International Journal of Computer Applications (0975 – 8887)
Volume 91 – No.14, April 2014

Framework for Offloading Android Applications using Cloud

Harsh Bandhu Parnami
Lecturer
IGCE
Mohali, India

Deepika Khokhar
Assistant Professor
IGCE
Mohali, India

Mayank Arora
Assistant Professor
CCET
Chandigarh, India

ABSTRACT

The usage of smart phones is increasing rapidly over the last few years. Due to their mobility and good connectivity, Smartphones are increasing thrice as compared to PCs. However they are still constrained by limited processing power, memory and Battery. Thus, the applications cannot be made very rich. In this paper we propose a framework for making the applications of these Smartphones intelligent enough, to offload their compute intensive parts from the smart phone to the virtual image of the Smartphone on the cloud thus using the unlimited resources of the cloud. By using this framework the application developers will be able to enhance the capabilities of the smart phones and will be empowered to make the applications even richer.

Keywords

Smartphone, Cloud computing, Offloading, Mobile cloud computing

1. INTRODUCTION

Cloud computing could be seen as a network service providing computational resources (hardware and software) on demand. A virtual pool of computational resources such as storage, CPU, networks, software etc. is created to fulfill the user's resource requirement and provides on demand hardware and software. Cloud computing enables seamless access to the user applications and data from anywhere anytime in the world thus making the user free from the confines of a PC and making collaboration of people easier irrespective of their location. The way of storing information and running applications has changed drastically and is still changing due to the emergence of Cloud Computing. Portability and scalability are some of the major reasons for the wide spread of Cloud computing.

With the advent of cloud computing the idea of mobile cloud computing also came forward. Mobile cloud computing means that any intelligent terminal equipments such as cell phones and personal computers can obtain services in wireless environment, which is also called mobile cloud computing. Mobile cloud computing integrates the advantages of mobile computing, mobile internet and cloud computing [10].

Smartphones are mobile phones with advanced computing capability, connectivity and rich set of functionality. In a nutshell, a Smartphone combines the functionalities of a phone, personal digital assistant (PDA) and a small computer. With the increasing popularity and a large number of developers developing applications for smartphones, the users of these phones have started using them for high end 3D gaming, to handle their finances i.e. internet banking and as their health and wellness managers (e.g. My Heart. my life app for Android [1]). These new applications could be very resource exhaustive and the phones have a limited memory, computational power and battery life. That's why it makes good sense to offload the heavy applications to the Virtual Smartphone running on the cloud, thus saving the actual phone's precious resources.

Most of the offloading techniques proposed use an application which needs to be installed in the phone, which makes the offloading decision for the other applications. Otherwise, changes are needed to be made at the operating system level. In the first case the application which is going to take the offloading decision for the other applications will be compute intensive and will impose overhead on the phone. Secondly, changes needed to be made at the O.S. level will not be practical. To fill these gaps, a different technique is proposed in this research which enables the application to take the offloading decision on its own. Using this technique the application developer will empower the applications to analyze the cost benefit of running itself on the phone versus the virtual Smartphone on the cloud and then take the offloading decision.

2. RELATED WORK

2.1 The reviewed offloading scheme

Quite a few approaches have been proposed for offloading applications from a Smartphone to the cloud, which includes offloading the complete application, offloading an application partially. Related work in the field of offloading applications from android phones to the cloud has been discussed below [2, 3, 4, 5, 6, 7, 8].

In 1998, Alexey Rudenko et al. [2] proposed a scheme to enhance the battery life of a laptop through wireless remote processing of power costly tasks. They proposed that the battery life of a laptop could be increased by shifting the power costly tasks on to a server through wireless connectivity or the internet.

