Computation Offloading for Mobile Cloud Computing Frameworks and Techniques

Hesham Abusaimeh

Middle East University, Amman, 11831 Jordan

***Abstract*- With mobile devices' limited resources, computing the increasing amount of data locally puts strain on their batteries, processors, storage and bandwidth making it necessary to offload compute-heavy tasks to a close proximity edge cloud server. A research field known as Mobile Cloud Computing (MCC) has emerged to address the shortcomings of mobile devices by means of offloading. However, cost of operation and maintenance of cloud servers alongside the mobile nature of devices that could potentially cause connectivity issues are some of the most prominent challenges that face an efficient offloading process.**

***Keywords* - framework, mobile cloud computing, mobile edge computing, offloading.**

1. Introduction

Offloading is the process of moving parts or entirety of mobile applications to the cloud for accelerated computation as shown in Figure 1[1].

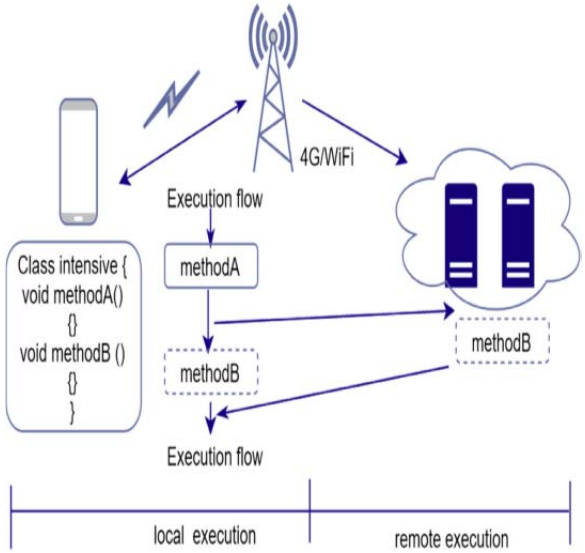

Figure 1. Offloading process

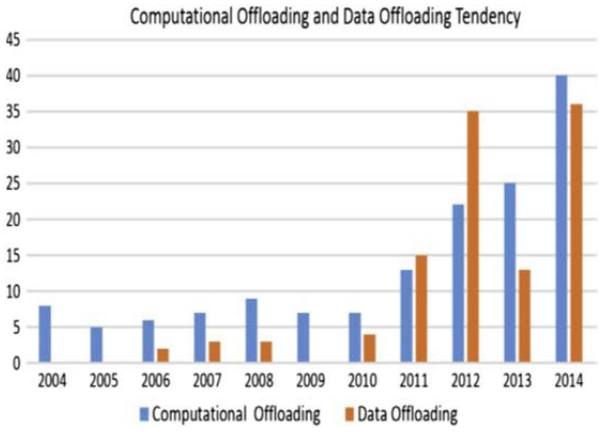

Figure 2. Comparison between data offloading and Computational offloading

As shown in Figure 2, there are tremendous increments in the computation offloading in the tenth year from 2004 to 2014 and it should be doubled now. Therefore, specialized servers (typically running a virtual machine) handle the requests from users running applications that interface with the server using frameworks, which make the decision of offloading based of different criteria to reduce power consumption and/or improve performance [2].

For an efficient offloading process, factors like security, performance, bandwidth and cost should be factored in the implementation of frameworks and

DOI: 10.18421/TEM113-08
https://doi.org/10.18421/TEM113-08

Corresponding author: Hesham Abusaimeh,
Middle East University, Amman, 11831 Jordan.
Email: habusaimeh@meu.edu.jo

Received: 15 April 2022.
Revised: 22 July 2022.
Accepted: 28 July 2022.
Published: 29 August 2022.

the cloud. Other challenges include a lack of a standardized offloading framework, cross-platform compatibility, contrasting network conditions, latency and operation costs like bandwidth and storage [3].

2. Methodology

In this paper, we employ a body of related work in order to highlight scenarios where offloading is beneficial to the mobile users.

We also cover scenarios where it is better for applications to run locally on mobile devices.

Finally, we present our own recommendations for better practices and implementation methods as a guideline for future research.

3. Related Work

We survey recently published papers in an effort to shed more light on the importance of load balancing, proper decision making, privacy and security in offloading for variety of mobile, embedded and cloud computing applications. We examine several approaches and implementations of MCC such as smartphones and vehicles.

