MCM Analyzer: A Fuzzy Logic-based Offloading Decision Trade-off for Mobile Cloud Computing

Gustavo Mascarenhas Kath and Daniel Antonio Callegari
FACIN - Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre/RS, Brazil

Keywords: Fuzzy Logic, Mobile Cloud Computing, Offloading.

Abstract: Mobile cloud computing is an emerging solution for remote data processing and offloading. However, simply sending all data to the cloud is sometimes inviable because of security, network and cost issues. Consequently, there should be a trade-off between local and remote processing of data. Hence, an important issue to mobile solutions relies on developing a way to determine which sets of information can be processed locally by the device and which sets of information could be sent to the cloud for remote processing. This paper presents a fuzzy logic-based decision module for mobile cloud computing that can be added to an Android application. Our proposal analyses execution data in real time and helps resolving the local-remote decision trade-off. Experimental results show our proposal offers a promising approach when dealing with such scenarios.

1 INTRODUCTION

Mobile Cloud Computing is an emerging computing paradigm that leverages cloud computing resources to smart mobile devices. It uses cloud storage services for providing online storage, and cloud processing services for augmenting processing capabilities of such mobile devices. Processing capabilities of mobile devices are extended by outsourcing computational intensive components of the mobile applications to cloud data centers (Orsini, et. al., 2015).

Although executing heavy tasks on cloud infrastructure could be beneficial, always delegating execution to remote servers might not be advantageous and, in some cases, more resources are spent sending and receiving jobs over the network (Shiraz et. al. 2013).

Nevertheless, we have seen that the batteries on such devices currently do not withstand too long for some real world scenarios (Cuervo et. al, 2010). For this reason, the mobile device should be intelligent enough to decide whether outsourcing computation is beneficial or not for a particular scenario.

Because solving this tradeoff is often based on previous experience or on a set of heuristics and it is subject to the designer or programmer's judgment (Orsini, et. al., 2015), we believe fuzzy logic offers a promising point of departure.

The classical methodology on knowledge representation uses conventional logic, which is often inefficient when dealing with uncertainty and vagueness – it does not provide an adequate model for reasoning on approximate information. Fuzzy Logic provides an efficient conceptual base for dealing with the problem of knowledge representation in uncertain and imprecise environments. It also can be used as a decision mechanism (Zadeh, 1965) (Cox, 1995).

Our solution is composed of two main parts: the Mobile Cost Monitor (MCM) and the MCM Analyzer modules.

The MCM module is added to an Android application and it automatically collects several variables for later decision. In this sense, MCM acts as a software profiler or "context monitor". Then, the results are evaluated by the MCM Analyzer module, which fuzzyfies the input variables – such as execution time, CPU, memory usage, connectivity and battery level – applies a set of fuzzy rules, and then defuzzyfies a final variable indicating the favored result. The following sections present our proposal in more detail.

2 MOBILE COST MONITOR

The Mobile Cost Monitor (MCM) is a software component designed to monitor a running Android

Kath, G. and Callegari, D.
MCM Analyzer: A Fuzzy Logic-based Offloading Decision Trade-off for Mobile Cloud Computing.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 2*, pages 180-185
ISBN: 978-989-758-187-8

application and collect data such as CPU usage, memory usage, networking status, processing time and battery charge. MCM was developed as a module that can be added as a library to Android applications. Its purpose is to collect information about the overall execution of both the device and the application that it is integrated to. MCM requires Android version 2.3.3 (Gingerbread) or later.

2.1 Data Collected by MCM

MCM collects 18 different variables and their respective values about the device and the running application, e.g. battery level, temperature and status; network type, status, signal level, inbound and outbound traffic; shared and private memory amounts; low memory flag; and app and general CPU usage. Some values are acquired directly from the Android's API; others are internally computed based on historical data.

In order to capture information, MCM must be added to a host application as a class library. Then, the app developer decides the monitoring scope and creates an instance of the *Monitor* class. In the beginning of the desired application scope, the developer calls the *Monitor's start* method. MCM then starts collecting data in a configurable regular basis. The *Monitor's stop* method should be called at the end of the desired monitoring scope. This architecture encapsulates all functionality for the final development team.

3 MCM ANALYZER MODULE

MCM Analyzer is a software component that can be attached to the output of MCM in order to process the monitored data. MCM Analyzer was built to help evaluating whether processing should be executed locally by the device or else by the cloud infrastructure, i.e., in a remote environment.