Year 2009 witnessed the proposal of Augmented Smartphone Applications through Clone Cloud Execution by Byung-Gon Chun and Petros Maniatis [3]. This research proposed to augment the smartphone's capabilities by offloading an application partially or completely to a clone Smartphone. A clone is a virtual system on the cloud running the same operating system as that of the phone using hardware from the cloud's pool of hardware. The application is offloaded partially because only the part of the application which is compute intensive is to be offloaded and thus reducing the load on the Smartphone. While the compute intensive part is being executed by the clone the actual phone executes the remaining application. After the clone is finished with the execution of the compute intensive part of the application it returns the results to the actual phone. The phone processes the results as required and provides the user with the results.

Following the vision provided by the above research, Byung-Gon Chun et al. in the year 2011 implemented an architecture named Clone Cloud for offloading an application partially to its clone in the cloud. This scheme uses a partition analyzer which partitions the application to be offloaded for remote execution. The partition analyzer has a static analyzer which discovers the possible migration points and the constraints for migration and a dynamic profiler to build a cost model for execution and migration. The partition analyzer helps the migration unit to migrate and re-integrate the application at the chosen points. The migration unit comprises of a migrator, node manager and a partition database. The migrator provides the part of application to be migrated to the node manager which migrates the part to the clone and an entry is made to the partition database which helps in re-integrating the partitioned application [4].

In the year 2011, a new approach of offloading the applications from android Smartphone to the cloud was introduced by Eric Y. Chen and Mistutaka Itoh named Virtual Smartphone over IP. In this approach the complete application was offloaded from the android Smartphone to the cloud [5].

In 2012, Eric Y. Chen et al. introduced a framework for offloading heavy back-end tasks of a standalone android application to an android virtual machine in the cloud, which is an extension of Virtual Smartphone over IP. This architecture uses android interface definition language in order to offload without modifying the source code. This framework incorporates a dedicated server for each client to offload their application [5]. This architecture divides an application into two parts i.e. GUI and a compute intensive component and it offloads only the compute intensive component to the android virtual machine. If the cost of remote execution is less than the cost of local execution then the service is offloaded otherwise it is not offloaded to the virtual android phone [6].

In 2010, Georgios Portokalidis et al. proposed a new scheme named Paranoid Android to provide security to android phones by applying security checks on remote security servers that host exact replicas of the smartphones in virtual environments [7].

Another system named MAUI was introduced in 2010 by Eduardo Cuervo, Aruna Balasubramanian and Dae-ki Cho. MAUI enables fine-grained energy aware offload of mobile's code to a cloud infrastructure. MAUI uses code portability to create two versions of a Smartphone application, one of which runs locally on the Smartphone and the other runs remotely in the infrastructure. Managed code enables MAUI to ignore the differences in the instruction set architecture between today's mobile devices (which typically have ARM-based CPUs) and servers (which typically have x86 CPUs) [8].

In 2012, Aki Saarinen et al. proposed Smart Diet, a toolkit to identify the constraints that reduce offloading opportunities and to calculate the energy-saving potential of offloading communication-related tasks [9].

2.2 Constraints in the reviewed schemes

2.2.1 Overhead

The proposed schemes use Controller [3], Partition Analyzer [4], Service offloader [6] etc. which need to be installed in the phone as a system software or an application software. These applications will make the offloading decision for the other applications by analyzing their binary. These applications will incur an overhead on the Smartphone if the decision has to be made for a number of applications but only a few heavy applications need to be offloaded.

2.2.2 Security

The schemes discussed work at binary level. The binary modification requires changes in the program loader which in turn introduces security vulnerabilities.

2.2.3 Compatibility

The scheme [6] is not compatible with applications written in native languages like C/C++, .Net etc. because NDK (Native Development Kit) is not supported by the Helper tool.

2.2.4 Android Customization required

Android operating system is needed to be modified for the schemes [3-4] which may hinder its practical use as any customization in the Android operating system will require collaboration with Smartphone manufacturers and even Google.

