Thoughtful Encounters in Mobile Computation Offloading to Cloud through Research

P.Mathivanan[1], Kodi.Satya Sridevi[2], S.Revathi[3], A.Sugandhi[4]

[1]P.Mathivanan, Assistant Professor, Dept. of Information Technology, Manakula Vinayagar Institute of Technology, Pondicherry, India
[2]Kodi.Satya Sridevi, Manakula Vinayagar Institute of Technology, Pondicherry & akshyasatyasri@gmail.com
[3]S.Revathi, Manakula Vinayagar Institute of Technology, Pondicherry & revathi221995@gmail.com
[4]A.Sugandhi, Manakula Vinayagar Institute of Technology, Pondicherry & sharmi150394@gmail.com

---***---

Abstract - *With rise in network connectivity and opportunity of low cost cloud properties, active task offloading to cloud give the impression to be an perfect solution to increase performance of mobile devices alongside with saving on battery consumption. Mobile Cloud Computing (MCC) allows covering limited properties available on mobile devices to execute complex and gorgeous mobile applications. This paper brings out the discoveries of the experimentations carried out to recognize the impression of application features, cloud end architecture and the android emulator used, on application performance when the application is increased to cloud.*

Keywords: *Mobile Cloud Computing, task offloading, emulator, android*

1. INTRODUCTION

Devices such as mobile phones and tablet computers, wear-able devices such as smart-watches, etc., are the new medium through which users carry out a substantial number of daily activities. Tipped to completely replace desktop computers in the years to come in terms of utility, mobile systems lag behind the former in terms of resources. Mobile systems have limited resources such as battery life, storage capacity and processor performance. This is where the concept known as computation offloading can be used for sending heavy computational activities within large applications to resourceful servers and receiving the results from these servers, thus, executing the applications on the mobile devices rather seamlessly. Many issues related to offloading have been investigated in the past decade. This paper first provides an overview of the techniques, systems and research areas for offloading computation, and then tries to understand the applicability of these techniques with respect to the type of applications, architecture followed, number of concurrent accesses, etc.

In complete contrast recent games are very much resource intensive in terms of processing and data transfer rates, this causes a huge drain on batteries of the smart-phones. Let us consider a scenario where a customer purchases a smart-phone which runs an application consuming more processing power such as an addicting game & starts playing that the game on his smart-phone. After a few minutes, when the events in that games have taken an interesting turn and his Smartphone stops running the game because of low battery. This scenario is very frustrating for the customer. But you would say that this example is only of games. Not everyone uses their mobile for playing games. So let's consider another scenario which would relate to the gravity of the situation. Consider an engineer working on a site and he is using smart-phone to make a video call to his boss. They are discussing about the implementation of a certain task in his project undertaking. Now suddenly he finds that his smart-phone is out of battery power. This situation is not only frustrating but could also cause loss and damage to the customer.

In the era of Cloud Computing (CC), the energy constraint on smartphones can be eased off by offloading heavy tasks from smartphones to the cloud. The mobile device can save energy by offloading heavy tasks to the cloud, and then the cloud executes the tasks and provides the mobile device with the results. For example, a smartphone can upload a video file to a cloud and request to encode the file into a desired format fitting the smartphone capability with less energy consumption than doing the encoding on the device.

Task offloading will become vital for the Information and Communication Technology (ICT) in the near future because CC will be a dominant operator for mobile computing. Mobile data storage and data processing will take place on the cloud, and a promising way to have this kind of ICT structure is to employ offloading techniques.

Task offloading is a critical technique because in some cases it increases the energy consumption of smartphones. To illustrate this, if a smartphone has to perform a task computation where task data exists on the smartphone, there are two scenarios: either execute the task locally (S1), or offload the task to the cloud (S2).The focus of this study is developing energy models to estimate the cost of task offloading caused by the networking activities. Specifically, we model the energy cost at the application level considering all the details of the network stack (i.e., Transmission Control Protocol (TCP), Media Access Control (MAC), and Physical layer (PHY).

