Smartphone Energizer: Extending Smartphone's Battery Life with Smart Offloading

Ayat Khairy
Department of Computer Science, Faculty of Computers and Information
Cairo University, Egypt
ayat.khairy@fci-cu.edu.eg

Hany H. Ammar
The Lane Computer Science and Electrical Engineering Dept., West Virginia University, USA
hammar@wvu.edu

Reem Bahgat
Department of Computer Science, Faculty of Computers and Information
Cairo University, Egypt
r.bahgat@fci-cu.edu.eg

***Abstract*—This paper presents "Smartphone Energizer", a novel technique for context-aware computation offloading for smartphones. Previous techniques to the offloading problem were based on a narrow set of contextual information, which made the amount of energy saving varies unexpectedly based on the context in which the application is running. Smartphone Energizer uses the benefit of supervised learning with a rich set of contextual information such as application, device, network, and user characteristics to optimize both the energy consumption and execution time of Smartphone's applications in a variety of contextual situations. In our evaluation, we show that Smartphone Energizer predicts both energy consumption and execution time in different contexts with error less than 9%, which in turn helped in taking the right offloading decision and saving energy by 40% to 56% and execution time by 43% to 58%.**

Keywords— Computation offloading; context-awareness; application partitioning; Smartphones; energy efficiency

I. INTRODUCTION

[1]Recently, we have witnessed a tremendous growth in smartphones; their capabilities increased over the years especially with regards to their processing power and storage. They are now equipped with rich sensors (e.g. GPS, accelerometer, light, and pressure sensors), have higher screen resolution and offer more communication bandwidth. Furthermore, the maturity of mobile applications has been reshaped from applications that perform basic computations to computationally intensive ones, ranging from advanced 3D games [1], to image processing [2], speech recognition [3], and augmented reality [4] applications. This has led to a growing need for energy in a way that outpaces the mobile battery's capacity [5], and is not aligned with the growth in the battery technology. Consequently, the battery technology has become one of the biggest obstacles standing in the way of future growth of smartphones usage [6].

In this paper we present "Smartphone Energizer", a context-aware technique for the smartphone's computation offloading and therefore it allows extending the smartphone's battery life. Smartphone Energizer enables a smartphone to perform computationally intensive tasks and maintain the Quality of Service (QoS) while saving the device's energy.

We conducted a set of experiments to illustrate the motivation behind context-aware computation offloading, we used Power Tutor [7] which is a power consumption analysis tool that tracks the power consumed by the major device's components such as CPU, network interface, display, and GPS on both the application and system level. We measured the energy consumption in different contexts (e.g. different application, network, user, and device characteristics). The networks we used in the experiment are WIFI 801.11n, and Mobinil 3G network that uses HSDPA with a bandwidth up to 3.6 Mbps. The energy consumption varies, as shown in Figure 1, with respect to the different contextual information and, accordingly, the computation offloading decision should vary accordingly.

When running the QR code reader application [8] to read three QR codes on LG Optimus ME P350, which is relatively a low end smartphone (600 MHz CPU), it consumed relatively a lot of energy by the display and the CPU. Therefore, computation offloading may be considered in this case if it will cut the application execution time and hence cut the consumed energy used by the display and CPU. On the contrary, running the same application on HTC Incredible S (IS), which is a high end smartphone (1GHz CPU), did not consume much of energy and therefore computation offloading need not be considered.. Additionally, Fig. 1 shows the energy consumption of running Photo translator (PS) [9], which is an application that performs OCR on textual images captured by the user then translates. The energy consumption varied depending on the context in which the application was running. Running the application over WIFI consumed less energy than running it over 3G, and running the application over 3G while the device is near a cell tower consumed less energy than running the application over 3G while the user is far from a cell tower.

