International Journal of Cloud Applications and Computing
Volume 15 • Issue 1 • January-December 2025

Mobile Cloud Computing:
A Comparison Study of Cuckoo and Aiolos Offloading Frameworks

Sanjay P. Ahuja
https://orcid.org/0009-0002-4314-8059
University of North Florida, USA

Inan Kaddour
University of North Florida, USA

ABSTRACT

Currently, smart mobile devices are used for more than just calling and texting. They can run complex applications such as GPS, antivirus, and photo editor applications. Smart devices today offer mobility, flexibility, and portability, but they have limited resources and a relatively weak battery. As companies began creating mobile resource-hungry and power-hungry applications, they have realized that cloud computing was one of the solutions that they could utilize to overcome smart device constraints. Cloud computing helps decrease memory usage and improve battery life. Mobile cloud computing is the current and expanding research area focusing on methods that allow smart mobile devices to take full advantage of cloud computing. Code offloading is one of the techniques that is employed in cloud computing with mobile devices. This research compares two dynamic offloading frameworks to determine which one is better in terms of execution time and battery life improvement. While executing light tasks Cuckoo does better with local execution while Aiolos outperforms Cuckoo when offloading a light computation task to the cloud. Similarly, Aiolos performs better than Cuckoo when offloading a heavy computation task to an EC2 instance. Regarding battery consumption, offloading using either framework saves 23% more power than the local environment. Aiolos consumes less battery power than Cuckoo when offloading a heavy computation task.

KEYWORDS

AIDL, Ailos, Cuckoo, Dynamic Offloading, EC2, Offloading, OSGi, Performance Comparison

1. INTRODUCTION

Offloading, also called augmented execution, is the method of sending a resource intensive task to a remote server; an old technique that has being rediscovered to reduce power consumption and speed up computation tasks. Since the beginning of mobile computing in the early 1990s, the resource poverty of mobile devices has been identified as a major constraint. According to Mahadev in (Mahadev, 1993), "Mobile elements are resource-poor relative to static elements. Regardless of future technological advances, a mobile unit's weight, power, size, and ergonomics will always render it less computationally capable than its static counterpart. While mobile elements will undoubtedly improve in absolute ability, they will always be at a relative disadvantage".

DOI: 10.4018/IJCAC.378695

The data shown in Figure 1 illustrates that the previous statement remains correct even with technological advances in mobile devices, as their resources remain limited when compared to a typical server.

Figure 1. Hardware comparison between servers and mobile devices

FIGURE 1: Evolution of Hardware Performance

	Typical Server		Typical Handheld or Wearable	
Year	**Processor**	**Speed**	**Device**	**Speed**
1997	Pentium® II	266 MHz	Palm Pilot	16 MHz
2002	Itanium®	1 GHz	Blackberry 5810	133 MHz
2007	Intel® CoreTM	9.6 GHz 2 (4 cores)	Apple iPhone	412 MHz
2011	Intel® Xeon®	32 GHz X5 (2x6 cores)	Samsung Galaxy S2	2.4 GHz (2 cores)
2013	Intel® Xeon®	64 GHz E5 (2x12 cores)	Samsung Galaxy S4	6.4 GHz (4 cores)
			Google Glass OMAP 4430	2.4 GHz (2 cores)

In 1997, to improve execution time, offloading was first introduced in mobile computing by Noble et al. (Noble, 1997) in the Janus speech recognition application. The application was modified to operate in three modes in Odyssey. The first mode was local execution of the application, the second mode was remote execution of the application, and the last mode was a hybrid, where one part of the speech processing application was executed locally, and the other part was executed on the server. Odyssey had the ability to dynamically decide the optimal execution mode based on many factors such as network bandwidth. A few months later, Flinn demonstrated that remote execution could save battery energy (Flinn, 1999).

The appearance of cloud computing in 2008 addressed a very important question around offloading which was "where should remote execution take place?" The success of Apple's cloud-based Siri speech recognition service validates the use of clouds at commercial levels and opened a new era of cloud offloading.

Offloading is considered a better option than online applications for two main reasons (Dejan). The first is that users do not always have access to the internet. The second is that online applications cannot gain access to the phone's features such as camera or motion detection. There are two types of offloading: static and dynamic. Static offloading is when the tasks to be executed on the cloud are identified at compile time or runtime. Dynamic offloading is when an external resource manager determines whether to run a specific task locally or on a remote server to achieve better performance and longer battery life. There are two main offloading approaches. The first approach requires a framework on the top of the existing runtime system, for example MAUI, Cuckoo, ThinkAir, Aiolos, and MCM framework. The second approach requires a modification to the operating system or virtual machine on which the process is running. As a result, this modification makes it hard for this approach to achieve real life deployment, especially because there are some security concerns