Peng, et al. (2019): This paper attempts to find an optimal method for balancing energy and cost savings by investigating multi-objective offloading approaches for workflow applications in mobile edge computing (MEC). MCC is typically located remotely to the mobile device, which could cause increased network latency. MEC is closer to the user and would be better suited for applications that require low latency and are usually deployed as edge server cloudlets [4].

Our takes: We believe that the algorithm that they proposed for offloading named MCOWA could be beneficial for workflow application which should be executed quickly. However, MCOWA does not thoroughly address multi-objective scenarios where execution is distributed among multiple clouds or cloudlets.

Xu, et al. (2019): An important application of edge computing is the Internet of Connected Vehicles (IoV) where vehicles offload data to an Edge Computing Device (ECD) to process data such as traffic data. This paper proposes an efficient and secure method named ECO. Majority of vehicles lack the processing power and the storage capacity needed requiring data to be sent to a nearby Mobile Edge Computing (MEC) server through roadside units (RSUs) using the vehicle-to-infrastructure (V2I) model [5].

Our takes: We believe in the emerging field of IoV and how it could improve transportation. We commend the authors' take on offloading in this special offloading scenario by introducing their ECO algorithm. However, the lack of established ecosystem and unreliable connection could prove difficult to implement practical offloading for IoV.

Akherfi, Khadija, Micheal Gerndt, and Hamid Harroud (2018): This paper compares existing frameworks of offloading while highlighting the challenges they may face in properly offloading data to the cloud. It also the raises the concern for the lack of a standardized offloading paradigm and suggests a middleware to make it easier to implement frameworks. Frameworks typically partition an application to offload the parts that are computationally demanding and they rely on two main methods, either by requesting a Remote Procedure Call (RPC), or by mirroring to a virtual machine (VM) where execution is paused on the mobile device.

This paper compares the following frameworks:

- CloneCloud
  This framework relies on partitioning offloadable segments of the software, while accounting for segments that rely on a mobile device's features such as sensors which require to be processed locally. The framework works by mirroring an image of the software to a VM on a cloud server and during runtime, threads are paused on the mobile device while the server computes the data and then sends it back to the device so the operation resumes locally. CloneCloud utilizes dynamic offloading.
- MAUI
  This framework prioritizes energy saving when offloading and relies on the developers of the applications to specify which parts of their applications should be offloaded and which parts should be run locally.
- Cloudlet
  For applications with low-latency requirements, offloading to a remote cloud server is not feasible. The concept of cloudlets is to put low proximity, single-hop edge servers next to the mobile devices to reduce latency. Cloudlets should be distributed on several areas and make themselves discoverable to the clients for offloading. The framework would be mirroring the application to a dynamic VM that would be sandboxing users for enhanced security and privacy.
- Jade
  This framework aimed to make it easy for developers to implement offloading by utilizing Java Runtime Engine (JRE) which made it a cross-platform framework. Jade decides whether or not to offload at runtime and supports Android servers and other operating systems.

- Mirror server
  Micro servers utilize Telecommunication Service Provider (TSP). Other than offloading, this framework also provides storage, security and is compatible with several mobile platforms. It is also unnecessary to partition applications as it is capable of offloading them entirely. However, mirror servers are computationally limited and are only capable of providing a limited set of services.
- Cuckoo
  This framework uses Java and offloads data to servers that run a Java Virtual Machine (JVM) and runs dynamically. Developers need to specify which portions of their applications run locally and which components get offloaded.
- Phone2Cloud
  This framework prioritizes both energy savings and performance improvement. Developers are required to write their applications with cloud in mind to utilize Phone2Cloud. The framework allows either part of the application or its entirety to be offloaded for computing [6].

A primary challenge that faces offloading is the limited support of platforms. For example, MAUI supports applications built in .NET which targets Windows, while Mirror Server targets Android. However, cross-platform frameworks should accelerate adaption of offloading.

Bajpai, Abhishek, and Shivangi Nigam (2017): Compare the additional following frameworks:

- MACS: The Mobile Augmented Cloud System (MACS) does partitioning at runtime and decides whether or not to offload to the cloud. The framework attempts to be efficient at offloading by reducing memory usage, keeping energy consumption minimal and reducing execution time.
- AHP and TOPSIS: Mobile devices could choose from many clouds offering similar offloading solutions using the analytical hierarchy process (AHP) which factors many parameters and The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) which is used to decide on which cloud to use.
- Energy Aware Design for Workflows: In this method, mobile devices decide on which cloudlet should be considered for offloading based on real-time data. If more than one task is being transferred to the cloud, saving energy could be achieved by reusing the same route. Users could choose whether or not to offload based on privacy requirements.
- MCSOS: The mobile cloud with smart offloading system (MCSOS) framework relies on edge cloudlets for offloading. To improve performance, the mobile device offloads the task to a server which then splits the process among multiple servers to distribute the task and enhance performance.
- Secure Offloading: This framework prioritizes security which is a primary concern for offloading. It secures offloading on both the mobile device and the cloud to secure the communication link [7].

