Indian Journal of Science and Technology, Vol 9(22), DOI: 10.17485/ijst/2016/v9i22/91978, June 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Offloading Computation for Efficient Mobile Cloud Computing

Hitika Atre, Karan Razdan and Raj Kumar Sagar

Amity University, Noida, Uttar Pradesh – 201313, India;
hitikaatre@gmail.com , karan.razdi@gmail.com, rksagar@amity.edu

Abstract

Background/Objectives: Mobile and cloud are two of the most used technologies today and it is only a matter of time before these technologies combine. In this study we try to combine these two. **Method/Statistical Analysis:** To study the combination of these two technologies, we offloaded a certain amount of computational work from a mobile device (An android device in our case) to a cloud server (AWS) and studied the real world performance benefits and battery gains that we achieved with it. **Findings:** We found a clear gain in terms of the load on the CPU of the device as well as the battery life consumption. This could lead to high real world gains in performance. **Applications/Improvements:** This technology can be used to create cloud-first mobile applications that not only just store their data on the cloud, but also rely on it for its computational needs.

Keywords: Android, AWS, Offloading, Mobile Cloud Computing, Power Consumption

1. Introduction

In its most basic form, a cloud offers various resources such as compute, storage, content delivery, analytics etc. over the internet. The user does not need to be physically in possession of these resources. The cloud 'offers' these resources to the end user via an Internet connection. The maintenance and the upkeep of the provisioned resources needs to be done by the cloud service provider and the user just 'consumes' these resources. People who do not have the capital to invest in their own infrastructure at an early stage can take advantage of this as it is mostly cost effective and takes away all the maintenance needs of the organization. Cloud computing has caught the imagination of millions and has been the IT buzz word for a while now. There are organizations that were 'born in the cloud itself and continue to operate in the same way. Even with its immense popularity, cloud resources are still majorly being accessed by desktop computers only. When it comes to mobile devices, they are still very much limited to their own hardware for all the compute tasks.

With the progression of wireless network services all over the world, smart phones have gained a strong user base. Just like cloud computing, mobile cloud computing aims at offloading the processing tasks and the storage tasks of the mobile device, to computational powerhouses with a large number of computers that performs all these tasks for the mobile devices[1]. The result of the computation is returned to the mobile device through the wireless network. This can lead to a large amount of savings for the device in terms of computational power, battery usage and storage space.

With more and more high-end phones coming out every year, the average consumer struggles to keep up with the pace. Applications become heavier and more taxing on the devices. The high-end devices are able to cope up with these ever increasing demands but the average phone struggles to keep up with the high computational needs of these applications. For such devices, a viable option is to offload their computational needs to a resource rich cloud server[2,3]. They can deliver the desired performance with close to no strain on the CPU. This

*Author for correspondence

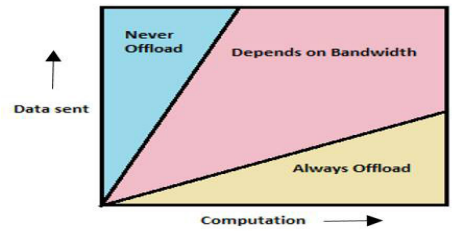

Figure 1. Suitable Conditions for Offloading.

gives more computational power to the system and the other user applications.

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. The ideal conditions for offloading a task involve a fairly large amount of computation being performed on the cloud server and the data that needs to be sent and received should be relatively small in size. In such a condition, the offloading of the computation should always be offloaded to the cloud server. On the other hand, if too much data needs to be sent to the cloud server to perform a relatively small computation, the offloading becomes counter-productive and should be avoided. When the conditions are a mix of the both, it depends on how good the internet conditions are. If a good network connection is available, the computation can still be offloaded. If not, the offloading may be avoided. This is well depicted by Figure 1[4]. We should try to be in the region that says "Always Offload" to be certain of our offloading being beneficial.

2. Related Work

In recent times, there have been a number of studies that try to tackle MCC. Some of them focus solely on improving the power consumption, a few of them focus on improving the overall throughput of the device and some focus on optimizing both of these at the same time. The maximum amount of gain on the processor load is around 75% and the on the power consumption is around 56%[5].

