Overview of Mobile Computation Offloading in Mobile Cloud Computing Environment

Hsu Mon Kyi
University of Computer Studies (Taunggyi)
hsumonkyi.ucsy@gmail.com

Abstract

Mobile device functionality is ever richer in daily life, but they still lack the required amount of resources when it comes to handling resource-hungry applications. Nowadays, Mobile Cloud Computing (MCC) bridges the gap between the limited capabilities of mobile devices (battery life, network bandwidth, storage capacity, processor performance) and the increasing user demand of mobile applications, by offloading the computational workloads from local devices to the remote cloud. This paper provides an overview of the back-ground, techniques, and research areas for computation offloading. Moreover, the issues, existing solutions, and approaches are presented.

Keywords: Mobile Device**,** Network Bandwidth**,** Mobile Cloud Computing, Computation Offloading

1. Introduction

In our daily life, mobile devices have become common entity. 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. These mobile devices provide us with many more exciting applications which require large computing power, memory, network bandwidth and energy to run applications as multimedia, GPS navigation, real time games etc. which also adds energy consumptions constantly. Energy or battery is the only resource in mobile devices that cannot be restored immediately and needs external resources to be renewed.

Computation Offloading is a solution to augment these mobile systems' capabilities by migrating computation to more resourceful computers (i.e., servers). These servers may use virtualized cloud server to provide offloading services. Offloading is typically used to augment the computational capability of a resource constrained device for a single user. Computational offloading occurs at the code level in which an application is partitioned or analyzed before its development. It can also be achieved at runtime by analyzing the application on the basis of local device parameters such as network availability, battery life, size of the application, etc. The concept of offloading is optional as it is not feasible every time to offload an application to its surrogate on the cloud. The terms surrogate computing and cyber forging are also used to address computational offloading.

2. Related Work

A significant amount of research has been studied computation offloading on various viewpoints. For example, Dinh et al. [7] summarize the overall idea of computation offloading, and discuss its different applications. Shiraz et al. [13] discuss different methods of implementing offloading from smart phones. Abolfazli et al. [2] study the impact of types of cloud resources used on computation offloading. Liu et al. [9] focus on the offloading algorithm used by different offloading frameworks. Finally,

Shaukat et al. [11] provide taxonomy of solutions that use cloudlets, which are defined as resource rich computers in proximity to mobile devices. This paper provides an overview of the motivations, techniques, and architectures for computation offloading. It surveys the common approaches used to make offloading decisions.

The rest of this paper is organized as follows. Section 3 presents the background theory including definition, architecture, and its applications. Then, Section 4 presents several issues and future research directions are outlined. Finally, the paper is summarized and concluded in Section 5.

3. Background Theory

Computation offloading, as one of the main advantages of MCC, is a paradigm/solution to improve the capability of mobile services through migrating heavy computation tasks to powerful servers in clouds. Computation offloading yields saving energy for mobile devices when running intensive computational services, which typically deplete a device's battery when are run locally.

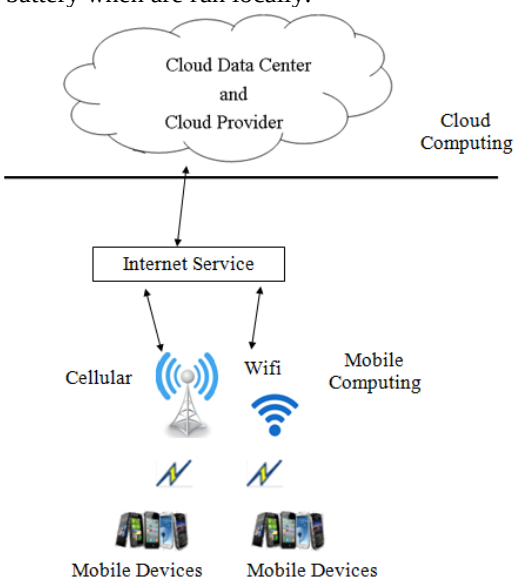

Figure 1. Mobile Cloud Computing

Nowadays, virtualization techniques enable cloud computing environments to remotely run services for mobile devices; there has been a significant amount of research focusing on computation offloading. These research themes are mostly related to explore ideas/ways to make computation offloading feasible, to draw optimal offloading decisions, and to develop offloading infrastructures. There are many factors that can adversely affect the efficiency of offloading techniques, especially bandwidth limitation between mobile devices and servers in cloud and the amounts of data that must be exchanged among them.