We agree that offloading should be hidden from mobile users. However, mobile systems that are capable of providing offloading tasks in a specific area could be a security and a privacy issue.

Enzai, Nur Idawati Md, and Maolin Tang (2016): This paper focuses on multi-site offloading implementation where servers are located in different places and presents a model.

Multiple servers could be used for offloading in order to improve performance. However, most offloading methods only account for a single cloud server. Moreover, current multi-server methods do not account for power, execution time and cost simultaneously [8].

We believe that multi-site offloading represents a more realistic offloading scenario due to the fact that a lot of users might be accessing the same service requiring scalability and this paper presents a heuristic algorithm to help address that. However, this paper didn't address the scalability of its proposed algorithm.

Jeevitha, N., and Dr G. Kesavaraj (2016) [9]: Due to the dynamic nature of the mobile environment, unreliable connection could lead to data loss which increases the complexity in making correct offloading decisions which could lead to the current approaches failing to switch context reliably. The cloud needs to be efficient and should be able to handle a large number of users accessing its resources at once for efficient offloading while factoring:

Workflow: It is recommended that an offloading method should be able to cover a side set of services depending on the needs of mobile users and execute computation tasks in parallel.

Mobility: Mobile environments continuously change which means that coverage and thus bandwidth may vary drastically.

Fault-tolerance: Unreliable connections should be expected when attempting to create and energy efficient offloading method [9].

We believe this paper sheds light on important offloading factors and provides guidelines for an efficient offloading designs and implementations. However, relying on the mobile user for the offloading is not always a good approach due to the varying technical knowledge of users.

Enzai, Nur Idawati Md, and Maolin Tang (2014) [10]: This paper offers best practices for implementing offloading while highlighting the issues and challenges it may face. Primary objectives for cloud offloading are power saving and to improve the execution speed. However, cloud execution is not always beneficial due to processing requiring more bandwidth, which could cause prolonged execution duration and reduce battery life of mobile devices, Offloading methods involve using a cloud server that matches the criteria to speed up the process based on either saving energy or reducing execution time. Sharing could also be adopted to utilize cases where same components that are requested by multiple users are using applications that share same components, which could be distributed and shared between them and data mining algorithms should be able to decide if sharing the same components across multiple users could be achieved. Security and privacy are main challenges for offloading and should be considered when more than one mobile user is accessing the same components. Also, Wide Area Network (WAN) latency and cloud servers' operation costs are considerations for efficient offloading [10].

We believe that resource sharing among cloud servers should improve efficiency as users usually require the same components. However, security and privacy should be addressed for safe offloading.

Kumar, et al. (2013) [11]: This paper examines the methods of offloading by surveying multiple algorithms that enable offloading to achieve better performance and to save energy. At the beginning of the millennia, the goal has shifted from making offloading possible to making contextual decisions whether offloading provides energy savings and/or performance gains. Algorithms decide whether it should offload or run the process locally based on several factors including network condition and available server performance. Offloading requires servers with high processing power. Also, it is preferred to utilize virtualization to ensure security by means of sandboxing. Several methods could be used to offload data at either class or object level such as Java Remote Method Invocation (RMI), .NET remoting, and Remote Procedure Calls (RPC). Other methods could be used for offloading on a virtual machine (VM) that allows elastic resources to be offloaded to multiple physical servers [11].

While we agree that there will be growth in the amount of data that require processing, modern mobile devices are now more capable of handling some tasks locally that would have benefited from offloading a decade ago.

Kumar, Karthik, and Yung-Hsiang Lu (2010) [12]: This paper addresses the power savings that could be achieved from offloading mobile applications and warns of the importance of evaluating the overhead of energy to maximize savings while retaining the security and privacy of data while in transmission. Offloading data and the cloud may raise users' concern for privacy and security since data is stored in remote servers, security and privacy policies vary depending on the cloud service providers. Reliability is another concern for mobile users as offloading and availability of content and service depends on the quality of the connection which could lead to users not being able to access content in crowded areas or locations such as basements or subways where network coverage could be weak. Also, reliance on cloud services could lead to setbacks during outages or lack of reliable connection. Real-time applications may benefit more by computing data locally on the mobile device and balancing computation between the mobile device and the cloud in order to reduce energy usage [12].

