each of the cases is calculated.

$$E_{Cuckoo} < E_{Local\ Execution}$$

ECuckoo- Energy consumed by the smartphone using Cuckoo framework (With Offloading)

ELocal Execution-Energy consumed by the smartphone (Without Offloading)

1.3 SCOPE OF THE PROJECT

The aim of the project is to reduce the amount of energy consumed by the individual tasks in the smartphones, So that the smartphones battery lifetime can be enhanced. According to Moore's law, the number of transistors on an integrated circuit doubles every two years. In contrast, battery capacity increases only by 5% every year. This fact implies that the gap between energy demand and supply grows by 4% annually. This concept offers a better conjunction between the battery life and the task handling. For a task performed locally, it involves the energy consumption based on the application. In my consideration,

2. RELATED WORK

The objective of the context sensitive offloading scheme is to derive an optimal offloading decision under the context of the mobile device and cloud resources to provide better performance and less battery consumption. The proposed framework adopts client-server communication model, in which the cloud resources (e.g. mobile device cloud, public cloud) are servers and the mobile device is the client to access the services on servers. On the client side, the framework consists of three components, namely a context monitor, a communication manager and a decision engine. On the server side, it includes a server side communication manager, a program profiler and a task manager. The cost model consists of three parts, namely the task execution time denoted by D, wireless channel energy consumption denoted by E and monetary cost denoted by M when related. Then the total cost of executing task ti is as follows:

$$P(ti) = \alpha 1 * D(ti) + \alpha 2 * \rho d * E(ti)$$

In the context of cloud computing, in energy trade-off analysis the critical aspect for mobile clients is the trade-off between energy consumed by computation and the energy consumed by communication. We need to consider the energy cost of performing the computation locally (Elocal) versus the cost of transferring the computation input and output data (Ecloud). If D is the amount of data to be transferred in bytes and C is the computational requirement for the workload in CPU cycles then

$$Ecloud = D/Deff$$

$$Elocal = C/Ceff$$

where Deff and Ceff are device specific data transfer and computing efficiencies. The Deff parameter is a measure for the amount of data that can be transferred with given energy (in bytes per joule) whereas the Ceff parameter is a measure for the amount of computation that can be performed with given energy. With these we can derive the relationship between computing and communication for offloading to be beneficial

The basic idea of COSMOS is to achieve good offloading performance at low monetary cost by sharing

All Rights Reserved © 2016 IJARBEST

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

cloud resources among mobile devices. Specifically, in this paper our goal is to minimize the usage cost of cloud resources under the constraint that the speedup of using COSMOS against local execution is larger than 1−δ of the maximal speedup that it can achieve using the same cloud service, where δ ∈ (0, 1). It consists of three components: a COSMOS Master running on a VM instance that manages cloud resources and exchanges information with mobile devices; a set of active COSMOS Servers each of which runs on a VM instance and executes offloaded tasks; and a COSMOS Client on each mobile device that monitors application execution and network connectivity and makes offloading decisions.

Mobile cloud computing (MCC) enables the development of computational intensive mobile applications by leveraging the application processing services of computational clouds. Contemporary distributed application processing frame works use run time partitioning of elastic applications in which additional computing resources are occurred in runtime application profiling and partitioning. Distributed application processing is an important software level technique for enabling computationally intensive mobile applications on SMDs. A number of augmentation algorithms have been proposed for alleviating the resources limitations of SMDs — energy augmentation, memory augmentation (AbebeandRyan,2012; Guetal.,2003), and application processing augmentation. The current APAs for elastic applications use a number of strategies for separating the intensive components of the mobile application

In the following, Majid Altamimi et al modelled the energy usage in two distinct cases, namely, file upload and file download. For simplicity, he assumed that the mobile device transceiver uses only two power levels, namely, PRX when it is idle, in backoff mode, or receiving and PTX when it is transmitting. For File Download Case, the mobile device is mostly receiving. Here, we address first the general situation where there is no limitation on the file download rate from the cloud. Next, we address the situation where the cloud restricts the file download rate. For every MAC frame to be received, the mobile device has to send a CTS and an ACK frame. The mobile device has to send a TCP ACK for every received TCP segment. During downloading a file, a smartphone will be receiving a data frame for a time T +3SIFS+TPHY +TRTS and it has to wait for the AP backoff time ɘ/t.