Many algorithms have already been proposed to optimize offloading strategies to improve computational performance and/or save energy. These algorithms and techniques mostly analyze a few system parameters including network bandwidths, computation capability, available memory, server loads, and the amounts of exchanging data between mobile devices and cloud servers to propose offloading strategies. The real world performance benefits and battery gains are achieved by offloading certain amount of computational work from a mobile device to a cloud server. The system found a clear gain in terms of the load on the CPU of the device as well as the battery life consumption.

The concept of Mobile Computational Offloading provides a solution for the execution of resource-hungry applications. Computation Offloading occurs at the code level in which an application is partitioned or analyzed before its development. While offloading computation to a cloud server, there are two important factors to keep in mind. The first factor is the size of the computation being performed and the second factor is the amount of data that needs to be sent and received for the computation to be successful.

Mobile Cloud Computing system consists of three parts, cloud, mobile clients, and wireless network. Cloud offers applications as services. Mobile clients access application services through proxies using wireless communication and proxies communicate with the cloud servers over fixed wired network.

The following section will study the variety of offloading application areas and architectural views of computational offloading system.

3.1. Application Areas

Leading mobile software repositories such as Google Play or Apple App Store have over a million different applications. Computation offloading can improve the performance of a variety of applications, including text editors, virus scanners, games, object and gesture recognition, face detection, speech recognition and indoor map reconstruction. These applications impose varying requirements on the mobile system.

The application level variations are reflected in the following parameters.

Degree of concurrency: Most smartphone applications are concurrent. There are two types of concurrency - task-level and data-level. Applications that execute independent tasks have task-level concurrency. Programs with stream data, such as multimedia applications, have data-level concurrency.

Workload Heterogeneity: Smartphones usually execute multiple applications with different user expectations. For example, a user might use a social networking app while listening to music. Each of these different applications may impose varying workload on the mobile system. A video streaming application imposes high computation and network costs. On the other hand, a social networking app imposes less computation cost, but utilizes the network after fixed duration.

Real-time Constraints: Mobile applications may or may not have real-time constraints. There are two types of real-time constraints. Soft constraints, usually found in streaming applications, require limited response time for only a proportion of the given tasks. On the other hand, hard constraints require guaranteed response time for each task. They are found in gaming applications.

3.2. General Offloading Architecture

Computational offloading architecture can be described by the means of different architecture views, namely: image offloading, method offloading and feature offloading.

Image Offloading: It is a complex and difficult approach towards computation offloading. In the case of image offloading, the entire code is offloaded to virtual environments on the cloud. The most prominent example used to describe image offloading is CloneCloud.

Feature Offloading: In feature offloading, a data set is prepared by the mobile OS, which is sent to the cloud. The execution of an application on the cloud depends on this data set. The cuckoo architecture uses the concept of feature offloading.

Method offloading: It involves the complete transfer of execution of a subroutine to the cloud. It is very much similar to feature offloading with some exceptions. MAUI architecture uses the concept of method offloading.

Most related research efforts on computation offloading for mobile devices include CloneCloud [14], MAUI[14] , Odessa[1] and OSMOS[1]. In the following, the paper is described their principles from the application modeling to the system architecture.

3.2.1. CloneCloud

Figure 2 shows high level architecture of the Clone Cloud. The main components of the Clone Cloud are migratory, node manager, profiler (dynamic profiler), database and partition analyzer.