We believe that offloading does provide energy savings. However, tasks like image processing offloading may raise privacy concerns and we believe that this task should be processed locally instead despite the benefits of offloading.

Kemp, et al., 2010 [13]: A practical implementation of offloading on Android mobile operating system is Cuckoo, a framework which makes it easy to implement network offloading that can be run dynamically to decide which portion of an application should be exported and which portion should be run locally. This paper examines the framework using live examples of applications. Android applications are written in Java, often in development environments such as Eclipse to streamline the development process. Cuckoo attempts to make it easier to implement offloading to save energy and improve performance on Android devices [13].

Our takes: The Cuckoo framework is easy to implement since it has integration for Eclipse, a popular Android development environment. However, it only supports Android making it difficult to develop cross-platform applications that rely on offloading.

4. Conclusion

Mobile cloud computing is now more accessible to application developers and mobile users alike, thanks to a wide selection of frameworks and offloading methods. In this paper, we surveyed a body of work related to the studying of mobile cloud computing, what benefits it provides to mobile users and how it could be enabled using frameworks. We covered use cases where it is beneficial, where it is not and highlighted the challenges facing implementing it efficiently like security, privacy, cost and performance.

However, better partitioning algorithms and more advanced communication technologies like the emerging 5G technology should allow faster and more reliable offloading due to higher bandwidth.

Finally, the following practices as a reference for future research are recommended:

- Cross-platform compatibility: Due to the wide variety of operating systems, offloading frameworks are expected to run on as many mobile devices as possible so cross-platform compatibility is crucial to ease the implementation process.
- Open source frameworks: Open source software allows collaborative development which should accelerate the implementation of offloading and should help in improving and auditing security while addressing privacy concerns.

References

[1]. Boukerche, A., Guan, S., & Grande, R. E. D. (2019). Sustainable offloading in mobile cloud computing: algorithmic design and implementation. *ACM Computing Surveys (CSUR)*, *52*(1), 1-37.

[2]. McNett, M., Gupta, D., Vahdat, A., & Voelker, G. M. (2007, November). Usher: An Extensible Framework for Managing Clusters of Virtual Machines. In *LISA* (Vol. 7, pp. 1-15).

[3]. Cox, J. H., Chung, J., Donovan, S., Ivey, J., Clark, R. J., Riley, G., & Owen, H. L. (2017). Advancing software-defined networks: A survey. *IEEE Access*, *5*, 25487-25526.

[4]. Peng, K., Zhu, M., Zhang, Y., Liu, L., Zhang, J., Leung, V., & Zheng, L. (2019). An energy-and cost-aware computation offloading method for workflow applications in mobile edge computing. *EURASIP Journal on Wireless Communications and Networking*, *2019*(1), 1-15.

[5]. Xu, X., Xue, Y., Qi, L., Yuan, Y., Zhang, X., Umer, T., & Wan, S. (2019). An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles. *Future Generation Computer Systems*, *96*, 89-100.

[6]. Akherfi, K., Gerndt, M., & Harroud, H. (2018). Mobile cloud computing for computation offloading: Issues and challenges. *Applied computing and informatics*, *14*(1), 1-16.

[7]. Bajpai, A., & Nigam, S. (2017). A study on the techniques of computational offloading from mobile devices to cloud. *Advances in Computational Sciences and Technology*, *10*(7), 2037-2060.

[8]. Enzai, N. I. M., & Tang, M. (2016). A heuristic algorithm for multi-site computation offloading in mobile cloud computing. *Procedia Computer Science*, *80*, 1232-1241.

[9]. Abusaimeh, H. (2020). Virtual Machine Escape in Cloud Computing Services. *International Journal of Advanced Computer Science and Applications*, *11*(7).

[10]. Enzai, N. I. M., & Tang, M. (2014, April). A taxonomy of computation offloading in mobile cloud computing. In *2014 2nd IEEE international conference on mobile cloud computing, services, and engineering* (pp. 19-28). IEEE.

[11]. Kumar, K., Liu, J., Lu, Y. H., & Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mobile networks and Applications*, *18*(1), 129-140.

[12]. Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy?. *Computer*, *43*(4), 51-56.

[13]. Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2010, October). Cuckoo: a computation offloading framework for smartphones. In *International Conference on Mobile Computing, Applications, and Services* (pp. 59-79). Springer, Berlin, Heidelberg.