A Study of Task Offloading in Mobile Cloud Computing

Sai Vamsidhar Rao V[1], Dr. Neeraj Sharma[2]

[1]Research Scholar of Sri Satya Sai University of Technology & Medical Sciences, Sehore, M.P., India.
[2]Research Supervisor of Sri Satya Sai University of Technology & Medical Sciences, Sehore, M.P., India.

Abstract

Despite the evolution and enhancements that mobile devices have experienced, they are still considered as limited computing devices. Today, users become more demanding and expect to execute computational intensive applications on their smartphone devices. Therefore, Mobile Cloud Computing (MCC) integrates mobile computing and Cloud Computing (CC) in order to extend capabilities of mobile devices using offloading techniques. Computation offloading tackles limitations of Smart Mobile Devices (SMDs) such as limited battery lifetime, limited processing capabilities, and limited storage capacity by offloading the execution and workload to other rich systems with better performance and resources. This paper presents the current offloading frameworks, computation offloading techniques, and analyzes them along with their main critical issues. In addition, it explores different important parameters based on which the frameworks are implemented such as offloading method and level of partitioning. Finally, it summarizes the issues in offloading frameworks in the MCC domain that requires further research.

Keywords: *Cloud computing; Mobile cloud computing; Computational offloading; Algorithms; Partitioning*.

1. INTRODUCTION

The main goal of CC is to allow IT departments to focus on their businesses and projects instead of just taking care of their data centers and keeping them working [2,18,20]. CC is a new concept that aims to provide computational resources as services in a quick manner, on demand, and paying as per usage. The CC paradigm is presented in three cloud delivery models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) as shown in Fig. 1. According to Gartner [3], CC will have in 2016 a Global Compounded Annual Growth Rate (CAGR) of IaaS: 41%, PaaS: 26.6% and SaaS: 17.4%. Recently, user preferences for computing have changed because of the latest developments and enhancements in mobile computing technologies. Several reports and studies have presented the importance of MCC and its impact on mobile clients and enterprises. For instance, and according to a recent study by ABI Research, more than 240 million business will use cloud services through mobile devices by 2015 and this will push the revenue of the MCC to $5.2 billion [11]. Moreover, the usage of smartphones has increased rapidly in various domains, including enterprise, management of information systems, gaming, e-learning, entertainment, gaming, and health care. Although the predictions that mobile devices will be dominating the future computing devices, mobile devices along with their applications are still restricted by some limitations such as the battery life, processor potential, and the memory capacity of the SMDs. Nowadays, modern mobile devices have sufficient resources such as fast processors, large memory, and sharp screens. However, it is still not enough to help with computing intensive tasks such as natural language processing, image recognition, and decision-making. Mobile devices provide less computational power comparing to server computers or regular desktops and computation-intensive tasks put heavy loads on battery power.

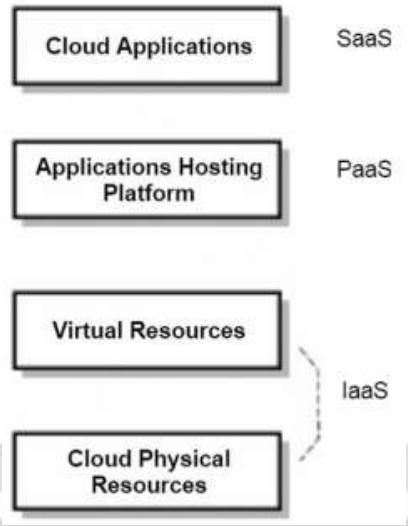

Figure 1 Cloud computing layers

Currently, there are several works and research in CC that aim at enhancing the computing capabilities of resource constrained mobile client devices by providing mobile clients access to cloud infrastructures, software, and computing services. For example, Amazon web services are used to protect and save clients' personal data via their Simple Storage Service (S3). In addition, there are several frameworks that allow to process data intensive tasks remotely on cloud servers. For instance, the ASM computation offloading framework [6] showed that computation offloading helped to reduce the energy consumption cost of mobile devices by 33%, and the turnaround time of the application by 45%.

Cloud computing

CC is a new way of providing computing resources and services. It refers to an on-demand infrastructure that allows users to access computing resources anytime from anywhere [25]. CC offers to users and business three main advantages: (1) enormous computing resources available on demand, (2) payment for use as needed and on a short-term basis (storage by the day and release them as needed), and (3) simplified IT management and maintenance capabilities [1]. CC provides clients with different applications as services via the Internet. As examples of public CC we can list Windows Azure and Amazon Web Services (AWS). Windows Azure is an open and flexible cloud platform which provides several services to develop deploy and run web applications and services in cloud data centers. AWS, which is considered as an example of a public computing tool, provides users with two models: infrastructure as a service and software as a service. These services allow the user to use virtualized resources in cloud data centers. Computational clouds implement a variety of service models in order to use them in different computing visions [4].