3. PROPOSED ARCHITECTURE

The goal of this framework is to make an application autonomous, which will offload itself from the Smartphone to the Smartphone image on the cloud to reduce the cost of running the application on the phone's scarce resources. To achieve this goal an application will be divided into two major parts by the application developer at the time of application development i.e. a user interactive part which takes the inputs from the user and provides the output to the user and a compute intensive non-interactive part which does the computations as shown in the figure 1.

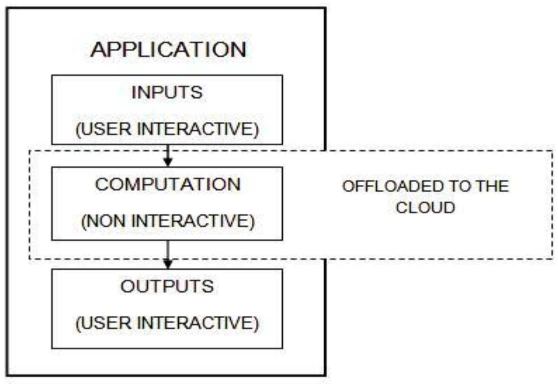

Fig. 1: Partitioning of applications into interactive and non-interactive parts.

Figure 2 shows an overview of an application using the framework. The application will become autonomous enough to analyze the cost of offloading itself partially and the cost of running on the phone. The compute intensive part of the application will then be offloaded to the cloud via internet if the cost of running it on the cloud would be less than the cost of running it on the phone. The analysis will be done using parameters like input size, battery available and internet connectivity.

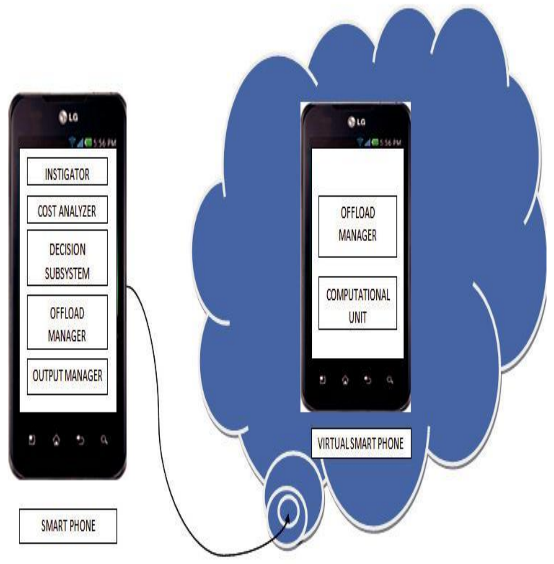

Fig. 2: Overview of autonomous computation offloading application for android phones.

Figure 2 shows an overview of an application using the framework. The application will become autonomous enough to analyze the cost of offloading itself partially and the cost of running on the phone. The compute intensive part of the application will then be offloaded to the cloud via internet if the cost of running it on the cloud would be less than the cost of running it on the phone. The analysis will be done using parameters like input size, battery available and internet connectivity.

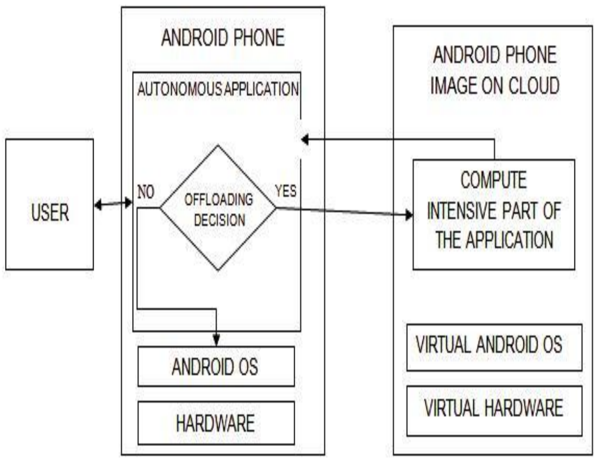

Fig. 3: Framework of autonomous computation offloading application for android phones.