The efficiency of offloading computational tasks to a cloud depends upon several factors, most of which are dynamic in nature[6]. Due to the dynamic nature of these properties, the efficiency associated with the cloud setup also varies over time. Depending on the size of the task and the amount of bandwidth available, sending data to the cloud might not actually turn out to be beneficial with respect to the overall battery consumption. Kumar et. al.[4] studied this and came up with the following formula:

$$P_c \times \frac{C}{M} - P_i \times \frac{C}{S} - P_{tr} \times \frac{D}{B} \qquad (1)$$

Here,

- Pc is the power consumption by the mobile when executing instructions.
- Pi is the idle power consumption of the mobile.
- Ptr is the power consumption while transmitting data.
- C is the total number of instructions in the given computation.
- S is the time taken by the cloud server to process instructions.
- M is the time taken by the mobile to process the instructions.
- D is the total amount of data to be sent in bytes.
- B is the total available network bandwidth.

The first element in the formula calculates the total battery consumption if the computation has been completely done on the mobile device. The next two components calculate the power consumption if the whole computation is sent to the cloud server. The factor takes into account the power consumption involved in sending the data over the network. If the result of the above formula comes out to be greater than 0, we can say that the task of offloading the computation is actually beneficial and efficient. If the result is less than 0, the process turns out to be counter-productive. It can be clearly seen that due to these factors, not every computation can be efficiently offloaded at a given time.

$$\frac{C}{M} \times \left(P_C - \frac{P_i}{F}\right) - P_{tr} \times \frac{D}{B} \qquad (2)$$

If we consider that the speed of the cloud server is a multiple of the speed of the mobile processor, i.e., S=FxM. This means that the cloud server is F times faster than the mobile processor. Replacing that in the equation (1), we arrive at the equation (2) [4]. For the result to be highly positive, the value of *Pi/F* should be a very small one. That means the value of F should be really large. This straight away means that the speed of the cloud server should be several times greater than the mobile processor. Upon

solving the equation, we come at a realization that the bandwidth requirement is directly dependent upon the ratio of the terms D/C. From this, we can conclude that the offloading process is highly efficient when we have a large amount of computation that needs to be done and the data that needs to be transmitted for the computation is relatively small.[7] This scenario will always yield a positive result for the equation and hence make the offloading process efficient.

3. Methodology

We developed an Android application for the purpose of running some tests. The Android application was used to establish a connection with the Amazon EC2 instance for compute as well as the Amazon S3 bucket for storage needs. The application comprises of several modules. These modules include the following.

A. Statistics Module

The statistics module is the backbone of the application. It keeps on making a log entry of all the vitals of the device after every second. This continuous logging helps us come up with the results of executing the computational tasks on the device and the cloud. The statistics module gives us 4 important stats that are the current battery life, the CPU load, amount of free memory and the network usage of the device. There is a separate method for calculating each one of these stats. As we need to continuously keep monitoring these stats, the whole process is performed on a separate thread than the main thread. This helps in keeping this task running even when the main thread is busy with some other work[8].

B. SSH Module

The Secure Socket Shell is a protected way of connecting to a remotely located server with no physical access to it[9]. It encrypts the data and sends it through secure channels in order to maintain the privacy and the integrity of the data being sent. The connection is not natively supported by the android sdk. We need to use an external library called "Jsch" to be able to connect to the remote server. Jsch provides a robust way of connecting to the server by giving us a full range of options for authentication, including but not limited to, Public-Private Key authentication[10]. We use the Private Key in order to match it with the Public Key that is stored in the AWS servers. After the connection is established, commands can be sent to the instances through the secure channel.

C. S3 Download/Upload Module

We have a module that can upload the user data to the bucket and download files from their secure servers using the application. The application creates a secure channel to connect to the AWS servers and then starts transmitting the data[11]. The identity of the mobile device is verified by the Amazon Cognito Identity Provider. It creates a cognito pool id which is used together with the bucket name to authorize the user access.

D. Image Processing on the Phone

Image processing on the phone is done using the OpenCV library[12]. The library can be used to perform image manipulation using many of its built in functions. In particular, we perform the Canny Edge Detection technique. The OpenCV library has a built in method for canny edge detection. We just need to pass the image and the threshold values and the rest is done by the library.

E. Image Processing on the Cloud

A similar image manipulation process is performed on the EC2 instance using Octave GNU[13,14]. We use the octave CLI to execute programs. For image manipulation, the image package was installed and used to execute the built in function for canny edge detection. We send the command to the instance using the SSH connection that was established using Jsch. We securely send our command to the server and in a similar way receive the output from the instance.

The EC2 module is used to execute the instruction on the server and the same computation is performed on the device as well. As the most common piece of data that can be worked upon in a modern device is photos, we decided to go with an image processing application. We performed simple image processing on a number of images. The statistics modules logs all the statistics while both the functions are underway. These statistics are then collected as an average of several values.