3. EXISTING SYSTEM

Existing system consists of two major parts, smartphones (i.e., user equipment, UE) and Cloud Computing (CC), both linked to the Internet. The smartphones are connected to the Internet through a WLAN access point or a cellular data network base station. These smartphones provide all of mobile computing functionalities to the end users via different applications. On the other hand, the CC part consists of cloud data center and cloud provider, which are accessible through the Internet. The cloud provides the end users (e.g., smartphone users) with all of the CC functionalities that are needed for mobile computing. In the offloading technique, smartphones access the cloud via the Internet. Therefore, offloading is considered as a Network Related Application (NRA). At the beginning of studying NRA, network interfaces (i.e., 3G/4G and WLAN) should be considered because each of these interfaces has its own characteristics, such as supported data rate. As a result, each network interface consumes unequal amount of energy. In addition, the Internet protocols, namely, the Hypertext Transfer Protocol (HTTP) and the File Transfer Protocol (FTP) need to be taken into account. The network interfaces and protocols are the major factors that affect the energy costs of task offloading. They are taken into account for energy cost modelling.

3.1 THE EXPERIMENT

They experimentally evaluate the energy cost on smartphones when the offloading technique is used over different network interfaces and Internet protocols. They conducted experiments in four broad experimental scenarios related to the location of the task data. The first scenario corresponds to S1, where there is a local task execution and the task data exists on the smartphone. The second scenario corresponds to S2, where uploading the task data, doing the task computation (encoding) by the cloud and downloading the task result is presented by the "Upload + CC encoding + Download". The third scenario corresponds to S3, where there is a local task execution and the task data is downloaded from the cloud, as shown by the "Download + Local encoding". The fourth scenario corresponds to S4, where the task data exists in the cloud and the task executed on the cloud, and the task result is simply downloaded, as presented by the "CC encoding + Download". For uploading and downloading files to and from the cloud, we consider the energy implications of: (i) using the HTTP and FTP protocols at the application level; and (ii) using the 3G and WLAN communications at the wireless interface level.

4. PROPOSED SYSTEM

All Rights Reserved © 2016 IJARBEST

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

The proposed system is going to work in a Mobile Cloud Computing environment as briefed out earlier. The cloud environment is able to serve the variety of requests from the users based on the needs of them. Likewise the mobile device is able to serve the users by installing the applications specific to the needs of the users.

4.1 THE CLOUD MODELLING

Even though the cloud environment is able to support all kinds of users. But we need little consideration before taking the smartphones to the cloud. The general problems of the smartphones include energy, speed and the memory. For all these considerations the cloud environment is brought into these devices to minimize the power consumed. The cloud computing service model involves the provision, by a service provider, of large pools of high performance computing resources and high-capacity storage devices that are shared among end users as required. There are many cloud service models, but generally, end users subscribing to the service have their data hosted by the service, and have computing resources allocated on demand from the pool. The service provider's offering may also extend to the software applications required by the end user. To be successful, the cloud service model also requires a high-speed network to provide connection between the end user and the service provider's infrastructure.

The deployment of a delegated application on the virtual machine of the cloud server node is a challenging aspect of runtime computational offloading. Current COF focus on partitioning an elastic mobile application dynamically and offloading the intensive partitions at runtime. A critical feature of current COFs is that the delegated application needs to be reconfigured on the virtual device instance of on the cloud server node. Therefore, the execution of the offloaded applications to the cloud server nodes requires the deployment of virtual phone instance(s) on the virtual machine of the cloud data center. Because the hardware architecture and operating system platform of the mobile devices are different, the operating system platforms implement platform-specific application frameworks.

The communication links can be WiFi, 3G/4G. In our proposed work, an interface is used to connect between the smartphone and the cloud environment. So the energy consumed in this case is minimized. This interface is deployed in the ADSL layer of the android devices so that the application is coupled with this interface to make computations with regards to energy consumption. Thus the amount of energy saved is

$$Pc=\frac{C}{M}\text{-}Pi\times\frac{C}{S}\text{-}Ptr\times\frac{D}{P}$$

S : the speed of cloud to compute C instructions

M : the speed of mobile to compute C instructions

D : the data need to transmit

Pc : the energy cost per second when the mobile phone is doing computing

Pi : the energy cost per second when the mobile phone is idle.