3.1 Blocks on the Smartphone side

3.1.1 Instigator

The instigator unit validates the input provided by the user at the phone side in order to bypass the decision process if the input provided by the user is not in the required format or is corrupt.

3.1.2 Cost analyzer

Our cost analyzer considers the input size, energy consumption, battery status and internet connectivity to determine whether offloading the compute intensive part of a certain application would be beneficial or not.

3.1.3 Decision subsystem

The decision subsystem will take the offloading decision with the help of the inputs from the cost analyzer. If the cost of running the application on the cloud is less than the cost of running the application on the phone then the decision subsystem will initiate the offload manager which will in turn offload the application on to the virtual Smartphone image on the cloud otherwise the application will continue running on the phone.

3.1.4 Offload manager

Offload manager keeps in account the IP address, port number or URL of the virtual Smartphone on the cloud. According to our framework we assume that each Smartphone user will be having its dedicated virtual Smartphone using the resources of the cloud. We are not relying on opportunistic resources as most cyber foraging attempt to achieve. Dedicated resources are easily available through the cloud at very nominal cost. The offload manager will create the communication between the actual Smartphone and the virtual Smartphone.

3.1.5 Output manager

The output from the virtual Smartphone will be sent to the output manager by the offload manager. Output manager will do any further processing of the output if needed.

3.2 Blocks on the cloud side

3.2.1 Offload manager

The working of offload manager at virtual Smartphone side will be slight different from the offload manager at actual Smartphone side. The offload manager at virtual Smartphone side will be maintaining a communication link with the smartphone’s offload manager and will be sharing information with the Smartphone.

3.2.2 Computational unit

The computational unit will be responsible of running the compute intensive part of the application which is sending the data.

Figure 4 shows the control flow diagram of the proposed framework depicting the working of the framework. The user initiates the application and gives the required inputs. The inputs are fed to the instigator for validation and to the cost analyzer to generate a cost model. Depending upon the inputs and the cost model prepared by the cost analyzer the decision subsystem takes the offloading decision i.e. if the cost of running the application on the cloud is less than the cost of running the application on the phone, then the application is offloaded to the cloud otherwise the application continues running on the phone itself. The output of the computation is provided to the user by the output unit.

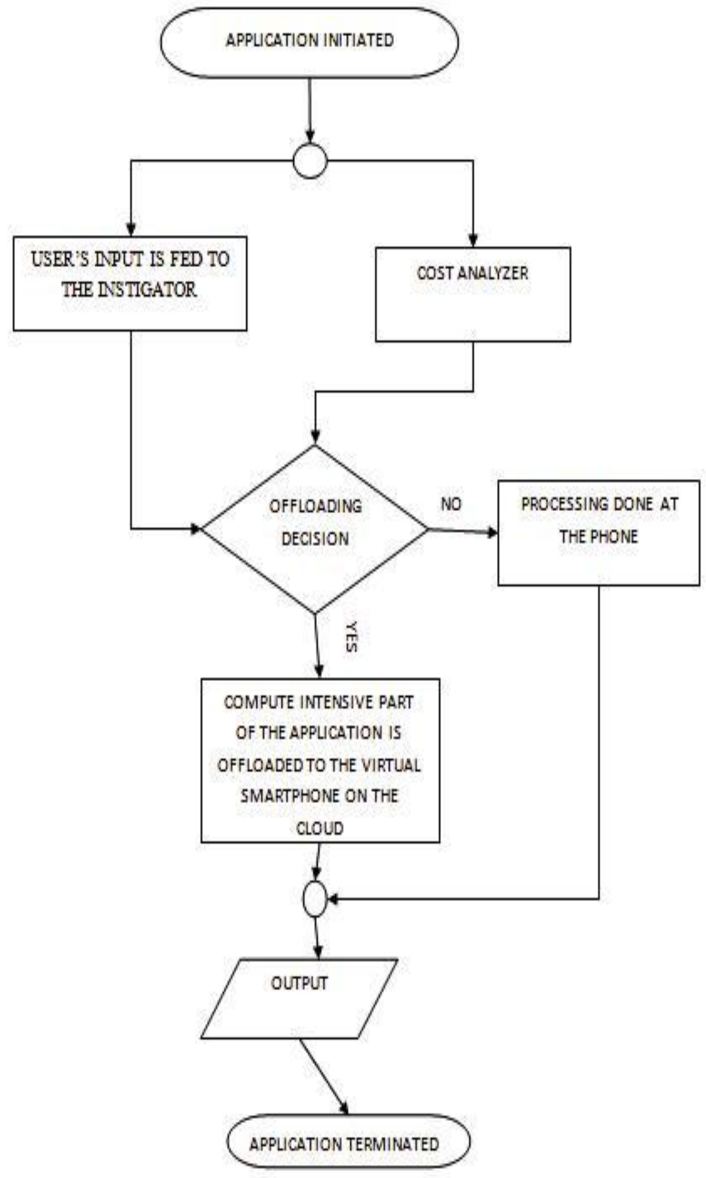
APPLICATION INITIATED
USER'S INPUT IS FED TO THE INSTIGATOR
COST ANALYZER
OFFLOADING DECISION
NO
PROCESSING DONE AT THE PHONE
YES
COMPUTE INTENSIVE PART OF THE APPLICATION IS OFFLOADED TO THE VIRTUAL SMARTPHONE ON THE CLOUD
OUTPUT
APPLICATION TERMINATED