[1]this work is sponsored by Orange Labs Cairo

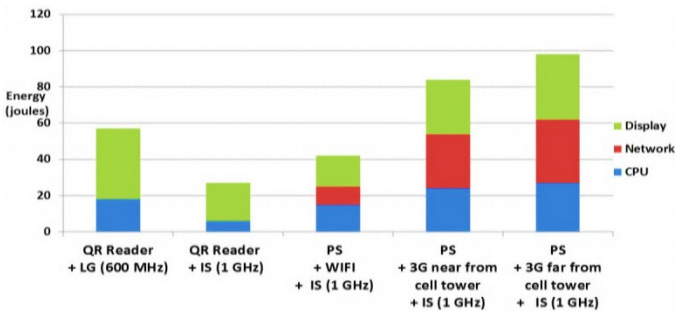

Fig. 1. Smartphone energy consumption with respect to different contextual situations

This paper has three main contributions:

- We propose a novel, adaptive, and context-aware offloading technique that uses Support Vector Regression (SVR) and several contextual features to predict the remote execution time and energy consumption then takes the offloading decision. The decision is taken so that offloading is guaranteed to optimize both the response time and energy consumption.
- We build a rich dataset based on different types of devices, networks, users, and application characteristics to predict the remote execution time and energy consumption accurately. Additionally, we analyze how each contextual feature contributes to the accuracy of the application energy and time consumption prediction.
- We evaluate our offloading technique with three real life example applications. During the evaluation, we run these applications under various situations (i.e. different devices, usage, and networks' characteristics).

The rest of the paper is organized as follows. In Section II we briefly give a background of computation offloading, we present Cuckoo which is a smartphone computation offloading framework that we extended to implement the proposed Smartphone Energizer technique, and we briefly describe SVR. Then in Section III, we explain our proposed technique to tackle the offloading problem. In Section IV, we give an overview on the related research in computation offloading and energy efficiency of mobile clients. Then, in Section V, we present and discuss the experimental results of using the proposed technique in computation offloading. Finally, section VI concludes the paper and outlines our future work.

II. BACKGROUND

Computation Offloading concerns the transfer of the processing of a computationally intensive task from a thin client like a smartphone to a rich server or cloud. Computation offloading was proposed initially to improve application's performance. However, since technology trends for the batteries show that their limitations will stay [5] and the energy will remain the main bottleneck for smartphones, Computation offloading has been viewed also as an opportunity to conserve smartphone's energy and extend its battery life.

A. Cuckoo

Cuckoo [10] is a computation offloading framework that targets the Android platform; it offers a simple programming model that supports local and remote execution. Cuckoo offloading mechanism takes the offloading decision based on the server availability only, without considering other contextual information. In our work, we extend the Cuckoo framework to implement our proposed offloading technique due to the following reasons:

- Cuckoo targets the android platform which is an open source and popular platform. Cuckoo offers a simple programming model and automates large parts of the development process; the application developer writes the offload-able code in a regular Android service using Android AIDL.
- It integrates with existing development tools that are familiar to developers such as Eclipse IDE, which does not put a burden on the application developers.
- The application user can easily add remote resources including laptops, home servers and other cloud resources to offload the computation to them; the application user simply pairs with the offloading server by scanning the QR code of the offloading server URL using his smartphone.

Cuckoo uses the Android programming model in order to determine the offload-able code in the application. Two of the main components of the Android applications are activities and services. Activities are components that interact with the user; they contain the graphical user interface and perform basic computation. On the other hand, services contain the CPU or network intensive operations and run in the background; they do not have graphical user interfaces. Cuckoo considers the application's services as the offload-able code in the application. Cuckoo takes the offloading decision based on the availability of a server to offload the computation to it. If there's a reachable server, then it will offload the service to the server. Otherwise, it executes the service locally. It takes the computation offloading decision based on the server availability only without considering any other contextual information. This raises the question of whether this guarantees conserving the smartphone's energy. The answer is either yes or no. It is yes if the computation cost outpaces the communication cost and the overhead of data movement is small, and no if the communication cost exceeds the computation cost. This could occur in certain cases such as offloading a small computational task that need not be considered for offloading, or offloading a computationally intensive task that takes or returns data of

large size over a network with limited bandwidth. We extend the Cuckoo framework to enrich its offloading mechanism to achieve context-aware energy conservation.