4. Experiments

Our experiments were carried out on 2 devices, the first one being a first generation Motorola Moto G. The device

has a 1.2GHz Quad Core processor, 2070 mAh battery and Android 5.1.1. The other device was an OnePlus One. The device has a 2.5GHz Quad Core processor, 3100 mAh battery and Android 6.0. The first phone was considered as a modern day low-end device and the second phone was considered as a high-end device.

Both devices had a stable 3G connection with good signal strength running at all times. They had a fresh start before the tests were run. No background applications were running. A number of tests were carried out and the average values obtained from them were used. We provide the application with 3 types of loads, small, medium and large. Basically, the number of images being manipulated by the application increases in each load. So, the application has to do more work in order to finish the work that has to be done.

We monitored three factors while conducting the tests; the average battery consumption, the time taken for the computation and the change in the CPU Load. Once the average values were calculated, we plotted the graphs for each of the test cases.

For the first test, we look at the average time taken for the computation. The cloud server performs its computations on the cloud server as compared to executing the computation on the phone itself. It can be clearly seen in the graph that there is a notable difference in the time taken by both the platforms. As we can see in Figure 2, even though it is slight, the cloud is clearly seen to have an edge over its competitor, the mobile device. The difference between the two increases at a very slow rate and remains nearly constant for a large distribution of the load. This was seen for both, the high as well as the low-end device.

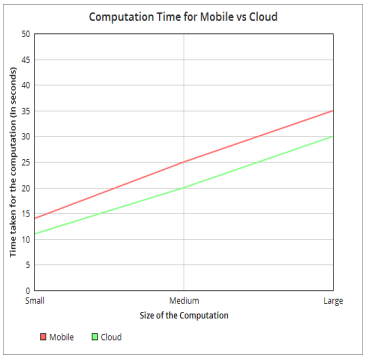

Figure 2. Computation time for mobile vs cloud.

For the next experiment, we took a look at the rate at which the application was consuming the battery life of the device. As the cloud implementation does not put any kind of load on the device, it was expected that the battery consumption would be much lower for it. At the same time, when we increase the amount of load on the application, the amount of battery it consumes also goes up. This is not the case for cloud as the battery consumption remains fairly linear in its case. From the results obtained in Figure 3, we could see that in case of a high end phone, due to the bigger screen, the application takes up a little more battery life which keeps on increasing when the amount of load is increased. Similar results are obtained in Figure 4 and hence we can see that both, the low and the high-end phones give a similar looking graph but at the same time, the high-end phone takes up more of the battery life.

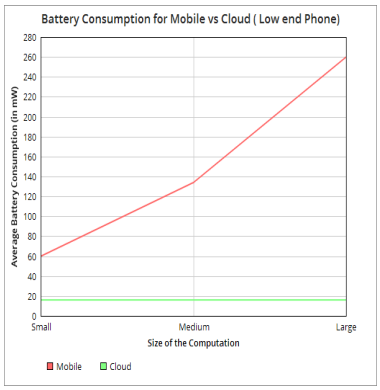

Figure 3. Battery Consumption for mobile vs. cloud (Low End Phone).

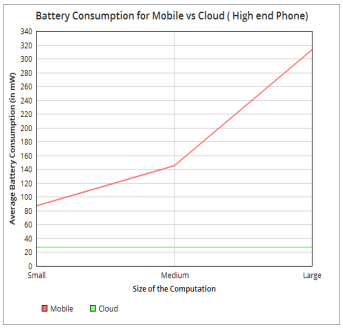

Figure 4. Battery Consumption for mobile vs. cloud (High End Phone).

For the last test, we look at the change in the CPU usage of the device once the computation starts to take place. Again, as the cloud implementation does not use much of the CPU horsepower, the expected increase in the load is not much. As compared to this, when we execute the code on the phone, it will obviously take up a lot of CPU horsepower and put the system under a certain load depending upon the complexity of the computation.

For a lower end phone, the computation is more taxing and puts the CPU under a higher load. This can be seen by the graph in Figure 6 where the CPU load increases at a rapid rate and takes up more than half of the available computational power. Similarly, we observe from Figure 5 that a high-end phone would be under load too but it would still be less than its low-end counterpart. The cloud implementation on the other hand sees close to no impact on the CPU usage as the only thing the application needs to do is to send a small amount of data to login to the instance and send it the command that needs to be executed.