Fig. 4: Control flow diagram of the proposed framework

4. IMPLEMENTATION

4.1 Simulation technology

The proposed architecture is implemented on Android OS version 2.3 as an Android native application framework. Any application being written in java for android OS version 2.2 can use these libraries.

To demonstrate the virtual device in The Cloud we are using Oracle VM Virtual Box to create the virtual smartphone on a Laptop.

We evaluate the benefits of the proposed framework by developing a prime number generating application in java using our framework. We developed two prime number generating applications, one without our framework i.e. a simple application to run on the phone and another application which is capable of offloading its compute intensive part. The application is developed for Android OS in java using Eclipse as an IDE.

4.2 Results and discussions

4.2.1 Measurement of execution time

To evaluate our framework we developed a prime number generator application. We ran that application on a phone i.e. monolithic execution without any offloading mechanism. We developed another application using our framework which could partition itself and offload to the virtual Smartphone on the cloud. The prime number generator we developed takes the size of the prime number to be generated as an input in bits and gives the prime number as the output. Table 1 shows the time required to generate prime numbers ranging from 8 Bits to 1024 Bits.

Table 1: Shows the comparison of execution time for a prime number generator application.

INPUT SIZE	EXECUTION TIME IN MILLISECONDS	
	SMARTPHONE	VIRTUAL SMARTPHONE(CLOUD)
8 Bits	13	16
16 Bits	17	20
32 Bits	39	32
64 Bits	56	46
128 Bits	131	73
256 Bits	255	115
512 Bits	6114	1801
1024 Bits	95312	28731
1500 Bits	238171	32120

We used Samsung Galaxy Y Duos Smartphone whose configurations are mentioned in Table 1 to run our standalone monolithic application to generate prime numbers ranging between 8 to 1500 Bits. The autonomous computation offloading application was tested using the same Smartphone as mentioned above and a Laptop to run the virtual Smartphone. The offloading decision depends upon the amount of computation required and the current battery status. The amount of computation required is calculated by the input size provided to generate the prime numbers and the processing power of the Smartphone. Table 1 shows the difference between the execution time required to generate the prime numbers. The disparity between columns 2 and 3 shows the opportunity to speed up the execution due to the difference between the cloud and phone's resources.