B. Supervised Learning

In supervised learning, a training set is constituted from a set of input data along with their corresponding labels. The learning algorithm uses the training set to build a model that enables making true predictions of unseen cases. The true values that we try to predict are the class labels of the unlabeled data in case of classification, while they are the predicted values in case of regression. Support Vector Machines (SVM) [11] is one of the recent state-of-the-art learning algorithms; it has become increasingly popular due to its significant generalization performance on many real-life problems. SVMs are based on the structural risk minimization (SRM) principle which has been shown to be better than the traditional empirical risk minimization (ERM), which used by conventional neural networks [12]. SRM minimizes the upper bound of the generalization error, unlike ERM which minimizes the training error only. SRM increases the generalization performance of SVM, which is needed in the learning problems. Additionally, training SVMs is equivalent to solving a linearly constrained quadratic programming problem, which makes the solution of SVMs unique and globally optimal, unlike other network's training which requires nonlinear optimization with the risk of getting stuck into local minima. In SVR the basic idea is to map the input space data to a higher dimensional feature space through nonlinear mapping, then to perform linear regression in this space.

In this work, we build SVR model to predict the remote execution energy consumption and time, given several contextual information. We perform grid search and cross validate the training dataset to identify the best parameters for the model.

III. SMARTPHONE ENERGIZER: OUR TECHNIQUE

A. Cuckoo Extensions

Cuckoo decides to offload computation without considering key contextual information. In our technique, in order to guarantee optimizing both response time and energy consumption, we enrich the offloading decision making process with different contextual information, which covers the following:

- Network characteristics, namely network interface type (e.g. WIFI, 2G, 3G, etc.), link speed, RTT, and network bandwidth. We selected these network parameters as the amount of consumed energy could be inversely proportional with them, while the response time could be directly proportional with them.
- Service (offload-able code) characteristics: service input size, computational needs, and its local execution time play important role in the decision making process. This is why each service is assigned an identifier, which uniquely identifies the service to facilitate tracking its behavior as well as predicting its future computational and energy needs.
- Device characteristics, namely the mobile's CPU speed and model, as the amount of consumed energy depends on the device hardware characteristics (e.g. processor speed, and network interface transfer bit-rate).
- User characteristics: battery level, date, time, and location information help in enriching the decision making process. We defined three testing locations with varying network coverage and distance from cell towers; during profiling and testing we select one of these three locations from Smartphone Energizer Client (SEC) depending on the testing location.

B. Smartphone Energizer Offloading Algorithm

Smartphone Energizer is a supervised learning-based technique for energy efficient computation offloading. Fig. 2 shows a high level view of the Smartphone Energizer's components, namely: On the smartphone, 1) A client which controls invoking and transferring data for the offloaded services; 2) a profiler which collects the measurements of energy consumption and execution time aligned with the different contextual information that was mentioned in section III.A; and 3) a local consumption predictor, which predicts the cost of executing the service locally (in terms of energy and execution time) based on the saved history by the profiler. SEC switches between two modes, namely, *learning* and *prediction* modes; these two different modes will be explained below.

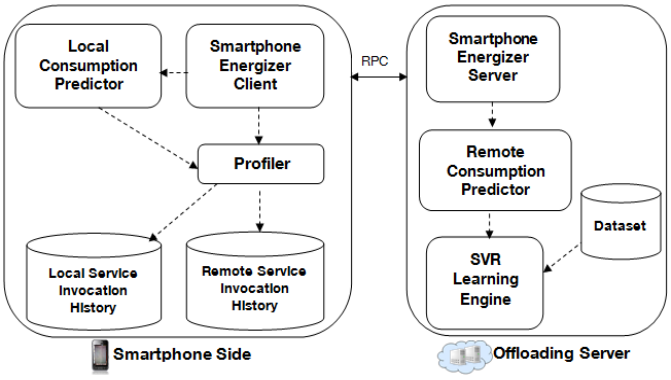

Fig. 2. High level view of Smartphone Energizer's components

1) Learning Mode: Feature extraction

SEC initially starts in the *learning* mode, in which it extracts the different network, device, and application features and stores them after each service invocation.