Ptr: the energy cost per second when the mobile is transmission the data.

4.2 CUCKOO FRAMEWORK

Cuckoo Framework focus on minimizing effort to enhance the performance and reduce the battery usage of applications with heavy weight computation. offering a very simple programming model that is prepared for connectivity drops, supports local and remote execution and bundles all code in a single package. Integrating with existing development tools that are familiar to developers. Automating large parts of the development process. Offering a simple way for the application user to collect remote resources, including laptops, home servers and other cloud resources. This model used the existing activity/service model in Android that makes a separation between compute intensive parts (services) and interactive parts of the application (activities), through an interface defined by the developer in an interface definition language (AIDL). Otherwise an interface can easily be extracted from the code. This interface will be implemented as a local service that has, when used, a proxy object at the activity.

The Cuckoo framework provides two Eclipse builders and an Ant build file that can be inserted into an Android project's build configuration in Eclipse. The first Cuckoo builder is called the Cuckoo Service Rewriter and has to be invoked after the Android Pre Compiler, but before the Java Builder. The Cuckoo Service Rewriter will rewrite the generated Stub for each AIDL interface, so that at runtime Cuckoo can decide whether a method will be invoked locally or remote. The second Cuckoo builder is called the Cuckoo Remote Service Deriver and derives a dummy remote implementation from the available AIDL interface. This remote interface has to be implemented by the programmer. Next to generating the dummy remote implementation, the Cuckoo Remote Service Deriver also generates an Ant build file, which will be used to

All Rights Reserved © 2016 IJARBEST

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

build a Java Archive File (jar) that contains the remote implementation, which is installable on cloud resources. The Cuckoo Remote Service Deriver and the resulting Ant file have to be invoked after the Java Builder, but before the Package Builder, so that the jar will be part of the Android Package file that results from the build process.

4.3 INTELLIGENT OFFLOADING

A part of the Cuckoo framework is a Resource Manager application that runs on the smartphone. In order to make a remote resource known to a phone, the remote resource has to register its address to this Resource Manager using a side channel. If a resource has a display, starting a server will result in showing a two dimensional barcode – a QR code on the resources' screen. This QR code contains the address of the server. Smartphones are typically equiped with a camera and can scan this QR code using a special resource manager application. If the resource does not have a visual output, a resource description file can be copied from the resource to the phone to register the resource. When the resource is known to the Resource Manager application, it can be used repeatedly for any application that uses the Cuckoo computation offloading framework.

4.3.1 eyeDentify

Our first example application is eyeDentify, a multimedia content analysis application that performs object recognition of images captured by the camera of a smartphone. The idea of the application is similar to the Google Goggles application which can recognize contact info, places, logos, landmarks, artworks, and books. By offloading the computation needed for this algorithm, we have shown that we can speed up the computation with a factor of 60, reduce the battery consumption with a factor of 40 and increase the quality of the recognition.

4.3.2 PhotoShoot

The second example that we will consider is a distributed augmented reality game, called PhotoShoot, with which we participated in the second worldwide Android Developers Challenge and finished at the 6th place in the category 'Games: Arcade & Action'. This innovative game is a twoplayer duel that takes place in the real world. Players have 60 seconds and 6 virtual bullets to shoot at each other with the camera on their smartphone (see Figure 4). Face detection will determine whether a shot is a hit or not. The first player that hits the other player will win the duel. The major compute intensive operation in this game is the face detection. The Android framework comes with a face detection algorithm, so it is possible to create a local implementation to detect faces in an image. Without offloading, the slower the processor of the smartphone, the longer it takes for the shot to be analyzed, which gives the user of a slow smartphone a significant disadvantage. Offloading can, however, be used to make the game fair again.

5. SYSTEM ARCHITECTURE

The System architecture describes how the smartphone is connected to the internet and the cloud. While if the system is connected to the internet it will not be offloading the task rather it extracts the information needed to perform computation.

5.1 OFFLOADING SCENARIO (eyeDentify)

This scenario deals with the computation of energy consumed by the application during its working over the particular task. These two tasks are computation intensive since they involve image processing. At the earliest results shows off little difference over things but improves rapidly when there are huge records that need to be processed. So the energy consumed in case of no offloading scenario is that the task to be executed locally on the smartphone itself. If Ttask is the time taken for executing the compute intensive algorithm, then time taken by the client for no-offloading scenario is:

Tclient = Ttask

The total energy consumption Eno_offload is given by:

Eno_offload = Eclient = Pclient * T client

5.2 OFFLOADING SCENARIO (Photoshoot)

For offloading scenario, the task is executed on remote server. Hence, in this case Tserver = Ttask and total time returned at the client is given by.

