Task Offloading Engine for Heterogeneous Mobile Clouds

Dawand Sulaiman
School of Computer Science
University of St Andrews
KY16 9SX, St Andrews, UK
djs21@st-andrews.ac.uk

Adam Barker
School of Computer Science
University of St Andrews
KY16 9SX, St Andrews, UK
adam.barker@st-andrews.ac.uk

ABSTRACT
The limitations in computational resources and battery power of mobile devices led to the concept of offloading compute-intensive tasks to powerful devices. We have developed a framework to offload tasks from a mobile device to other nearby heterogeneous devices. It contains an offloading engine to selectively choose the target devices for the execution of the offloaded tasks to address optimal scheduling across devices with diverse capabilities. Our initial conducted runtime measurements show the feasibility of this concept. As preliminary results, we show that offloading compute intensive tasks from a device with less computational capability to a set of nearby more powerful devices can reduce the overall computational time by approximately 50%.

CCS Concepts
• Computer systems organization → Peer-to-peer architectures; Human-centered computing → Ubiquitous and mobile computing design and evaluation methods.

Keywords
Mobile Clouds; Task Offloading; Heterogeneity.

1. INTRODUCTION
The concept of using cloud hosted components as a means to overcome the resource-constraints of mobile devices is known as Mobile Cloud Computing (MCC). The mobile devices can be leveraged with the computation and storage resources provided by the distant cloud servers such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure. However, as smartphones and tablets gain more CPU power and longer battery life, the meaning of MCC gradually changes. Instead of being fully dependent on the cloud, a number of nearby devices can be used to coordinate and distribute content and resources in a decentralized manner. The local mobile cloud or ad hoc mobile cloud is a research domain which investigates leveraging heterogeneous resources of mobile devices in the vicinity for the execution of compute-intensive tasks. It has been evident that a collection of mobile devices can be used to perform compute intensive tasks in a coordinated manner.

The advantages which local mobile clouds provide including network congestion, cellular data network saturation, and energy saving have made it an attractive research domain. The approach of offloading computation to nearby devices meets the need of the mobile devices with limited processing capabilities such as wearable devices. Our initial results show that offloading does not always reduce the makespan and it might lead to longer execution time and wastage of energy consumption. Only when the task is offloaded from a less computationally capable device, we gain the reduction in the task runtime and save energy.

2. RELATED WORK
The continuous advancement in the processing, memory, network, and battery power of mobile devices has attracted researchers to position local mobile clouds as a core component to the future of mobile computing. Authors at [1] envision mClouds which enables local data exchanges between the devices over a free high-bandwidth local networks. The computational resources of nearby devices are used to leverage mobile devices in the context of mobile crowd computing [2]. The device to device communication mechanism used is Wi-Fi Direct technology as described in [3]. Unintelligent workload distribution is one of the research problems in task scheduling and allocation among the heterogeneous devices in a mobile cloud [4].

3. OVERVIEW OF THE FRAMEWORK
We propose an experimental framework in the context of mobile ad hoc clouds. It is implemented on top of multi peer connectivity library [5] and is designed to work with both OSX and iOS applications. Figure (1) shows an overview diagram of our framework and its offloading engine. We describe the different components of the framework in the following subsections.

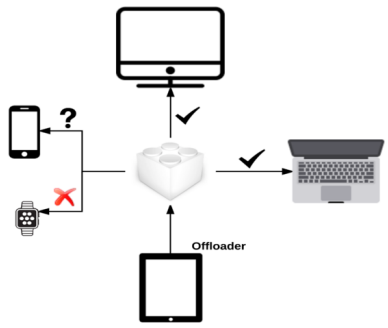

Figure 1 Framework overview and its task offloading engine

3.1 Device Discovery
The process of discovering nearby devices is achieved by the multi peer connectivity library. The library uses the concept of zero configuration network technology [6] which enables devices to advertise services and to discover what services other nearby devices on the local network are offering. A browser object in a host device searches for peers which have an advertiser object.

MobiCASE 2016, November 30-December 01, Cambridge, Great Britain
Copyright © 2016 EAI
DOI 10.4108/eai.30-11-2016.2267048

The OSX (MacBook, iMac) and iOS devices (iPhone, iPad, Watch) which are connected to the same local network can be discovered. Another discovery mechanism is based on Bluetooth. All our preliminary tests are based on infrastructural Wi-Fi but we are planning to extend our communication mechanisms to Bluetooth for the future tests and compare the performance of different discovery mechanisms.

3.2 Connection Establishment

The framework allows the devices to act both as a service discoverer or as a service advertiser. Once a device discovers another nearby device, an invitation is sent for establishing a connection between them. The invitation can be accepted or rejected by the receiver. After the connection is established, a session will be created and any further devices need to join that session. The process of sending an invitation and accepting it needs manual intervention from the user. We can automate this process and opportunistically transfer content and tasks from the devices by establishing a trust mechanism between the devices. For that we need to consider the security and privacy concerns and establish of a trust certificate between the devices before they can establish the connection. Lessons can be learned from [7].