The field of mobile computing has its origin in a fortunate alignment of interests by technologists and consumers. Since the dawn of the computing age, there have always been technological aspirations to make computing hardware smaller, and ever since computers became widely accessible, there has been a huge interest from consumers in being able to bring them with you (Atkinson 2005). As a result, the history of mobile computing is paved with countless commercially available devices. Most of them had short lifespan and minimal impact, but others significantly pushed the boundaries of engineering and interaction design. It is these devices, and their importance, that I wish to emphasize here.

2. LITERATURE

A lot of work has been done to study and develop computation offloading with the scenario we aimed [1], [2], [3]. A lot of work has been done on partitioning applications and various architectures [5], [6].

There are many approaches and debates about cloud computing. As it is now most recent research area especially in the information technology industry and education. Moreover, many applications about how cloud computing provides resources and computing infrastructure on the urgent demand from consumers in different sectors. Meanwhile, the consumers can use the services and applications on the cloud through internet.

There are many factors that make cloud computing an attractive technology, but energy consumption is a fundamental criterion for battery powered devices and needs to be care-fully considered for all mobile cloud computing scenarios. While energy can be a challenge for mobile cloud computing, it is also as an opportunity. Mobile cloud computing is therefore a fruitful area for further research. While the most energy efficient setup for many current mobile applications is local computing, there clearly are workloads that can benefit from moving to the cloud. The vastly superior computing resources available in the cloud open also interesting possibilities for completely new applications. Identifying these new applications is one interesting topic for future research [5].

The partition algorithm finds optimal program partitioning for given program input data. We use a heuristic method to deal with different program partitioning for different execution options. Experimental results show that the computation offloading scheme can significantly improve the performance and energy consumption on handheld devices. However, computation offloading scheme can still be improved. First, the cost model for execution time can be improved. More accurate time estimation models may get better result. Second, the point-to analysis used in our experiment is control and flow insensitive; we plan to implement more accurate analysis. Last, our computation offloading may be improved by dynamically adapting the program partition to the inputs and system information at run time [9].

For Accessing native methods, the methods accessing the phone's hardware API must be located on the mobile and also for the Shared variable; Methods which share state must be co-located. The nested deadlocking states that the methods blocking for migrated code may not migrate more code before the completion of previous migration. It also saves power and time with the computation [10].

Even though high performance often implies higher power requirements our examples show that high performance can also contribute towards better energy efficiency. This is especially true for wireless communication where achieving high energy efficiency requires high throughput. It is also important to realize that the performance metrics of real world scenarios can be significantly different from theoretical maximums implied by device components. For example radio throughput can be limited by device interconnects, processing capabilities and memory subsystem performance.

We see tools and technologies for managing the complexity of the issues involved as important topics for future research. Developers and content producers would benefit especially from tools that integrate seamlessly to the normal development flow. This requires models and estimation mechanisms that are sufficiently light-weight but still able to guide design decisions towards better energy efficiency.

Energy aware middleware solutions should therefore be researched to evaluate the feasibility of automatic decision making between local and remote processing [7].

The performance measurements of Xen lives up to its claim of high performance virtualization of the x86 platform. We find that Xen can easily support 16 moderately loaded servers on a relatively inexpensive server class machine, but falls short of the 100 guest target they set. We use this study as an example of repeated research and argue that this model of research, which is enabled by open source software, is an important step in transferring the results of computer science research into production environments [11].

3. EXPERIMENTATION

The experimentation on the scenarios made will be explained with the figure 1. Figure 1 shows the architecture followed for computation offloading [4]. An offloadable application will have a device component, which has the original application along with proxies for offloadable classes, and a cloud component which has all offloadable classes.