Tclient = Ttask + Tcomm

Where, Tcomm is communication time for sending the input file to server and getting the result back from server. In this case, the offloading was done using Wi-Fi as well as 3G interface. For offloading, total energy consumption Eoffload_wifi and Eoffload_3G respectively are given by:

Eoffload_wifi = Eclient + Einternet_wifi + Edatacenter

Eoffload_3G = Eclient + Ebase_station + Einternet_3G + Edatacenter

All Rights Reserved © 2016 IJARBEST

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

Here, Eclient = Energy consumed by the client device,

Edatacenter = Energy consumed at the datacenter.

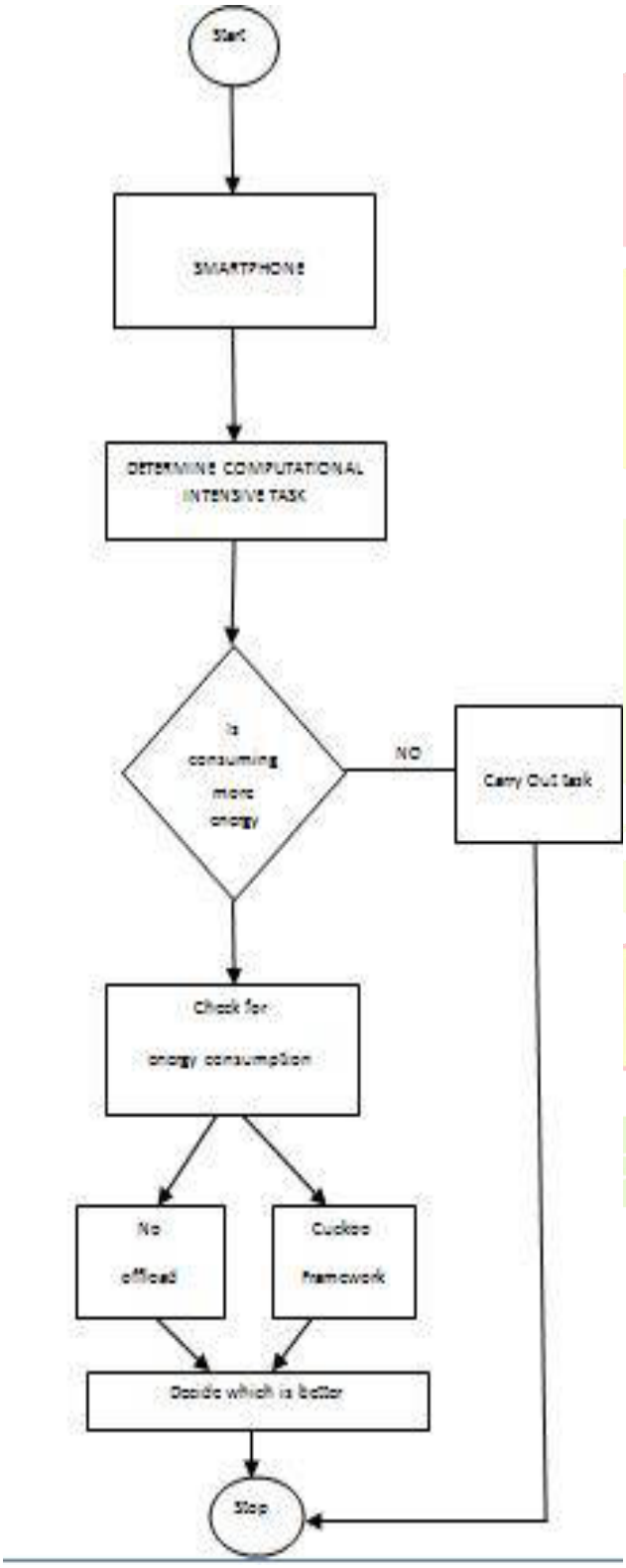

Fig 2. Task offloading using Cuckoo framework

Einternet = Energy consumed by the internet infrastructure

Energy consumption at the datacenter is given by:

Edatacenter = Eserver + Eoverhead

Where, Eoverhead = Energy consumed by HVAC, power supply and other overheads at datacenter.

