IOSR Journal of Mobile Computing & Application (IOSR-JMCA)
e-ISSN: 2394-0050, P-ISSN: 2394-0042.Volume 2, Issue 2. (May. - Jun. 2015), PP 15-18
www.iosrjournals.org

Dynamic Framework Design for Offloading Mobile Applications to Cloud

V. Suganya[1], Dr. J.Dhillipan[2] D. B. Shanmugam [3]
[1] *MPhil., Research Scholar, Dept. of Comp.Sci, Dr. MGR. Chockalingam Arts College, Arni.*
[2]*Asst.Prof.,(S.G) & Head, MCA Department, SRM University, Ramapuram Campus, Chennai*
[3]*Asso.Prof., Dept. of Comp.Sci, Dr. MGR.Chockalingam Arts College, Arni.*

Abstract*: Mobile Cloud Computing (MCC) is an infrastructure where the data and the processing of data are outsourced. MCC integrates cloud computing into the mobile environment and executes the applications in the mobile device effectively by partitioning and offloading the computation intensive task to external resources (e.g. Public Clouds). The effective offloading is mainly focused on the decision maker which tells "when to offload" during execution time. Though prior decision making techniques has its own pros and cons, it doesn't support dynamic changing environment and also consumes more time and energy for training the input instances. Also while offloading there is no security for the information to be transmitted. So, the proposed dynamic framework is designed with efficient intelligent classifier for offloading mobile application in dynamically changing by estimating the applications on-device and on-server performance. And to ensure security, Steganography technique is additionally used within the proposed framework to conceal the information.*

Keywords:*Mobile cloud computing, Offloading, Security in Offloading, SOM Classifier, Dynamic Offloading.*

I. Introduction

The usage of smartphones and the mobile applications are rapidly increasing nowadays. Compared to desktop systems, the smartphones are less resource-constrained devices [pomac]. Therefore, the resource-intensive part of the mobile applications are fully or partially offloaded to the resourceful servers, such as clouds. Prior researches [4], [5], [6], [7], [8] has proposed various approaches for application partitioning and offloading the computation-intensive tasks to cloud servers or cloned virtual machines.

Implementing these approaches requires modification of applications or binary executables [6], [8] or some special compilation process [14]. On cloning the applications [13], though it doesn't require any modification in offloading the mobile application, the full image of the mobile app has to be uploaded to the cloud server. The deciding factors considered for offloading the mobile applications to cloud servers by existing researches is less efficient. For example, Young et al. [8] recommends offloading, when the data size for transmission is greater than 6 MB. MAUI [6] follows Linear regression model for offloading with certain hardware, software and connectivity features of both mobile handsets and server. But it gives 50% of wrong decisions, which leads to higher energy consumption and longer response time.

And also while offloading the data parameters, there is no secured transmission mechanism. So there occurs a larger possibility for the intruders to steal the information. Since the internet is an open arena for any online users, it is necessary to conceal the information during offloading process.

In this paper, we designed a dynamic framework for offloading resource constrained part of mobile applications to cloud servers, to address the above said offloading challenges. We design a transparent offloading mechanism through method interception at Dalvik Virtual Machine level to let the mobile application offload its resource constrained methods without modifying the application's source code, binary execution or any special compilation [pomac]. And also we ensured the privacy of offloading the sensitive data to the cloud servers are concealed by using two techniques such as Encryption of data and Steganography techniques [10].

The rest of the paper is organized as follows. After offloading impediment, the challenges in decision making, offloading mechanism, secure data transmission are discussed in section 2. In section 3 the dynamic framework design and its comparative results are discussed. Some preliminary results are received in section 4 and concluding remarks in section 5.

II. Offloading Impediments

To offload the mobile applications dynamically to the cloud, it is necessary to decide when to offload and how to offload securely. A decision maker can be used to anticipate the former part and an efficient offloading mechanism can be used for the later part. But existing techniques for the above said problem has some disadvantages.

2.1 Difficulties in Decision Making:

It is recommended to offload the mobile applications to cloud when the on-device execution time and energy is greater than the on-server's execution. In Y.W. Kwon et.al [8], computation is offloaded only when the method's parameter data size is greater than the threshold value (6 MB). But practically, it is difficult to have a common threshold value for all applications. These threshold based mechanisms will not work in dynamic environments where the resource information and bandwidth between the server and mobile device are changing dynamically. Therefore, without considering these factors, often the decision made by this policy is same.