Using the autonomous offloading application we achieved up to 7X speed up and were able to generate prime numbers bigger than 1500 bits which was not possible on a standalone phone. When we generate a prime number which is very small in size i.e. 8 Bits or 16 Bits, the autonomous application takes a few milliseconds more than the standalone application because there is an extra overhead of the decision system. But as the size of prime number keeps on increasing and thus the computation power required increases the autonomous application starts offloading the compute intensive part to the virtual Smartphone on the cloud.

When the size of the prime number reaches 128 bits, we achieve almost 2X speed up because as the computation requirement is increasing for the application, the overheads of decision system and network usage starts seeming negligible in comparison to the computation cost. All measurements are average of three runs. If we try to generate a prime number greater than 1500 Bits the Smartphone stops responding if running the standalone application.

4.2.2 Measurement of battery consumption and CPU utilization

The key factors motivating the idea of offloading have been CPU and battery consumption. Thus, experiments were performed to analyze the battery consumption by the standalone application and the autonomous application using offloading techniques to generate prime numbers. To monitor the battery consumption by the applications we used a battery analyzer tool named Power Tutor [15]. Power Tutor is a power estimation tool that has been implemented for Android platform Smartphone. Power Tutor provides accurate, real-time power consumption estimates for power-intensive hardware components including CPU and LCD display as well as GPS, Wi-Fi, audio and cellular interfaces. It uses power models and automated characterization techniques [16-18].

Table 2: Shows the comparison of energy consumption by running the prime number generator application.

INPUT SIZE	ENERGY CONSUMPTION IN JOULES	
	SMARTPHONE	VIRTUAL SMARTPHONE(CLOUD)
128 Bits	4.2	5.3
256 Bits	7.3	6.3
512 Bits	18.8	7.0
1024 Bits	109.2	20.3
1500 Bits	308.2	58.5

As shown in table 2 the energy consumption by the standalone application is less when the input size is 128 bits because with this input size the autonomous application doesn’t offload its compute intensive part. As the input size goes on increasing, the autonomous application starts offloading its compute intensive part to the cloud thus the energy consumption reduces abruptly.

Another factor worth noticing is that the energy consumption constitutes of the energy consumed by the LCD, CPU and the Wi-Fi connection. The phone’s battery power is mainly consumed by the LCD or the CPU.

These applications were tested for five input sizes starting from 128 bits to 1500 bits. With the input size of 128 bits the autonomous application decides to run the application monolithically on the phone and thus consumes even more energy than the standalone application because of the extra processing required for the decision making. As the input size keeps on increasing the energy consumption keeps on decreasing as the autonomous application decide to offload the compute intensive part to the virtual Smartphone on the cloud.

5. CONCLUSION AND FUTURE SCOPE

Although several approaches for offloading an application from Smartphone to the cloud to save the scarce resources of the Smartphone have been proposed over the last few years, automated offloading of partial applications of android phones are still at an infant stage. The above literature survey reveals that a separate application is needed to offload applications by making changes at its binary level, but no study has been carried out on the effectiveness of offloading by making changes in the source code while developing an application, moreover the separate application needed to offload applications may incur overhead on the mobile phone in some cases. This research derives motivation from these concepts and it makes changes in the source code at the time of development to make an application autonomous which will offload itself from the Smartphone to the cloud if required and thus eradicating the need of a separate application to make the offloading possible. As revealed by the above literature survey most of the approaches proposed use static as well as dynamic analysis to take the offloading decision. There is a good potential of reducing the overhead of offloading decision by just using static analysis thus the framework proposed in this research uses static analysis. Furthermore, an extended framework is made using dynamic analysis and the comparison of computation overhead of both the frameworks i.e. the one using just static analysis and the other using dynamic analysis is done. The focus of the present work is to make an application autonomous enough to offload itself partially to the cloud based on a static or dynamic decision. Each time the application will run the decision will have to be taken by the application by doing the same analysis. However, for future extension the application can be made intelligent enough to learn from the previously taken offloading decisions and thus reducing the overhead of analysis for offloading decision and improving the overall performance. We are also planning to incorporate security, privacy, and trust related models [19-22] in the proposed framework.