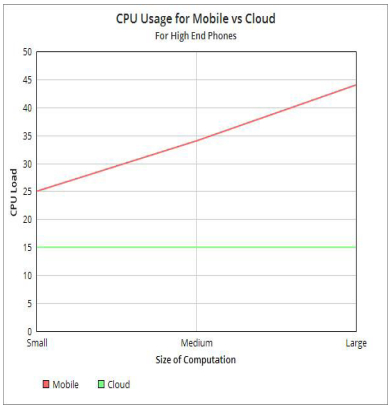

Figure 5. CPU Usage for mobile vs. cloud (High End phones).

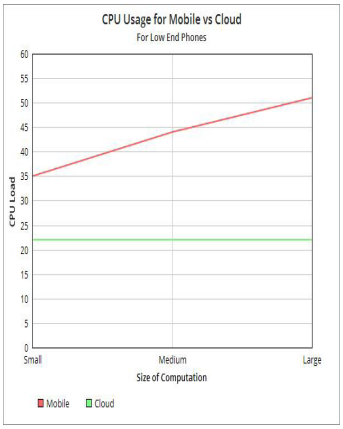

Figure 6. CPU Usage for mobile vs. cloud (Low End phones).

5. Conclusion and Future Scope

In this paper, we have tried to study the various benefits that can be attained using MCC, through a series of tests. We saw that in the case where a small amount of data transfer leads to a large amount of computation, it is always preferred to offload the computation. We tested this scenario and got conclusive evidence that MCC shows a huge advantage over its counterpart. It was clear that given the right circumstances, the process of offloading the tasks to the cloud can be beneficial for the mobile device, in terms of battery life as well as available computational power.

This clearly shows that mobile cloud computing is a highly viable option that can be widely used in the future. The benefits of using such a system are clearly visible and lead to actual real world performance as well as battery life benefits for the users[15,16].

For future work, A system can be built that would continuously monitor these stats and would make an informed decision based on the amount of data that needs to be offloaded. The system can tell us when it's good to offload the computation and when it's better to do it on the device itself. This can be implemented as a standalone application or system-wide layer within the operating system. The methods within an application that can be offloaded to the cloud can have a special label assigned to them. When such a method is called, it can check with the system if the conditions are viable for a cloud offload. The system would check the payload and the network bandwidth and accordingly make a decision. This kind of a system is most feasible on an android operating system due to the amount of freedom it offers to the developer. Other operating systems may not allow the same amount of flexibility.

6. References

1. Nielsen Informate Mobile Insights. Mumbai: Vserv: Smartphone User Persona Report. 2015.
2. Wang C. and Li Z. Parametric Analysis for Adaptive Computation Offloading. ACM SIGPLAN Notices. 2004; 39(6):119-30.
3. Wolski R et al. Using Bandwidth Data to Make Computation Offloading Decisions. Proc. IEEE Int'l Symp. Parallel and Distributed Processing (IPDPS 08). 2008; p. 1-8.
4. Kumar K and Lu YH. Cloud computing for mobile users: Can offloading computation save energy? IEEE Computer. 2010 April; 43(4):51–56.
5. Lai CF, Chao HC, Lai YX, Wan JF. Cloud-assisted real-time translating for http live streaming. In Proc. Wireless Communications, IEEE. 2013 June; 20(3).
6. Zhan K, Lung C and Srivastava P. A Green Analysis of Mobile Cloud Computing Applications. 2014; p. 6.

7. Ying-Dar Lin, Edward TH Chu, Yuan-Cheng Lai and Ting-Jun Huang. Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds. IEEE Systems Journal. 1932-8184 © 2013 IEEE.
8. Saad S and Nandedkar S. Energy Efficient Mobile Cloud Computing. International Journal of Computer Science and Information Technologies. 2014; 56.
9. What Is Secure Shell (SSH)? Available from: http://search-security.techtarget.com/definition/Secure-Shell
10. JCraft. JSch. Available from: http://www.jcraft.com/jsch/examples.
11. Murty J. Farnham: O'Reilly: Programming Amazon Web Services - S3, EC2, SQS, FPS, and SimpleDB. 2008.
12. OpenCV documentation index. (n.d.). Available from: http://docs.opencv.org.
13. Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T and Epema D. A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing. Cloud Computing. 2010; 34:115-31.
14. Eaton JW. Bristol, UK: Network Theory: GNU octave: A high-level language for numerical computations: Octave version 2.0.17 (stable). 2005.
15. Mohammad Saad S and Nandedkar S. Energy Efficient Mobile Cloud Computing. International Journal of Computer Science and Information Technologies. 2014; 56.
16. Kalpana V, Meena V. Study on Data Storage Correctness Methods in Mobile Cloud Computing. Indian Journal of Science and Technology. 2015 Mar; 8(6). Doi: 10.17485/ijst/2015/v8i6/70094.