Mobile cloud computing

MCC can be seen as a bridge that fills the gap between the limited computing resources of SMDs and processing requirements of intensive applications on SMDs. The Mobile Cloud Computing Forum defines MCC as follows: ''Mobile Cloud Computing at its simplest form refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smartphone users but a much broader range of mobile subscribers". MCC has attracted the attention of business people as a beneficial and useful business solution that minimizes the development and execution costs of mobile applications, allowing mobile users to acquire latest technology conveniently on an on-demand basis. Fig. 2 shows the general view of MCC which is composed of three main parts: the mobile device, wireless communication means, and a cloud infrastructure that contains data centers. These latter provide storage services, processing, and security mechanisms for both the cloud environment and mobile devices.

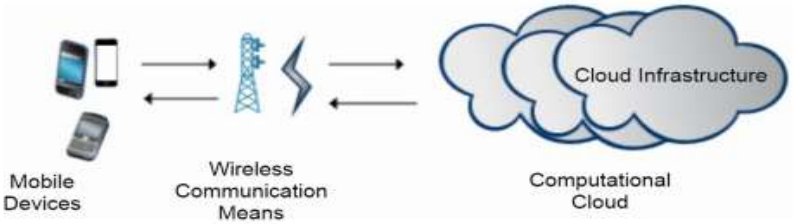

Figure 2 General view of MCC

Computation offloading

Computation offloading is the task of sending computation intensive application components to a remote server. Recently, a number of computation offloading frameworks have been proposed with several approaches for applications on mobile devices. These applications are partitioned at different granularity levels and the components are sent (offloaded) to remote servers for remote execution in order to extend and enhance the SMD's capabilities. However, the computation offloading mechanisms are still facing several challenges. In the remaining part of this section, our objective was to give a summary about the MCC offloading research by discussing the following:

1. Usage scenarios for offloading in MCC.

2. Techniques being applied in offloading.

3. A classification of proposed offloading frameworks.

Computation offloading emerged around 1970s. However, its potential has been widely explored only when wireless communication and Internet speed became sufficiently enhanced and could support it. The potential of mobile offloading mainly depends on the mobile network technologies such as cellular and WiFi. They determine the viability of mobile offloading. Today, the WiFi technology is able to provide high bandwidth connections. However, the data transmission using the cellular network requires a considerable amount of energy from the mobile device as opposed to a WiFi network.

Fig. 3 illustrates the environment that supports computation offloading. In this overview, the mobile device decides to offload method B to a cloud server or a powerful machine. The cloud here provides the virtual computation resources to run the offloaded components. The powerful machine can be a server or cluster in a computing center, or a computing grid, or a virtual server in the cloud. Fig. 4 shows the number of published papers since 2004 citing the word ''Offloading" and ''Computation". Most of the research works tackling data offloading have the goal to store data in remote large repositories. It can be seen in Fig. 4 that the research work citing ''computation offloading" and ''data offloading" is increasing progressively. Clearly, computation offloading is worthwhile only when the local execution (mobile device) consumes more time and energy than the offloading overhead. Many factors can impact the offloading decision and could influence the offloading process.