6. REFERENCES

[1] Health and Wellness – *Phone Apps. University of California*,http://wellness.ucr.edu/wellness%20apps%20resources.pdf

[2] A. Rudenko, P. Reither, G. J. Popek and G. H. Kuenning : *Saving portable computer battery power through remote process execution*. In MCCR'98 – ACM SIGMOBILE Mobile computing and Communications Review Newsletter, Vol. 2, no. 1, 19-26 (1998).

[3] B. G. Chun, P. Maniatis : *Augmented Smartphone Applications Through Clone Cloud Execution.* In Proceedings of 12th conference on Hot topics in operating systems, 8-8 (2009).

[4] B. G. Chun, S. Ihm, P. Maniatis, M. Naik and A. Patti : *Clone Cloud : Elastic Execution between Mobile Device and Cloud*. In Proceedings of 6th Conference on Computer Systems, 301-314 (2011).

[5] E. Y. Chen and M. Itoh : *Virtual Smartphone over IP*. In Proceedings of IEEE international conference on World of Wireless Mobile and Multimedia Networks, 1-6 (2010).

[6] E. Chen, S. Ogata and K. Horikava : *Offloading Android Applications to the Cloud.* In Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 788-793 (2012).

[7] G. Portokalidis, P. Homburg, K. Anagnostakis and H. Bos : *Paranoid Android : Versatile Protection For Smartphones*. In Proceedings of the 26th Annual Computer Security Applications Conference, 347-356 (2010).

[8] 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, 49-62 (2010).

[9] A. Saarinen, M. Siekkien, Y. Xiao, J. K. Nurminen, M. Kemppainen and P. Hui, SmartDiet : *Offloading Popular App. To Save Energy*, ACM SIGCOMM Conference, 297-298 (2012).

[10] W. SONG and X. SU. : *Review of Mobile cloud computing*. In IEEE 3rd international conference on Communication Software and Networks, 1-4 (2011).

[11] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang and Q. Li : *Comparison of Several Cloud Computing Platforms*. In Proceedings of IEEE international conference on Information Science and Engineering, 23-27 (2009).

[12] National institute of Science and Technology: *The NIST Definition of Cloud Computing*.http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (2010)

[13] R. Buyya, J. Broberg and A. Goscinski : *Introduction to Cloud Computing*. In Cloud Computing: Principles and Paradigms. Wiley Press [Online], 1-44.http://media.johnwiley.com.au/productdata/excerpt/90/04708879/0470887990-180.pdf

[14] Android Open Source Project: Philosophy and Goals, Google.http://source.android.com/about/philosophy.html (2012).

[15] S. Singh and S. Bawa : *A Privacy, Trust and Policy based Authorization Framework for Services in Distributed Environment*. International Journal of Computer Science, Vol. 2, no. 1, 85-92 (2007).

[16] S. Singh and S. Bawa : *A framework for Handling Security Issues in Grid Environment using Web Services Seurity Specifications.* In Second International Conference on Semantics, Knowledge and Grid, 2006, SKG'06, Guilin, China, 68 (2008).

[17] G. Singh and S. Singh: *A Comparative Study of Privacy Mechanisms and a Novel Privacy Mechanism [Short Paper]*. In ICICS'09 Proceedings of the 11th International Conference on Information and Communication Security, Beijing, China, 346-358 (2009).

[18] S. Singh: *Trust Based Authorization Framework for Grid Services*. Journal of Emerging Trends in Computing and Information Sciences, (2011), Vol. 2, No. 3, 136-144.

[19] Power tutor application (2013)http://ziyang.eecs.umich.edu/projects/powertutor/

[20] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A break in the clouds: towards a cloud definition," SIGCOMM Computer Communications Rev., vol. 39, 2009, 50-55.

[21] "Google App Engines," Google code (2010).http://code.google.com/appengine/

[22] "API Dashboard," Programmable Web (2010)http://www.programmableweb.com/apis