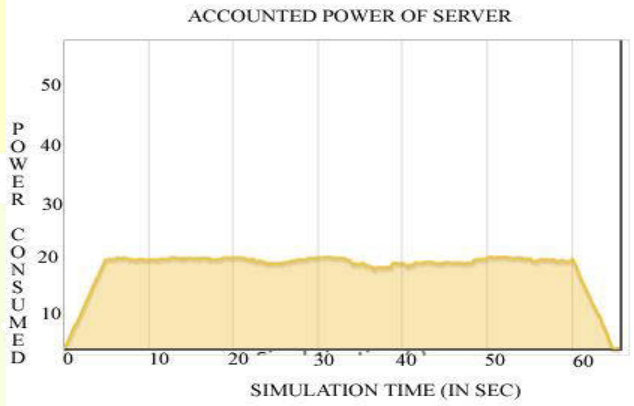

Fig 3. Amount of Time Spent By Offloaded Task Versus Power Consumed By application 1.

6. IMPLEMENTATION

This constitutes energy computation for both the mobile as well as the cloud environment. This can be done by installing the cloud environment using green cloud and mobile environment can be brought into the device using android SDK and eclipse with built- in java configurations. The green cloud installs the cloud environment along with its data center. The energy consumption for the devices which are connected can also be listed. So that the task installed in the cloud will be represented the graph for parameters like performance, response time, memory access, storage access, failure rate etc.,. are drawn. While installing the mobile environment power tutor application also needs to be installed, which is helpful for Calculating energy consumption of each of the applications.

The eclipse installing involves java RT environment to be installed onto the device. But the green cloud virtual disk is being coupled with the ns2, eclipse and c, c++ coding. Power tutor represents with all the parameters in the form of graphs. With all these parameters of energy responses of all the applications

All Rights Reserved © 2016 IJARBEST

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

having graphs of all the cases are depicted. And from the results the application is decided upon whether to offload or not. In this case,

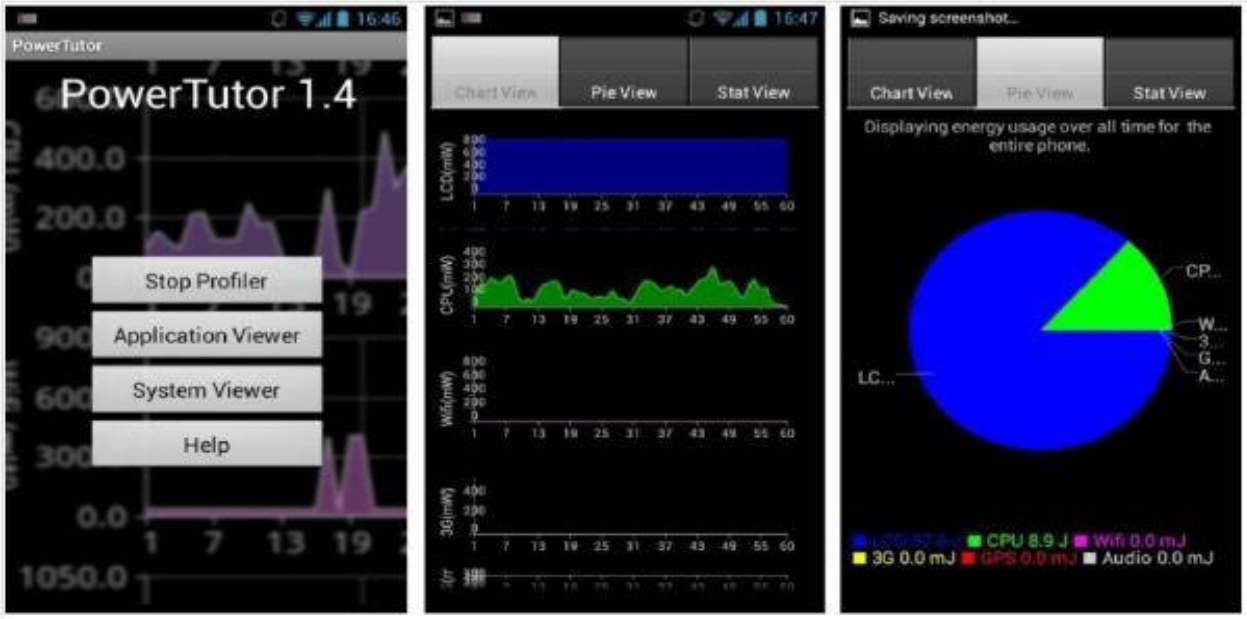

Fig 4. Power tutor Application

both the applications needs to be offloaded since they are involving image processing.