a) *Local Execution Cost Learning:* at the beginning of the learning mode, the service will be executed locally for a fixed number of times (local learning capacity). After each local service invocation, the Smartphone Energizer's profiler stores the context parameters which include the service identifier,

input size, and output size along with the consumed energy and time during service execution.

b) *Remote Execution Cost Learning:* When the number of local service invocations exceeds the local learning capacity, the SEC switches to the remote execution by checking if there's a reachable offloading server, then the service will be installed on the server (for the first time only) and will be executed remotely, otherwise it will be executed locally. After each remote service invocation, the profiler stores the different contextual information that was mentioned in section III.A along with the service's consumed energy and time during execution. The Smartphone Energizer's client remains in the learning mode as long as the size of the stored remote service invocations history does not exceed a predefined size (the remote learning capacity which is a constant number specified based on trial and error).

2) Prediction Mode

If the size of the profiled data for both local and remote service execution exceeds the learning capacity, then SEC switches from learning to prediction mode, in which prediction models are constructed.

To build the remote execution cost model, the SEC sends the recorded data to the offloading server, and then deletes this recorded data from the smartphone's memory to reduce the memory consumption and free a space for new data to be recorded. Based on this data, the offloading server receives the profiled data from the client, puts the received data in SVR dataset format, cross validates the dataset and searches for the best parameters to build the prediction model, and finally builds an energy consumption model and execution time model that enables the prediction of remote execution energy consumption and time, as illustrated in Figure 3. To predict the Local Execution cost, the SEC uses local consumption predictor to predict the local execution energy consumption and time. The prediction is kept simple since the prediction process is performed on the smartphone. This is why our local consumption predictor calculates the local execution cost by taking the average of the previously recorded cost of the local service execution.

3) Decision Making

On the smartphone, when the service is invoked, as long as SEC is in the *prediction* mode, it decides whether to offload computation or not by sending the current contextual information (service identifier, input size, network interface type, link speed, rough estimate of bandwidth, device type, date, time, and location) to the server to predict the remote execution energy consumption and time. The client compares the predicted remote execution energy consumption and time with the predicted local execution energy consumption and time (based on equations 1, 2, and 3) then decides whether it's beneficial to offload the computation or not.

$$E(C_l) > E(C_r) \quad (1)$$

Where $E(C_l)$ is the expected cost of local execution and $E(C_r)$ is the expected cost of remote execution in terms of energy and time.

$$E(C_l) = \alpha E(C_{le}) + \beta E(C_{lt}) \quad (2)$$

$$E(C_r) = \alpha E(C_{re}) + \beta E(C_{rt}) \quad (3)$$

$E(C_{le})$ is the expected cost of local execution in terms of energy, $E(C_{lt})$ is the expected cost of local execution in terms of time, $E(C_{re})$ is the expected cost of remote execution in terms of energy, $E(C_{rt})$ is the expected cost of remote execution in terms of time, and α, β represent the significance of execution time and energy in the application respectively, and ranges from 0 to 1. In our work we set both α, β to 0.5 to equalize the significance of energy saving and execution time, however Smartphone Energizer's users (application developers) can set α, β depending on their preferences, and whether they care about the energy consumption more or less than the execution time or equivalently. For example in the responsive applications, the application developer could set α (execution time significance) to 0.75 and β (energy consumption significance) to 0.25 to indicate that the significance of execution time is three times more than the significance of energy consumption. If the application user's battery level is less than 15%, then SEC set α to 0 and β to 1 to indicate that energy consumption has the most important significance in taking the offloading decision regardless the execution time. Fig. 3 depicts how Smartphone Energizer keeps adapting the remote execution prediction model even after switching to the *prediction* mode. Smartphone Energizer saves the different contextual information defined above in section III.A along with the consumed energy and time during service execution and uses this new data to update the prediction model. SEC keeps updating the remote execution prediction model till a suitable mean error is reached (less than 10%).