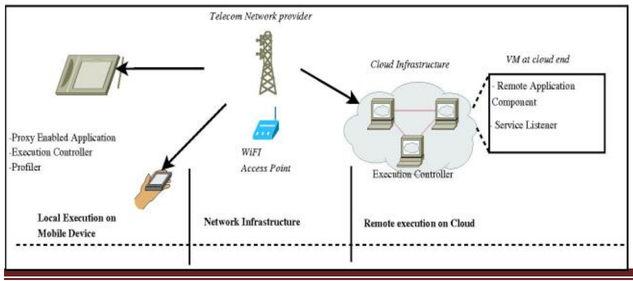

Fig. 1: Generalized Architecture for Computation Offloading

Profiler monitors various parameters such as available memory, CPU utilization, bandwidth, etc., and based on the profiler response an *Execution Controller* makes the computation offloading decision The *Proxy* forwards a function call to the remote cloud server if the function is offloaded to the cloud else it is executed on the mobile client. This offloading can be synchronous where mobile devices waits/polls cloud-end application component till results are available or it can be asynchronous where the application continues executing other tasks and receives notification from the cloud component whenever the results are ready. The connection to the cloud end can be through GSM /WCDMA (2G/3G) network or a WIFI hotspot connected to back-end wired network.

3.1 Scalability

In scalability related experiments we try to explore the impact of multiple clients trying to offload the computation. In this experimental set-up a VM is dedicated to an application and each mobile device acts as a client requesting execution on this application server. The client server architecture used in the experimentation even supports the dynamic offloading of application execution.

3.2 Application Selection

- *Emulator Impact on Offloading* - From Table I it is evident that the emulator has an impact on performance of an app when offloaded to cloud, and Genymotion pro-vides better performance as compared to AVD, because unlike AVD, Genymotion runs on top of x86 architecture through virtual box.

Applications can be developed for mobile platforms, which would run by utilizing dynamic offloading. Such applications are sparse in play store. Experiments were carried to identify suitability of applications for offloading based on its compute intensiveness, device interactions and coupling or interactive-ness among different modules of the application.

4. OBSERVATION

- *Emulator Impact on Offloading* - From Table I it is evident that the emulator has an impact on

TABLE I: N-Queen execution on different Emulators

N Value	AVD	Geny motion
10	1	0
11	7	0
12	39	0
13	218	4
14	611	29
15	4023	209
16	30000	1663

- *Impact of Offloading*- Application performance improves as the mobile device memory is increased. The compute intensive applications such as N-Queen, crash or take a lot of time on real or virtual mobile devices for higher values of N even with good memory such as 1 or 2 GB. However, when the compute intensive portion is offloaded to the cloud set-up, the applications are executed within seconds. Thus, it can be concluded that performance of suitable applications can be drastically improved when compute-intensive portions of the applications are offloaded to the cloud.

- *Application Suitability for Offloading* - Though compute intensive applications performance improve with offloading, not all applications benefit from the same. The experimentation.

- *Computational Performance* - It proves that the performances of applications that are GUI-intensive, have less computation, and are interactive degrade when offloaded. It is because for these applications execution time is less as compared to network latency.

- *Impact of Multiple Clients* - When multiple clients access the server at the same time, the performance has been seen to be reduced, resulting in each client taking more time than expected. As can be observed from the Table II, multiple client connections were not hampering the performance expected until the chosen hardware could not handle the requests. The performance can be considered similar to those of client-server applications.

TABLE II: Effect of Multiple Clients

N	1 Client	2 Clients	3 Clients	4 Clients	5 Clients
12	1	1	1	1	1
13	3	3	3	4	4
14	19	19	20	21	22
15	131	133	159	165	198
16	972	972	974	1184	1336

- *Offloading Performance* - With offloading, all devices should be able to give equal computational performance for an application as the same back-end infrastructure is working for all. The mobile device will act as only thin client depending upon the profilers output. With different approaches in existence for defining the cloud-end architecture, the scalability could impact the performance of applications after offloading.