In MAUI [6], a Linear regression model has been used as a classifier. It anticipates and compares the on-server execution by considering only the bandwidth or latency features. So, it may take wrong decisions in offloading. In POMAC [9], the Linear regression model and Support Vector Machine classifiers are compared for performance. The experiment result shows that a Linear regression model has 58.01% root relative absolute error rate and for Support Vector Machine (SVM), it is only 17.66% error rate.

In POMAC [9], a Multi-Layer Perceptron (MLP) has been used as a classifier. It dynamically collects the system environment and resource information such as bandwidth, latency, data size, CPU and memory availability for making the offloading decision. It also has a feedback channel for self-learning. But MLP consumes more time and energy for training the instances.

2.2 Difficulties in offloading mechanisms:

Though there are several offloading mechanisms available for offloading the computation intensive part of mobile applications to cloud, the offloading mechanisms are classified into two broad categories: a) Application partitioning and b) Virtual machine cloning.

In Application partitioning researches [6], [11], [12], [7], [8], the resource intensive portion is offloaded either switching the execution [11], [7] or invokes RPC [6], [8]. For these operation, it requires either annotation [6], binary modification [8] or compilation [14] to every application in the offloading framework. In VM based cloning, the smart phone's full clone image will be created and maintained in the cloud [4], [14] and [5]. While offloading, it suspends the smartphone's execution and transmit the execution to VM clone in the cloudlets. Also it requires proper synchronization of data (100 MB) between the clone and mobile device during transmission.

In POMAC [9], an efficient offloading mechanism and a transparent scheme has been proposed with no modifications for applications source code or binary executables or special compilations. But, there is no privacy concern on transferring the sensitive data to the server. Due to lack of this consideration for security, during applications offloading it opens door for hackers and malicious insiders to breach the data.

Therefore, it is necessary to determine a dynamic design framework to securely offload the mobile applications to the cloud server.

III. Classifier and DFDOMAC Design

In this section, selection of optimal classifier and design of DFDOMAC (Dynamic Framework design for Offloading Mobile Application to Cloud) are described.

3.1 Optimal Classifier and Crucial feature selection

To determine an optimal classifier for making offloading decision, several classifiers are compared for its accuracy and classification time. The various classifiers performance are verified with face recognition dataset of Picaso [2] and executed in Open source library Weka [3]. All the considered classifiers are examined for 60% to 70% training instances.

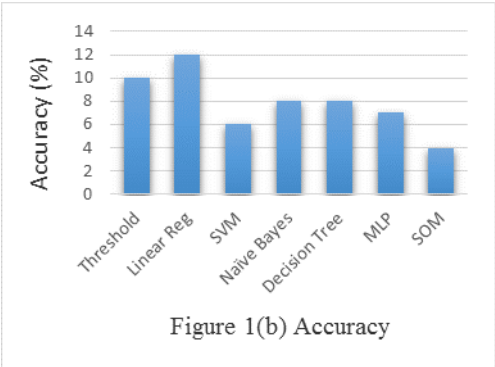

Figure 1(b) Accuracy

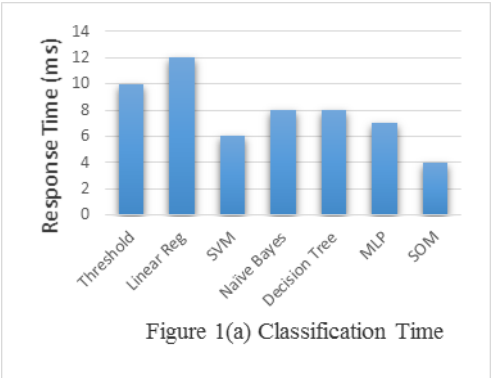

Figure 1(a) Classification Time

In Figure 1(a) and Figure 1(b), all the classifiers are compared for its classification time and accuracy. Figure 1(a) shows that the classification time of one instance remains same for all the classifiers, except multi-layer perceptron (MLP). Because MLP takes more time and energy to train, an optimal unsupervised efficient classifier called Self Organizing Map (SOM) has been proposed. The unsupervised SOM classifier equally performs well as the supervised MLP classifier in comparatively less time [15].