3.3 Offloading Engine

After a device successfully joins a session, it will be checked whether it is a new device or a returning device. The devices are identified by a unique device ID stored in the framework. The framework needs to perform a CPU benchmark test on the new devices. The benchmarking test includes a Mandelbrot set which is executed four times and the average score is then stored in a file. The offloading engine uses that score to decide whether to offload tasks to nearby devices or to execute them locally as it is shown in Figure 1. The framework logs computational time of the running tasks for every device as well as the data transfer time.

Table 1. The devices used in our testbed

ID	Device Name	Processor	Clock Speed (GHz)	Benchmark Score
D1	iPhone 5	Apple A6	1.3	729
D2	iPhone 6	Apple A8	1.4	1536
D3	MacBook Air	Intel Core i7	1.8	2331
D4	MacBook Pro	Intel Core i7	2.5	3907

4. PRELIMINARY RESULTS

Using our framework, we have run a number of test results for an application scenario which is a string search on a large text file. The text file consists of 717,574 characters. This is a compute intesive task for a resource limited device. Boyer Moore string search algorithm is used for searching a keyword in the text file. The tests are performed both locally on the host device and on the nearby devices after the session is formed among them. The devices in Table 1 were used for our experiment. As it is mentioned in Section 3.3. the score of each device's CPU benchmark is recorded in the file. We store a unique ID for each device consisting of 16 digits but for simplicity of presentation of results we use D1, D2, D3, and D4 respectively.

We executed the string searching task in three different scenrios: locally on each device, offloading the task to all other devices, and offloading the task using our offloading engine to a set of devices. We can notice that only the overall runtime of D1 reduces in the second scenario which does not use any decision engine. For other devices, the offloading has increased the overall runtime of the task thus wasting computational resources and energy of the nearby devices. However, when the offloading engine is used, the task is only offloaded to the nearby devices which have a higher benchmark score. Figure 2 shows the results of the task execution in each scenario. It is worth noting that D3 did not benefit from the offloding engine. This is due to the the round trip time between D3 and D4 which increased the total makespan compared to local execution.

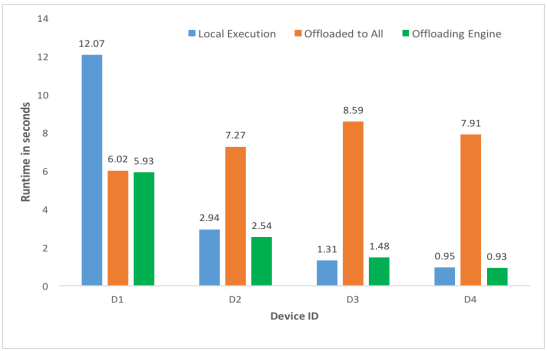

Figure 2 Comparison between local execution, offloading to all the nearby devices and our offloading engine

5. CONCLUSION AND FUTURE WORK

In this paper, we have shown that it is feasible to use an offloading engine to decide on the offloading compute intensive tasks from one device to other nearby devices and reduce the overall computational time of the offloaded tasks. Our future work includes offloading tasks to nearby devices and compare the results with offloading the same tasks to a nearby cloudlet and a distant cloud. We also want to setup an energy testbed to measure the battery usage for each offloading and test if the framework can help in saving the battery of the device as well. Generalizing the framework and enabling parameterization of the methods can help application developers write work sharing mobile applications.

6. REFERENCES

[1] Miluzzo, E., Cáceres, R. and Chen, Y.F., 2012, June. Vision: mClouds - computing on clouds of mobile devices. In Proceedings of the third ACM workshop on Mobile cloud computing and services (pp. 9-14). ACM.

[2] Fernando, N., Loke, S. W., and Rahayu W. 2016. Computing with Nearby Mobile Devices: A Work Sharing Algorithm for Mobile Edge-Clouds. In IEEE Transactions on Cloud Computing, vol.PP, no.99, pp.1-1.

[3] Camps-Mur, D., Garcia-Saavedra, A. and Serrano, P., 2013. Device-to-device communications with Wi-Fi Direct: overview and experimentation. IEEE wireless communications, 20(3), pp.96-104.

[4] Yaqoob, I., Ahmed, E., Gani, A., Mokhtar, S., Imran, M., and Guizani, S. 2016. Mobile ad hoc cloud: A survey. Wireless Communications Mobile Computing.

[5] Multipeer Connectivity Framework Reference. 2013. Retrieved September 10, 2016 from https://developer.apple.com/reference/multipeerconnectivity

[6] Steinberg, D.H. and Cheshire, S., 2005. Zero Configuration Networking: The Definitive Guide. O'Reilly Media, Inc.

[7] Lacuesta, R., Lloret, J., Sendra, S. and Peñalver, L., 2014. Spontaneous ad hoc mobile cloud computing network. The Scientific World Journal, 2014.