The local processing environment in our scenario is the smartphone or the tablet computer. Such devices have greater limitations when compared to general-purpose computers. The hardware is often simpler and, most importantly, they run on batteries (as opposed to regular computers, which run plugged to a power outlet). Batteries provide considerable less energy for computational purposes and they do not stand long periods without recharging. Processing a large data set is a task that can draw a lot of energy from the battery. Moreover, some types of processing can also increase energy consumption because a higher frequency CPU usage directly draws more energy from the battery (Datta, 2013).

In certain scenarios, sending the data set to be processed by the cloud infrastructure can diminish this issue, as computational limitations would be lower in comparison. However, this is only possible when the smartphone is connected to the network. In the other hand, there is the cost of sending the data to the servers. The device's radio also consumes a considerable amount of energy, so that some experts recommend us to check whether the network connection is already established before trying to send data as well as to compress the data to be sent by the network infrastructure (Sharkey, 2009).

3.1 MCM Analyzer Heuristics

The MCM Analyzer module uses embedded heuristics in order to solve the offloading tradeoff. In order to decide which variables from MCM are useful to our decision process (and to that extent), we analyzed them one by one when running experiments. We found the following variables as the most important for our case:

Battery Level: This variable is considered critical to the decision process. A very low battery raises the risk to start a process, requiring more resources, and eventually not being able to complete processing. The remote processing time of the same information is many times expected to lessen the use of such resources, but we should not ignore the effort of sending and receiving the information back.

Battery Status: This variable is important to the decision tradeoff since we can easily determine whether the device is connected to some power source even when the battery level is low. However, because USB ports provide considerable less energy than standard power outlets, the device may inform it is charging even though the battery level is decreasing. In order to account for this issue we embedded specific rules to the decision heuristics.

Network is Connected: This is a critical variable because if the device is not connected to a network than it is obviously impossible to rely on the remote processing of the data. On Android devices, though, this variable only informs whether we are connected to some sort of networking (e.g. Wi-Fi, 3G, 4G), but it does not tell whether the connection itself is working properly.

Network Signal Level: This is an important variable in our scenario since the lower is the value, the greater is the risk in data loss during the transmission of data to a remote processing location.

This, in turn, will probably incur the need to resend data, as well as demand more control by the transmission protocol, consequently increasing energy consumption (Jung et. al. 2012).

Network Type: This is another important variable for our decision. Mobile networks such as 3G and 4G may compromise data transmission when compared to a generally more stable Wi-Fi connection. In addition, the smartphone's user may choose whether those types of connection are allowed or not. Finally, there is the monetary cost of using such connections, depending on the carrier.

Network is Roaming: The importance of this variable in our case is moderate. It is usually subject to additional charges from the carrier.

Network Wi-Fi TX and Network Wi-Fi RX: Both variables are relevant in choosing the processing environment. They inform the number of bytes that the monitored application sent and received via the network. The use of the network infrastructure has a direct impact on battery consumption.

PSS, Shared and Private Memory:
Although those variables are highly interrelated, we did not find enough and meaningful information from them that can be used for our decision tradeoff. The values for the variables do not find influence on information on energy consumption or connection data.
The drawback of using too much memory locally is that it supposedly increases the frequency of calls to the garbage collector, which according to (Li et. al, 2013) implies significantly on resource usage and also increases the execution time. Sending data to be processed by the cloud could help in reducing such cost as well as it helps in freeing memory. This heuristic can be provided by looking at the Low Memory Flag variable.

Available Memory: This is an important variable for the decision. However, it is not included in the decision heuristics because the Android API (Application Programming Interface) informs that its value should not be considered as an absolute value. Due to the nature of the operating system's kernel, a significant portion of this amount is required for the proper OS's performance.

Low Memory Flag: This is a very important variable for us, since it tells whether the system finds itself in a state of low available memory. Sending data for processing by the cloud is an alternative because it tends to use fewer resources and free memory.

Table 1: Sample rules from MCM Analyzer.

Rule	MCM variable	Favor local proc.	Favor remote proc.
Device is charging	Battery Status	X	
Device is discharging	Battery Status		X
Antenna has *weak* signal	Network Signal Level	X	
Antenna has *medium* signal	Network Signal Level		X
Antenna has *strong* signal	Network Signal Level		X
Battery charge is *high*	Battery Level	X	
Battery charge is *medium*	Battery Level		X
Battery charge is *low*	Battery Level		X
Device is connected to mobile network	Network Type	X	
Device is connected to a Wi-Fi router	Network Type		X
Device is roaming	Network is Roaming	X	

4 SOLVING THE OFFLOADING TRADEOFF

We developed a set of heuristic rules that act on the values of the acquired variables. Table 1 presents a sample of the rules. As we can see, some rules use conditions with constant values, while others present conditions with some uncertainty. Such points of uncertainty are written in italics in Table 1.