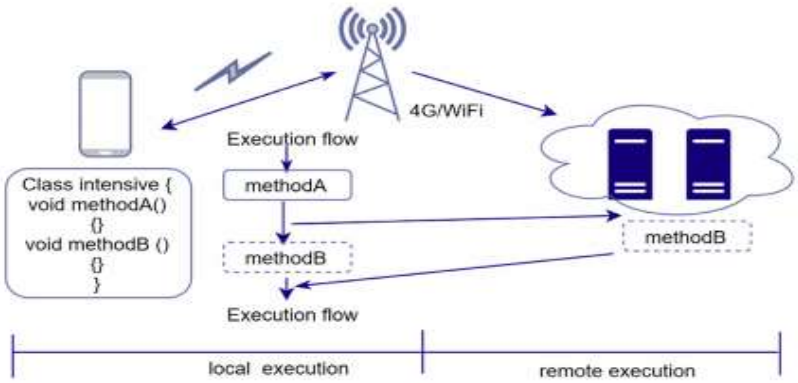

Figure 3 Offloading process overview

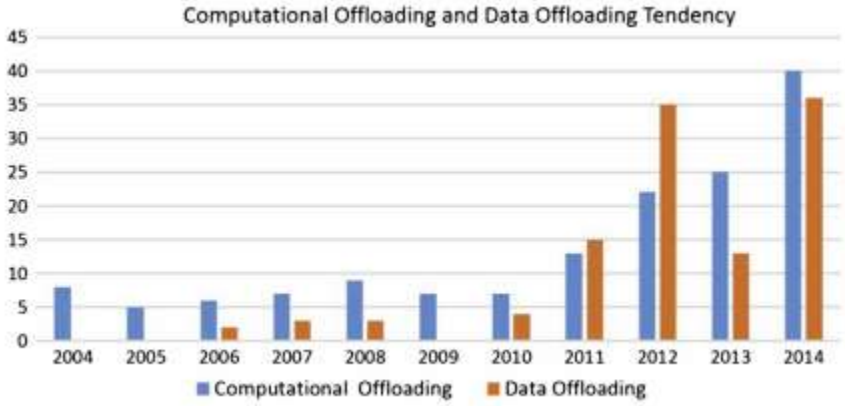

Figure 4 Number of computation offloading and data offloading papers

As it is known, the battery life of mobile devices and the limited processors' capabilities remain key limiting factors in the design of mobile applications. Today, the demand for resource intensive applications such as 3D video games and voice recognition is increasing day by day. To close this gap between the users demand and the mobile devices limitations, research studies have been exploring computation offloading in MCC to bring the power of cloud computing to the otherwise limited mobile devices capacity.

2. COMPARISON OF OFFLOADING FRAMEWORKS IN MOBILE CLOUD COMPUTING

This section introduces different existing offloading frameworks. For each of the frameworks we identify the approaches used in the three steps introduced in the previous section. At the end of the section, the different frameworks will be compared with respect to their most important properties.

1. Clone Cloud

The Clone Cloud framework which aims at improving the battery life and performance on the mobile device by offloading intensive components to cloud servers. The partitioning step in this framework combines static program analysis with program profiling to produce a set of off loadable components while meeting some constraints, such as methods that use mobile sensors should be executed locally. The framework uses thread level granularity for partitioning of applications. The role of static analysis is to discover constraints on possible migration points while profiling aims to build a cost model for offloading and execution. Partitioning and integration of the applications are performed at the application level. As a preparation step, a duplicate of the smartphone's software is captured and stored within a cloud server. At runtime, the offloading decision is taken and threads are migrated from the mobile device to the clone in the cloud. In Clone Cloud, threads must be suspended, all states of the threads must be transferred to the server, and then threads resume on the server in order to offload computation. The framework is based on VM instance migration to the cloud server. Initially, a duplicate of the smartphone's software is created in the cloud. The state of the smartphone and the clone is synchronized periodically or on-demand. After the execution of offloaded components, results from the execution on the clone are reintegrated back into the smartphone state. Clone Cloud employs dynamic offloading. The objective of the distributed execution mechanism in Clone Cloud was to implement a specific partition of a given application process executing inside an application-layer virtual machine. In Clone Cloud framework, the distributed execution goes through several steps as follows. When the user tries to run a partitioned application, the framework looks in a database of pre-computed partitions for current execution conditions (such as available network bandwidth and cloud resources). The result of the verification is a partition configuration file. The partition is loaded by the application binary which instruments the selected methods with migration. On the mobile device, once the execution of the process reaches a migration point, the running thread is suspended and its state is wrapped and shipped to a synchronized clone. In this clone, the thread state is instantiated into a new thread with the same stack and reachable heap objects, and then resumed. On the cloud server, when the migrated thread reaches a re-integration point, it is suspended, packaged, and then shipped back to the mobile device. Finally, the received packaged thread is merged into the state of the original process in the mobile device. To evaluate the prototype, the authors implemented three applications (virus scanner, image search, and privacy preserving targeted advertising). All measurements are the average of five executions. Phone, Clone Cloud

with WiFi, and Clone Cloud with 3G are the used environments. A clear tendency is that larger workloads benefit from offloading because of amortization of the migration cost over a larger computation [6].