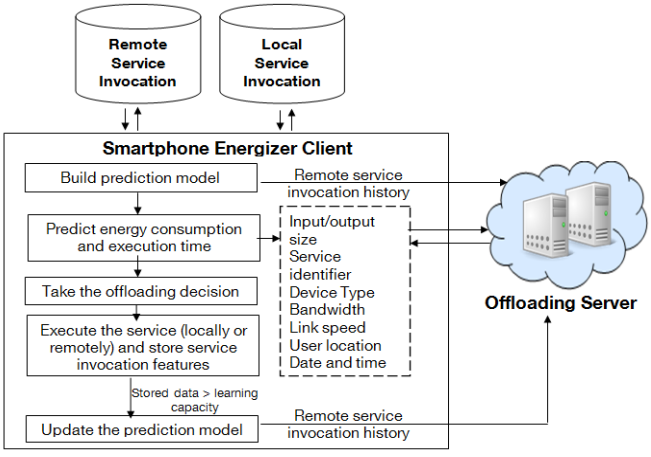

Fig. 3. Smartphone Energizer client in the prediction mode

IV. EVALUATION

In this section, we evaluate the Smartphone Energizer's ability to improve the energy consumption and performance of smartphone applications, using three different metrics. The first metric is to evaluate how much does Smartphone Energizer reduce the energy consumption and improve the performance of smartphone applications. The second metric is to evaluate how each aspect of the context contributes to the accuracy of consumption prediction. The third metric is how Smartphone Energizer deals with different contextual scenarios.

During the learning and prediction phases, we used two CPU intensive android applications that were pre-built but we modified their code to make them run on our extended framework. The applications are Eyedentify [2] which is an object recognition application that has an offload-able service that gets the feature vector of an image to recognize, and OCR translator [13] which is an application that recognizes the characters of a captured image then translates, we modified the code of the OCR translator to make it provide an offload-able service that performs OCR on the captured images. We run the applications several times using random inputs on different devices with different characteristics and processing capabilities, namely, Google Nexus S (1GHz CPU, android 2.3), and LG Optimus ME P350 (600 MHz CPU, android 2.2) and under different contextual situations such as different network types (WIFI 801.11n, and Mobinil network with varying bandwidth and link speed such as HSDPA, UMTS, and EDGE), different locations with different distances from the cell tower and different network coverage. For the offloading server, we used dual-core desktop with a 3GHz CPU and 4GB RAM. For energy consumption measurements, we used a power- meter [14] attached to smartphone's battery that samples the drawn current from the battery at 3000Hz and obtains fine-grained energy measurements.

A. Methodology

In our evaluation, we use the same devices that we used in building the prediction model, as well as adding a new device with varying characteristics and processing capabilities to figure out how would Smartphone Energizer deal with changing device characteristics. The new device is Samsung Galaxy Y (835 MHz CPU, android 2.3). Smartphone Energizer is evaluated using the two applications that were used in building the prediction model (Eyedentify, and OCR Translator), as well as another more complex application to figure out how Smartphone Energizer would adapt to changing application's needs. The new application is Photoshoot [15] which is a distributed augmented reality game in which two players fight a virtual duel in real world, they shoot with virtual bullets and the game applying face detection algorithms to decide whether shots are hit or not, Photoshoot has an offload-able service that recognizes the opponent face, we modified the code of photoshoot to make it work using our extended framework.

B. How much does Smartphone Energizer reduce the energy consumption and improve the performance?

To evaluate the benefits of Smartphone Energizer, we run the test applications under 25 varying contextual scenarios; 9 scenarios on Nexus S, 8 scenarios on LG Optimus ME, and 8 on Galaxy Y. On each device, the three test applications are run under varying network types (WIFI 801.11n, and Mobinil network with varying bandwidth and link speed such as HSDPA, UMTS, and EDGE), different locations with varying distance from the cell tower, and random inputs with different sizes. Then we compare the accumulative energy consumption and execution time of running these scenarios using Smartphone Energizer, cuckoo, and locally. Fig. 4 and 5 depict how Smartphone Energizer context-awareness optimized both the overall energy consumption and execution time compared to cuckoo and local execution. For Eyedentify, Smartphone Energizer saves 40% of the consumed energy and 43% of the consumed time by local executions. For OCR Translator, Smartphone Energizer saves 52% of the consumed energy and 54% of the consumed time by local executions. For Photoshoot, Smartphone Energizer saves 56% of the consumed energy and 58% of the consumed time by local executions.