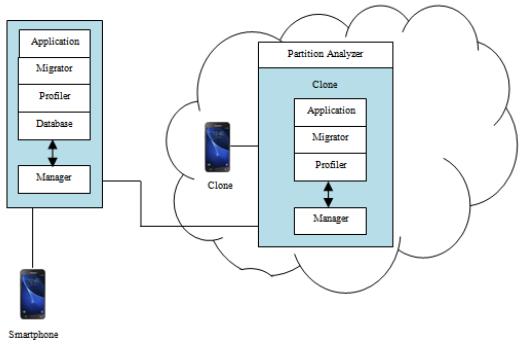

Figure 2. Architecture of CloneCloud

The migrator is responsible for resuming, suspending, merging and packaging the thread states on both sides. The node manager performs image synchronization, provisioning and handles communication between the migrated threads. For three tasks Chun et al. tested Clone Cloud prototype i.e., virus scan, behavior profiling and image search. The advantage of this model is the recovery of data when a Smartphone is lost or destroyed.

3.2.2. MAUI (Mobile Assistance Using Infrastructure)

The main objective of this model is to minimize energy consumption of mobile devices, which is the first main challenge of the mobile industry. MAUI offloads all the resource intensive methods to the nearby infrastructure or cloud, provided the offloading is beneficial in terms of energy. MAUI uses a profiler (optimization engine) that helps in analyzing energy consumption involved in the local and remote execution of the code. In MAUI, the application partitioning is dynamic. The offloading is done on the basis of methods instead of complete application modules to minimize the offloading delay. For local and remote execution MAUI creates two versions of Smartphone application, which uses Microsoft .NET Common Language Runtime (CLR). The architecture of MAUI is shown in Figure 3.

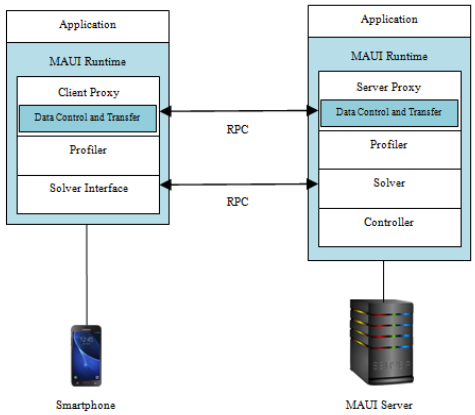

Figure 3. Architecture of MAUI

In MAUI, the mobile device consists of three main components, i.e., solver interface, profiler and client proxy. The interaction with the solver is provided by solver interface (decision engine) and improves the offloading decision making. The information regarding the application energy consumption and data transfer is collected by profiler. The client proxy concentrates on the method offloading and data transfer. Similarly, the server side consists of server proxy, profiler, solver and controller. However, the profiler and server proxy working is similar to the smartphone. The solver is the main decision engine of the MAUI that holds the scheduled methods and call graph of the applications. Lastly, the controller is responsible for resource allocation and authentication for incoming requests. The energy consumption is involved in the offloading procedure and MAUI focuses on it. MAUI will not be able to offload those methods which are not marked by programmer (for remote execution). Also, MAUI uses online design to create an energy consumption model and it also saves information about the offloaded methods.

3.2.3. Odessa

Odessa is a simple programming model that focuses on performance complex executions which are hidden from the application

developers. The system was constructed to utilize the computing power of different multi-core machines. It involves the use of three techniques, namely, offloading, pipelining and data parallelism. Pipelining divides the application into different stages and allows them to run in parallel frames thus increasing the throughput. The goals of Odessa architecture are increase the throughput, low computational and communication overhead and enhance the limited capacity of mobile device.

3.2.4. COSMOS (Composable, Open, Scalable, Mobile-Oriented System)

Another architecture used for computational offloading is COSMOS. It solves the mismatch that exists between the resource requirements of mobile device and the resources rendered by the cloud provider. COSMOS architecture comprises three components, namely, COSMOS master, COSMOS servers and COSMOS clients. Master gathers information about different workloads on various COSMOS servers. It decides upon the existence of a COSMOS server, i.e., whether a server is active or not. COSMOS server is to execute the offloaded tasks of a mobile user. COSMOS client decides whether an application should be offloaded or not based on resource consumption and resource requirements of an application.