Additionally, rules have different levels of significance and distinct levels of interdependence. As an example, the third rule states that "*if the device has a weak antenna signal, then we should favor the local processing of data*" (the MCM variable Network Signal Level provides that information).

Table 2 presents part of the decision heuristics we developed when dealing with the inherent uncertainty of such type of information. The uncertainty measurements are placed in the first column and in the first row of the table, and the corresponding value for each rule is presented by combining both values.

Table 3 lists sample fuzzy rules for the local and remote execution time variables. We compare the local and the remote processing time by evaluating the rate between the total processing time as a function of the volume of processed data.

Table 2: Fuzzy input and output variables and terms.

Fuzzy Variable	Input Terms	Output Terms
Battery Level	Low	Remote ++
	Medium	Remote +
	High	Local ++
Network Level	Low	Local ++
	Medium	Remote +
	High	Remote ++
Network Connection	False	Local ++
	True	Remote ++
Network Type	Mobile	Local +
	Wi-Fi	Remote +
Network is Roaming	False	Remote +
	True	Local +
Battery Status	Not charging	Remote +
	Charging	Local +
Memory is Low	False	Local +
	True	Remote +

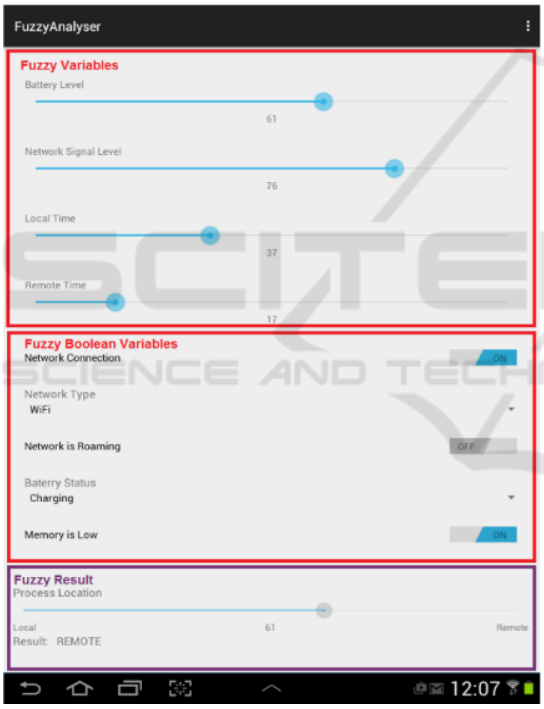

Figure 1: Interface of the testing application.

The output variables have the self-explanatory terms Local-, Local--, Remote+ and Remote++.

Table 3: Fuzzy variables for the execution times.

Remote time / **Local time**	***Short***	***Medium***	***Long***
Short	Local +	Local ++	Local ++
Medium	Remote +	Remote +	Local +
Long	Remote ++	Remote ++	Remote ++

In order to reassure our assumption that a fuzzy logic-based solution is applicable to such cases, we also developed a specific app that can be used to experiment with the heuristics that compute the final decision between processing the data locally or remotely by the cloud infrastructure. The app presents a user interface that allows the developer to experiment with the input variables for the decision system.

The application works on the input data, then uses the fuzzy inference engine and outputs the "Process Location" variable, which represents the degree to which the system "favors" the local or the remote processing of the data. By evaluating this output value in the range 0 to 100, one could make the final decision by applying some threshold interval, for example. Figure 1 shows the main interface of the app.

By experimenting with the testing app, the team can perform tests in order to fine-tune their heuristics for each particular case.

5 EXPERIMENTAL RESULTS

We ran our experiments in two distinct Android smartphones: a Samsung Galaxy S3, 1.4 Ghz, 1 GiB RAM, running Android 4.3 and a Motorola Moto X, 1.7 GHz, 2 GiB RAM, running Android 4.4.4.

In our experiments, we found some interesting relations among the data collected by the MCM module. Those relations are important because if we find a correlation between two or more variables, it means we can reduce the monitoring overhead by leaving the dependent variables out of the sampling.

This is very important when profiling applications in order to make decisions since all profiling implies both a processing and a potential memory overhead. Thus, by using less CPU cycles, less memory and less battery resources, we can minimize the computational footprint of the MCM module.