7. CONCLUSION

Task offloading is one of the emerging topics in Mobile Cloud Computing. Task offloading serves as better job of taking the task to be done on to the cloud by creating communication link between the cloud and the mobile device. The communication links also plays a important role in task offloading. WiFi serves better offloading than 3G/4G. Because they don't deserve a physical layer communication link as wifi. Likewise offloading gives better yield when there are number of tasks to be done. Cuckoo integrates with the popular open source Android framework and the Eclipse development tool. It provides a simple programming model, familiar to developers, that allows for a single interface with a local and a remote implementation. Cuckoo will decide at runtime where the computation will take place. Furthermore, the Cuckoo framework comes with a generic remote server, which can host the remote implementations of compute intensive services. A smartphone application to collect the addresses of the remote servers is also included. In this paper I evaluated the Cuckoo framework with two real world smartphone applications, an object recognition application and a distributed augmented reality smartphone game and showed that little work was required to enable computation offloading for these applications using the Cuckoo framework.

8. FUTURE ENHANCEMENT

As of wireless network is concerned security breaches may occur. (1) Ensuring security for the communication link can also be taken as a next step. (2) Offloading from smartphone involves certain security violation that can also be analysed and rectified. (3) Sometimes incompatibility in both the device and cloud system may result in different results, it should be checked out and a fixed framework can also be devised. (4) 5G communication link can also be tested. (5) This offloading can also be extended as data, application partitioning etc., in order to perform the variants of offloading. This future enhancement may allow the device to have a better scope in the Mobile Cloud Computing environment. The energy as well as memory and computation cost can also be reduced.

9. REFERENCES

[1]. A. P. Miettinen and J. K. Nurminen, "Energy Efficiency of Mobile Clients in Cloud Computing," in Proc. of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10), 2010, p. 4.

[2] J. Paradiso and T. Starner, "Energy Scavenging for Mobile and Wireless Electronics," Pervasive Computing, IEEE, vol. 4, no. 1, pp. 18 – 27, January-March 2005.

[3] S. Robinson, "Cellphone Energy Gap: Desperately Seeking Solutions," Strategy Analytics, Tech. Rep., Mar. 30 2009.

All Rights Reserved © 2016 IJARBEST

[4] A. Kansal and F. Zhao, "Fine-Grained Energy Profiling for Power- Aware Application Design," SIGMETRICS Perform. Eval. Rev., vol. 36, pp. 26–31, Aug. 2008.

[5] N. Vallina-Rodriguez, P. Hui, J. Crowcroft, and A. Rice, "Exhausting Battery Statistics: Understanding the energy demands on mobile

handsets," in Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile hand- helds, ser. MobiHeld '10. ACM, 2010, pp. 9–14.

[6] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," in Proc. IEEE 73rd VehicularTechnology Conf., 2011, pp. 1–6.

[7] K. Naik, "A Survey of Software Based Energy Saving Methodologies for Handheld Wireless Communication Devices," Dept. of ECE, University of Waterloo, Waterloo, ON, Canada, Tech. Rep. 2010-13, 2010.

[8] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When Mobile Terminals Meet the Cloud: Computation Offloading as the Bridge," Network, IEEE, vol. 27, no. 5, pp. 28–33, 2013.

[9] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a Unified Elastic Computing Platform for Smartphones with Cloud Support," Net- work, IEEE, vol. 27, no. 5, pp. 34–40, 2013.

[10].ADC2.http://code.google.com/android/adc

[11]. G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[12]ADT Eclipse plugin. http://developer.android.com/sdk/eclipseadt.html.

[13] Android. http://developer.android.com/.

[14] Android Market. http://www.android.com/market/.

[15] Apache Ant. http://ant.apache.org/.

All Rights Reserved © 2016 IJARBEST