In figure 1(b), when comparing all the classifiers such as SVM, MLP for accuracy, the SOM outperforms for 100% of accuracy. But SOM requires more training data than the MLP and SVM [15]. Eventually, since Decision tree and SVM classifiers do not support online training, it is not recommended for optimal decision making process. Although SOM requires large percentage of data for training, it classifies the given instances in short span of time. So in our current design, we are recommending SOM as optimal decision maker for offloading.

In addition to the classifier, the offloaded method's vital features such as bandwidth, latency, data size, CPU and memory availability are collected by fProctor to impact the energy consumption and response time of an offloaded method.

3.2 Secure Offloading Mechanism design

Once the classifier makes the offloading decision, the offloading of computation intensive part will be performed transparently and securely.

DFDOMAC intercepts the method invocation at the Dalvik Virtual Machine instruction level. Figure 2 shows different modules of DFDOMAC; including Interceptor, fProctor, SOM Classifier, Encoder, Stego objects, mInterface, sInterface and Decoder.

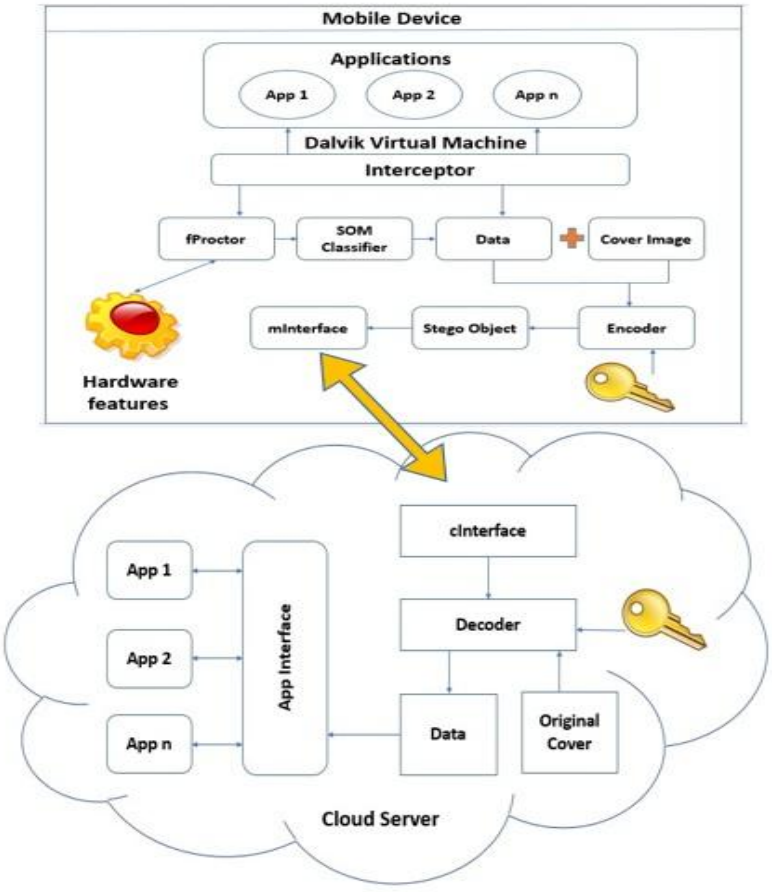

Fig 2. DFDOMAC Modules.

While intercepting, when the Dalvik interceptor reaches any method invoking instruction, it saves the current method's frame page and program counter to restart from the same point after returning from the invoked method. At this juncture, the interceptor intercepts the method call by getting all the input parameters field values, application name, method name and class name and packs them into a byte array.

After packing, mInterceptor sends the byte stream to the Encoder and suspends for the return value. Meanwhile, the vital features such as system environment and resource information are collected by fProctor dynamically. The bandwidth and latency are measured by sending small packets to the server and the mobile device. The /proc/stat and /proc/meminfo file in Android and server are used to get the CPU and memory availability. The invoked method's data size can be calculated from the input parameter.

The SOM classifier is the offloading decision maker of DFDOMAC design by accurately characterizing the relationship among different features. When a method is intercepted, the SOM classifier makes the decision either to offload that method or execute it locally by considering the feature values collected by fProctor and the calculation of the arguments size.