In order to calculate the overhead of our proposed technique, namely, the overhead of profiling the contextual information and the overhead of building local/remote cost model, predicting energy/time cost; we made a standalone mobile application that takes as an input the number of application executions locally and remote (assume x is the number of local application executions and y is the number of remote application executions), performs the overhead functionalities (profiles the local contextual information for x times and the remote contextual information for y times, builds the local cost model, sends the profiled data to the server to build the remote execution cost model, and predicts energy/time cost), then measured this standalone application execution cost in terms of time and energy, and finally compared that to the total energy/time cost of x and y application executions using Smartphone Energizer. The overhead did not exceed 6 % of the total energy/time cost of the application executions.

Smartphone Energizer saves more energy and time than cuckoo since in certain contextual situations Smartphone Energizer decides not to offload and performs the computation locally -unlike cuckoo- to save more energy/time; some of these cases are described in section IV.D.

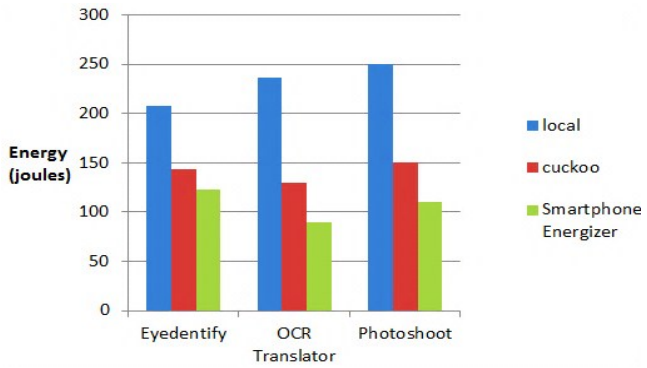

Fig. 4. Eyedentify, Photoshoot, and OCR Translator energy consumption (in joules) while executing them locally versus using cuckoo and Smartphone Energizer

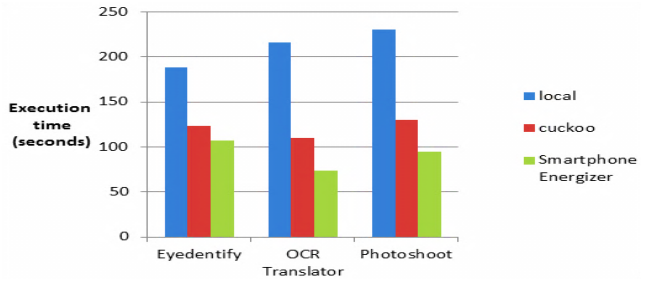

Fig. 5. Eyedentify, Photoshoot, and OCR Translator execution time (in seconds) while executing them locally versus using cuckoo and Smartphone Energizer

C. *How each aspect of the context contributes to the prediction?*

In order to figure out how each aspect of the context contributes to the prediction, we re-build the prediction model and test it by leaving out individual features to learn how much precision is gained from each of them. From the testing scenarios that we mentioned in section IV.B, we build test dataset of twenty test cases (features aligned with the actual consumed energy and time).