2. MAUI

MAUI [8] is a framework that considers energy saving on smartphones as the main objective function for the offloading process. MAUI is a highly dynamic offloading framework because of a continuous profiling process. The framework hides the complexity of a remote execution from the mobile user and gives the impression as if the entire application is being executed on the mobile device. In MAUI, partitioning is done based on developer annotations to specify which components can be executed remotely and which cannot. In the preparation step, two requirements should be met: (1) application binaries must be in both mobile and server sides and (2) proxies, profilers and solvers must be installed on both the mobile device and server sides. At the beginning, MAUI profiler measures the device characteristics. Then, it keeps monitoring the program and network characteristics during the whole execution time because these characteristics can often change and any old or inaccurate measurement may lead MAUI to make the wrong decision. The offloading decision is taken at runtime. The framework chooses which components should be remotely executed according to the decision of the MAUI solver. The decision is based upon the input of the MAUI profiler.

3. Cloudlet

Offloading to the cloud is not always a solution, because of the high WAN latencies, mainly for applications with real-time restrictions. Thus the cloud has to be moved closer to the mobile user in the form of cloudlets VM based cloudlet framework. A cloudlet can be defined as a hosting environment for offloaded tasks that is deployed to remote resources, as different as individual servers or parallel systems. Cloudlets are virtual-machine (VM) based on support scalability, mobility, and elasticity. They are located in single-hop nearness to mobile devices. In the preparation step, the framework requires the cloning of the mobile device application processing environment to a remote host. It offloads the entire application using VM as the offloading mechanism and more precisely it uses a technique called dynamic VM synthesis. The VM would encapsulate and separate the guest software from the cloudlet's host software. The mobile device serves as a thin client providing only the user interface, whereas the actual application processing is performed on the cloudlet infrastructure. Device mobility is the main critical issue for mobile users on the move while connected to cloudlets.

4. Jade

Sharing the same concern but from a different perspective, Qian et al. present a system that monitors application and device status and that automatically decides where the code should be executed. The goal of Jade was to maximize the benefits of energy-aware computation offloading for mobile applications while minimizing the burden on developers to build such an application. During partitioning, applications are partitioned at the class level in Jade based on the collected information. As a preparation, the system checks the application and device status by monitoring the communication costs, work load variation, and energy status. The framework provides a sophisticated programming model with a full set of APIs, so developers have total control on: how the application is partitioned, and how remote code interacts with local code. The offloading decision is taken at runtime to decide where the code should be executed. Jade supports two types of servers: (1) Android servers and (2) Non-Android servers running operating systems such as Windows and Linux. Non-Android servers must have Java installed in order to support Jade. Jade's runtime engine runs as a Java program on a non-Android server.

3. GENERAL ISSUES AND CHALLENGES IN COMPUTATION OFFLOADING FOR MCC

The selected issues are presented from three perspectives: the resource-intensive structures of the existing frameworks, the security perspective, and the optimal offloading platform.

1. Platform diversity

One of the challenges in the current computation offloading frameworks is the diversity and heterogeneity of smartphone architectures and operating systems. This diversity is seen in the following example: MAUI is an offloading framework which is applicable for the .Net framework whereas Mirror Server is a framework which is compatible with the Android platform. A consistent access to cloud services is expected wherein SMDs are enabled

to access cloud computing services regardless of the installed operating system or the used hardware. A standardized offloading framework for different smartphone platforms is still a challenging issue in the MCC field.

2. Security and privacy in mobile cloud applications

Security of data transmission is an important concern in cloud based application processing. Security and privacy are two crucial concepts that need to be maintained during the offloading process. These concepts can be addressed from different angles: (1) Mobile device, (2) cloud data centers, and (3) during data transmission over the network. Besides all the technologies, there is a great increase in the variety of sophisticated attacks on mobile phones which are the main targets for attackers. Regarding the security in the cloud data centers, threats are basically related to the transmission of data between the different nodes over the network. Thus, high levels of security are expected by both the mobile clients and the cloud providers. In the current frameworks [10,12], binary transfer of the application code at runtime is continually subjected to security threats. Despite the available solutions, strong measures and a secure environment are required for the three entities of MCC model.

3. Fault-tolerance and continuous connectivity

In MCC, mobility is one of the most important attributes of SMDs. This is because freedom of movement and autonomy of communication during the consumption of mobile cloud services, are crucial criteria for users' satisfaction. However, there are some constraints that prevent the achievement of seamless connectivity and uninterrupted access to cloud services while on the move. As mobile users move, data exchange rates and network bandwidth may vary. Moreover, users may lose their connection while sending or receiving data; therefore, offloading approaches should be provided with suitable fault-tolerant strategies in order to resend the lost components, minimize the response time, and reduce the energy consumption of mobile devices. It should be noted that the guarantee of a successful execution of offloaded applications is very crucial for mobile users.

4. Automatic mechanism

The available computation offloading frameworks still need to be automated. This will help the offloading process to be performed in a seamless fashion while discovering the surrounded environment. The achievement of such automation is not an easy task as it needs the implementation of a protocol dedicated to finding and discovering services depending on the current context and its constraints.