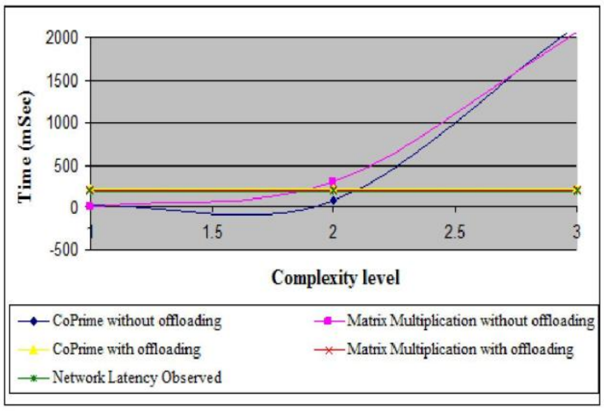

Fig. 2: Optimal Condition for Offloading

- *Optimal Condition for Offloading* - Application perfor-mance degrades with offloading if the computation is less in whereas it improves drastically for higher computing requirements. Figure 2 shows how to get the optimal condition for offloading. This can be one of the input for the controller that decides" to offload or not to offload". OR The optimal condition for offloading has already been implied in Application Suitability for Offloading. This can be one of the input for the controller that decides" to offload or not to offload".

5. CONCLUSIONS

In our project, the major challenge in task offloading is to estimate accurately the energy consumed during the network activities of task offloading.. For that, we have designed and implemented a decision framework for computation offloading. The decision is based on estimated execution time and energy consumption. However, estimating the energy consumed in task offloading is crucial to making task offloading beneficial, which happens only when the energy consumed in the offloading process is less than the energy consumed without it. We aim to save both execution time and energy consumption at the same time. Unlike previous works, which consider only binary decisions, our ternary decision is suitable for multiple offloading targets. In our experiment, we presented a case study to validate the applicability in different situations. Based on our decision framework, the Task module tends to be offloaded to more powerful processors, such as local GPU or cloud.

REFERENCES

[1] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: Elastic execution between mobile device and cloud. In *EuroSys, Salzburg, Austria*, pages 80–86, April 2011.

[2] Byung-Gon Chun and Petros Maniatis. Augmented smartphone applica-tions through clone cloud execution. In *HotOS 2009*, 2009.

[3] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chan-dra, and P. Bahl. Maui: Making smartphones last longer with code offload. In *Proceedings of the The 8th Annual International Conference on Mobile Systems, Applications, and Services (ACM MobiSys)*, pages 49–62, 2010.

[4] Abhishek Dwivedi, Padmaja Joshi, and Abhay Kolhe. Mobile stand-alone application code off-loading: Architecture and challenges. *International Journal of Computer Applications (0975-8887)*, 94:22–27, May 2014.

[5] Jiwei Li, Kai Bu, Xuan Liu, and Bin Xiao. Fast dynamic execution offloading for efficient mobile cloud computing. In *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing (ACM SIGCOMM)*, pages 39–44, 2013.

[6] Sehoon Park, Youngil Choi, Qichen Chen, and Heon Y. Yeom. Some: Selective offloading for a mobile computing environment. In *IEEE International Conference on Cluster Computing*, pages 588–591, 2012.

[7] 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.*

[8] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A Survey of Computation Offloading forMobile Systems,"*Mobile Networks and Applications, pp. 1–12, 2012.*

[9] C. Wang and Z. Li, "A computation offloading scheme on handheld devices", J.Parallel and Distributed Computing, vol.64, no.6, pp.740-746, June 2004.

[10] C. Wang and Z. Li, "A computation offloading scheme on handheld devices", J.Parallel and Distributed Computing, vol.64, no.6, pp.740-746, June 2004.

[11] P.Barhamet al., "Xen and the art of virtualization", in Proc. 20th ACM Symp. Operating Syst. Principles, Lake George, NY, 2003, pp.164-177.