Once the SOM classifier decides to offload, the computation-intensive data will undergone for steganography and encryption process. In steganography the data is hided before sending them to the server and avoids unauthorized access [10]. Therefore, the computation-intensive data is hided with some cover image to improve to improve security, the data and cover image are encrypted by encoder using some random key by the mobile user and generates stego objects. Then the resultant stego objects [10] will be passed to the mInterface and suspends for the return value.

The mInterface communicates the server with the provided stego objects for execution and gets the result back. In the server side, the cInterface of the cloud server collects the stego objects. After that, the Decoder decrypts the data by providing receiver side random key and original cover image. Eventually the server side application interface collects and sends the decrypted parameter to the appropriate application's method for execution.

After the execution is completed, the DFDOMAC gets the result back from mInterface, decrypts and deserializes the result to build the appropriate object and returns to the invokes of the method[9]. Meanwhile, if any remote server failure, DFDOMAC continues with normal execution flow for that application.

IV. Conclusion and Future Amendments

In this paper, we examined two problems about offloading mobile application to clouds. Firstly we analyzed whether offloading can be conducted and secondly we addressed how to perform offloading task securely and transparently. Analogized to the existing classifiers, the static policy worsen the performance of the application. Also the dynamic natured classifiers such as MLP takes more time and energy for training the data. Thus, we have designed decision maker based on SOM, which performs equally well as MLP in less time but requires more amounts of training instances. And analogizing to the existing offloading mechanism, lack of security in offloading process. In this paper, we have designed offloading mechanism which doesn't requires any source code modification and ensures security by concealing the data with steganography and encryption. But for method offloading, it demands network communication from Dalvik VM, which in turn requires application to let the network communication for permission checking.

References

[1]. Power Tutor. www.powertutor.org/.
[2]. Picaso. http://code.google.com/p/picaso-eigenfaces/.
[3]. Weka. http://www.cs.waikato.ac.nz/ml/weka/.
[4]. B.G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: Elastic execution between mobile device and cloud. In Proc. of EuroSys, pages 301–314, 2011.
[5]. M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for VM-based cloudlets in mobile computing. In IEEE Pervasive Computing, volume 8(4), October 2009.
[6]. E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making smartphones last longer with code offload, In Proc. ofMobiSys, San Francisco, CA, USA, June 2010.
[7]. M.R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan. Odessa: Enabling interactive perception applications on mobile devices. In Proc. of Mobisys, pages 43–56. ACM, 2011.
[8]. Y. W. Kwon and E. Tilevich. Power-efficient and fault-tolerant distributed mobile execution. In Proc. of ICDCS, 2012.
[9]. Mohammed A. Hassan, Kshitiz Bhattarai, Qi Wei1 and Songqing Chen, POMAC: Properly Offloading Mobile Applications to Clouds, In HotCloud'14 Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing, Pages 7-7.
[10]. S.Masiperiyannan, C.M.Mehathaf Begum, I.Mohammed Farook Ali,G.Mayuri Priya, S.Sudhakar, Security in Offloading Computations in Mobile Systems Using Cloud Computing, In International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Issue 3, March 2014
[11]. R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb. Simplifying cyber foraging for mobile devices. In Proc. of Mobisys, San Juan, Puerto Rico, June 2007.
[12]. Rajesh Balan, Jason Flinn, M. Satyanarayanan, Shafeeq Sinnamohideen, and Hen-I Yang. The case of cyber foraging. In Proceedings of the 10th ACM SIGOPS European Workshop, Saint-Emilion, France, July 2002.
[13]. Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In INFOCOM, 2012 Proceedings IEEE, pages 945–953. IEEE, 2012.
[14]. Mark S Gordon, D Anoushe Jamshidi, Scott Mahlke, Z Morley Mao, and Xu Chen. Comet: code offload by migrating execution transparently. In OSDI, 2012.
[15]. Arpneek Kaur, Narpat Singh, Abhishek Bahrdwaj, A Comparison of Supervised Multilayer Back Propagation and Unsupervised Self Organizing Maps for the Diagnosis of Thyroid Disease, In International Journal of Computer Applications (0975 – 8887) Volume 82 – No 13, November 2013.