As an example, if two variables are linearly proportional, then it may be only necessary to collect one of them in order to make a decision based on their values. The testing application provides an easy to use fuzzy inference engine that enables the development team to try several combinations of values for the input variables. It also provides a convenient way to fine-tune the fuzzy inference rules for the offloading decision to a particular case.

6 RELATED WORK

During the development of this research, we found other works and applications already available with similar functions and ideas of MCM. We observed, however, that much of such initiatives were only intended for application debugging purposes (before the application was made available to the public) and not for real time execution scenarios as in our case. Two examples of such initiatives are DevScope and the Android SDK's Dalvik Debug Monitor Service (DDMS) (Jung et. al, 2012) (Google, 2015).

Some other initiatives are Qualcomm's Trepn Profiler (Qualcomm, 2015), and AppScope (Yoon, 2012). However, those profiling modules do not focus on getting hardware specific information for the running application but only for the device as a whole. Such specific information cannot be obtained via API calls because they are only available when debugging is active.

AppScope and DevScope, in particular, focus on the development of an energy model of the mobile device. An energy model is a framework based on mathematical calculations for estimating energy expenditure of the main hardware components of a device and therefore the total expenditure power of the device. According to the authors, this technique allows for a decision-making process based on energy expenditure of certain components, as well as the system altogether. In addition, it is also possible to estimate the remaining battery time. Among all related work, Qualcomm's Trepn Profiler application is the one that most resembles Mobile Cost Monitor.

Nevertheless, as stated previously, because the offloading decision in mobile cloud computing environments is often based on expert judgment, we suggest using approaches that better deal with uncertainty and yet provide convenient and suitable decision mechanisms for such scenarios.

7 CONCLUSIONS

As a model that enables ubiquitous network access to a pool of computing and networking services, Cloud Computing provides computational resources in an on-demand, pay-as-you-go manner through minimal interaction with the service provider. This sort of utility computing infrastructure offers a scalable environment to store and also process large amounts of data. A solution to the offloading trade-off may benefit from a fuzzy logic-based approach.

This paper presented MCM and MCM Analyzer modules, as well as our investigation of variables and potential fuzzy logic rules to address this decision. This work offers new insights that build on top of our previous research (Callegari et. at., 2013).

We developed different Android applications and running scenarios, compared them and performed several experiments in order to attest its practicality. We verified that MCM and the decision mechanism represent a new approach to mobile cloud computing solutions. The next steps involve improving our solution by adding learning capabilities. MCM runs on Android devices and it enables the collection of several types of information that allow the technical team to evaluate proper scenarios where a mobile could computing solution is feasible or even necessary.

REFERENCES

Callegari, D. A., Jersak, L. C., da Costa, A. C., 2013. Technical Trends and Challenges in Mobile Health - A Systematic Review of Recent Available Literature. In *Proceedings of the 15th International Conference on Enterprise Information Systems*.

Cox, E. D., 1995. *Fuzzy Logic for Business and Industry*. Charles River Media.

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, R. Chandra, R., Bahl, P. 2010. MAUI: making smartphones last longer with code offload. In *8th International Conference on Mobile Systems, Applications, and Services*.

Datta, S. K., Bonnet, C., Nikaein, N., 2013. Minimizing energy expenditure in smart devices. In *IEEE Conference on Information and Communication Technologies*.

Google Inc.. Android., 2015. Debugging using DDMS. Available: http://developer.android.com/tools/debugging/ddms.html.

Jung, W., Kang, C., Yoon, C., Kim, D., Cha, H., 2012. DevScope: a nonintrusive and online power analysis tool for smartphone hardware components. In. *Proceedings of the 8th IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*.

Orsini, G., Bade, D., Lamersdorf, W., 2015. Context-Aware Computation Offloading for Mobile Cloud Computing: Requirements Analysis, Survey and Design Guideline. In *12th International Converence on Mobile Systems and Pervasive Computing*.

Qualcomm Inc., 2015. Trepn Profiler. Available https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler.

Sharkey, J., 2009. Coding for Life - Battery Life, That Is, Google I/O. Available: https://dl.google.com/io/2009/pres/W_0300_CodingforLife-BatteryLifeThatIs.pdf.

Shiraz, M., Gani, A., Khokhar, R., Buyya, R., 2013. A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing. In *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3.

Yoon, C., Kim, D., Jung, W., Kang, C., Cha, H., 2012. AppScope: application energy metering framework for android smartphones using kernel activity monitoring. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference*.

Zadeh, L. A., 1965. Fuzzy sets. *Information and Control*. 8:338-53.