5. Offloading economy/cost

Using cloud infrastructure resources imposes financial charges on the end-users, who are required to pay according to the Service Level Agreement (SLA) agreed on with the cloud vendor serving them. Generally, the operations of content offloading and data transfer between cloud providers incur additional costs on end-users. Therefore, economic factors should be taken into consideration while making the offloading decisions.

6. Partition offloading and external data input

At runtime, it is challenging to decide which application components need to be offloaded and to find the suitable server for that. Algorithms answering this problem need resource intensive effort, which can affect the execution time of the offloaded partitions of the application. Although existing application partitioning algorithms allow an adaptive execution of the application between the mobile devices and the cloud servers, they still do not provide any solution on how to utilize and benefit from the elastic resources in the clouds. This is specifically needed in order to make the applications scalable when a large number of mobile users need to be served and when the application requires input data that are stored in other remote servers.

4. CONCLUSION

This paper discusses three main concepts: (1) cloud computing, (2) mobile cloud computing, and (3) computation offloading. More specifically, it presents existing frameworks for computation offloading along with the various techniques used to enhance the capabilities of smartphone devices based on the available cloud resources. The paper investigates the different issues in current offloading frameworks and highlights challenges that still obstruct these frameworks in MCC. Moreover, the paper shows the different approaches that are used by the frameworks to

achieve offloading. Some of these approaches use static offloading while others employ dynamic offloading. Even though there exist a variety of approaches, all of them target the same objective which is the improvement of the smartphone device capabilities by saving energy, reducing response time, or minimizing the execution cost. We notice that current offloading frameworks are still facing some challenges and difficulties. For instance, lack of standard architectures. This shortage leads to more complications while developing and managing a proposed framework. Finally, it is important to come up with a lightweight paradigm or model that will help to overcome the difficulties and minimizing efforts while developing, deploying, and managing an offloading framework. We believe that exploring other alternatives, such as introducing a middleware based architecture using an optimizing offloading algorithm, could help better the available frameworks and provide more efficient and more flexible solutions to the MCC users.

5. REFERENCES

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.
2. R. Barga, D. Gannon, D. Reed, The client and the cloud: democratizing research computing, IEEE Internet Comput. 15 (1) (2011) 72.
3. B. Butler, Gartner: Cloud Putting Crimp in Traditional Software, Hardware Sales, Networkworld, 2012.
4. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generat. Comput. Syst. 25 (6) (2009) 599–616.
5. W.-C. Chuang, B. Sang, S. Yoo, R. Gu, M. Kulkarni, C. Killian, Eventwave: programming model and runtime support for tightly-coupled elastic cloud applications, in: Proceedings of the 4th annual Symposium on Cloud Computing, ACM, 2013, p. 21.
6. B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, CloneCloud: elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, ACM, 2011, pp. 301–314.
7. B.-G. Chun, P. Maniatis, Augmented smartphone applications through clone cloud execution, HotOS 9 (2009) 8–11.
8. E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, Maui: making smartphones last longer with code offload, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, ACM, 2010, pp. 49–62.
9. Y. Cui, X. Ma, H. Wang, I. Stojmenovic, J. Liu, A survey of energy efficient wireless transmission and modeling in mobile cloud computing, Mobile Networks and Applications 18 (1) (2013) 148–155.
10. J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
11. H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Commun. Mobile Comput. 13 (18) (2013) 1587–1611.
12. A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, V.H. Tuulos, Misco: a mapreduce framework for mobile systems, in: Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments, ACM, 2010, p. 32.
13. I. Giurgiu, O. Riva, G. Alonso, Dynamic software deployment from clouds to mobile devices, in: Middleware 2012, ACM, 2012, pp. 394–414.
14. M.S. Gordon, D.A. Jamshidi, S. Mahlke, Z.M. Mao, X. Chen, Comet: Code offload by migrating execution transparently, in: Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 2012, pp. 93–106.
15. K. Ha, G. Lewis, S. Simanta, M. Satyanarayanan, Cloud Offload in Hostile Environments, Technical Report, DTIC Document, 2011.
16. S. Hakak, S.A. Latif, G. Amin, A review on mobile cloud computing and issues in it, Int. J. Comput. Appl. 75 (11) (2013).
17. J. Hauswald, T. Manville, Q. Zheng, R. Dreslinski, C. Chakrabarti, T. Mudge, A hybrid approach to offloading mobile image classification, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE, IEEE, 2014, pp. 8375–8379.