Table 1. A List of applications that have been optimized using offloading

Application	Offloading Solution
Face recognition	MAUI[6]
Image search	CloneCloud[5]
Virus-scanning	CloneCloud[15]
Chess game	CloneCloud[5]
Speech recognition	COSMOS[12]
Gesture recognition	Odessa[10]

4. Issues and Future Directions

In this section, this paper summarizes the challenges and issues that still remain towards computational offloading. Offloading frameworks currently need to adapt some of the parameters, and focus on improving only a few Quality of Experience (QoE) parameters such as completion time, energy consumption, network usage, server usage and security.

Moreover, the various factors and emerging trends that formulate the future of offloading are described in this section. In the past few years, two important trends have occurred:

Evolution of Smartphones: The processing power of smartphones would outshine its battery power thus resulting in offloading to become an obvious solution.

The growth of Internet: The rise in the Internet technologies like IoT would lead to a drastic increase in multimedia data that could only be managed and monitored through the cloud.

5. Conclusion

In this paper, the potential of cloud-based computation offloading have been investigated. A computation offloading techniques used on handheld devices are presented. The paper classifies the types of applications that have been used to demonstrate offloading. Moreover, the paper presents the different parameters that are considered in offloading systems to better support user expectations.

References

[1] A.K.Sarishma, "*Mobile Cloud Computing Principles and Paradigms*", International Publishing House Pvt. Ltd, 2016.

[2] Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., & Buyya, R, "Cloud-based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges", IEEE Communications Surveys & Tutorials, 2014, pp. 337–368.

[3] A.Bonkra, "Survey on Computational Offloading In Mobile Cloud Computing", *International Journal of Engineering Science and Computing*, May, 2011.

[4] A.Bhattacharya, "A Survey of Adaptation Techniques in Computation Offloading", *Journal of Network and Computer Applications*, November, 2016.

[5] Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., & Patti, "Clonecloud: Elastic Execution between Mobile Device and Cloud", In Proceedings of the Sixth Conference on Computer Systems EuroSys , 2011, pp. 301–314.

[6] Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P, "MAUI: Making Smart phones Last Longer with Code Offload", In Proceedings of the 8th International

[8] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu Bharat Bhargava, "A Survey of Computation Offloading for Mobile Systems", Springer Science and Business Media, LLC 2012.

[9] Liu, J., Ahmed, E., Shiraz, M., Gani, A., Buyya, R., & Qureshi, A, "Application Partitioning Algorithms in Mobile Cloud Computing: Taxonomy, Review and Future Directions", *Journal of Network and Computer Applications*, 2015, pp. 99–117.

[10] Ra, M.-R., Sheth, A., Mummert, L., Pillai, P., Wetherall, D., & Govindan, "Odessa: Enabling Interactive Perception Applications on Mobile Devices", In Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, 2011, pp. 43–56.

[11] Shaukat, U., Ahmed, E., Anwar, Z., & Xia, "Cloudlet Deployment in Local Wireless Networks: Motivation, Architectures, Applications, and Open Challenges", *Journal of Network and Computer Applications*, 2016.

[12] Shi, C., Habak, K., Pandurangan, P., Ammar, M., Naik, M., & Zegura, "COSMOS: Computation Offloading as a Service for Mobile Devices", In Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing , 2014, pp. 287–296.

Conference on Mobile Systems, Applications, and Services MobiSys , 2010, pp. 49–62.

[7] Dinh, H. T., Lee, C., Niyato, D., & Wang, P, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches", Wireless Communications and Mobile Computing, 2013.

[13] Shiraz,M., Gani, A., Khokhar, R., & Buyya, " A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing", IEEE Communications Surveys & Tutorials, 2013, pp. 1294–1313.

[14] Vinayak D. Shinde, Usha S Patil, Anjali Dwivedi, "Survey on Application Models using Mobile Cloud Technology", *International Journal of Advanced Research in Computer and Communication Engineering* ,Vol. 4, Issue 10, October 2015.

[15] Zhang, W., Wen, Y., & Zhang, X, "Towards Virus Scanning as a Service in Mobile Cloud Computing: Energy-Efficient Dispatching Policy under n-version Protection.", IEEE Transactions on Emerging Topics in Computing, 2015.