Fig. 6 and Fig. 7 show the MSE of the energy consumption and execution time test datasets, the testing carried out using all the features, all the features without network features (network type, link speed, RTT, and bandwidth), all the features without location, all the features without application's features (CPU needs), all the features without timing's features (date and time), and finally all the features without input size. As seen in Figure 6 and 7, using the entire features, Smartphone Energizer maintains a high level of prediction accuracy (the model mean error is less than 9%) given different contextual situations, opposite to other techniques which ignored this contextual information. Using all the features without network, application, device, user location, and timing features also varies in their effect on the mean square error (MSE). Removing the input size (the amount of data that is offloaded over the network and get processed at the server) feature, and network features (network type, link speed, rough bandwidth, and RTT) has the highest impact on the prediction accuracy. Device, application, timing and location features also affected the prediction accuracy respectively.

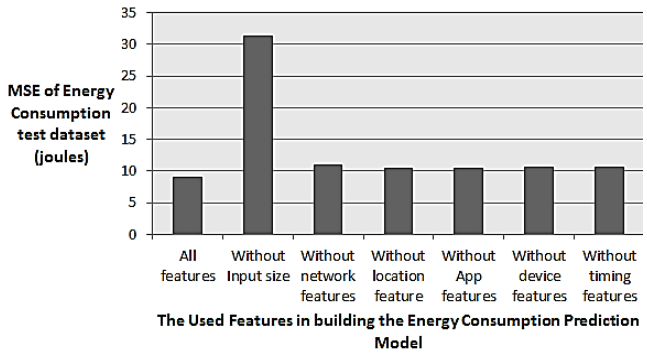

Fig. 6. MSE of energy consumption test dataset in joules using different features

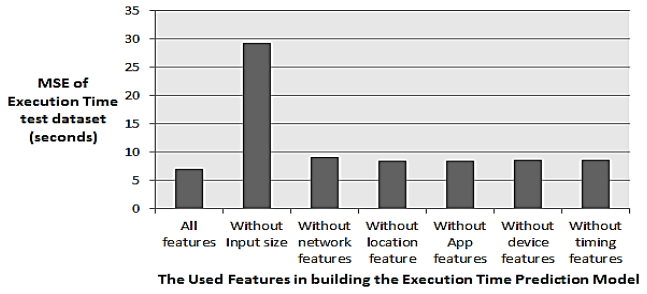

Fig. 7. MSE of Execution time test dataset in seconds using different features

D. *How would Smartphone Energizer deal with varying contextual scenarios?*

To answer this question, we need to highlight three example scenarios in which Smartphone Energizer deals with varying contextual situations. The first example is running Eyedentify over 3G/WFI on Nexus S device, Eyedentify's processing needs is relatively simple and copes with Nexus S processing capabilities (1 GHz CPU). Hence Smartphone Energizer decides not to offload the computation and to perform the processing locally, unlike the case of offloading other applications with a higher processing needs or the case of offloading the same application on a device with limited processing capabilities (LG Optimus Me 350 and Galaxy Y). The second example is offloading Eyedentify over 3G on Galaxy Y, Smartphone Energizer decides not to offload, unlike the case of offloading over WIFI. The third example is how Smartphone Energizer deals with a different device (Galaxy Y) and application (Photoshoot) that were not used during building the prediction model, Smartphone Energizer predicts (mean error less than 9%) the energy consumption and execution time of Photoshoot application under varying contextual scenarios such as running it on LG Optimus Me 350 and Nexus S (devices that were used in model building) and Galaxy Y (not used during model building). In conclusion, we can see that context-awareness is important at the learning and prediction/decision making phases to achieve more accurate offloading decisions.

V. RELATED WORK

Computation offloading used to be seen as an opportunity to increase the performance and responsiveness of the

resource-constrained devices like mobile phones. However since the growth in the smartphone's processing and storage technology outpaces the growth in their battery technology, computation offloading has been seen as a solution for the smartphone's energy bottleneck problem. In this Section, we give an overview of the carried out research in computation offloading and energy efficiency of mobile clients.

In [16], Cuervo et al proposed a system called MAUI that makes a smartphone last longer using computation offloading. In this system, developers annotate the methods that they want to execute remotely as REMOTABLE. MAUI dynamically profiles the smartphone's applications and optimizes both the energy consumption and execution time using an optimization solver.

Similar to MAUI, Chun et al proposed CloneCloud in [17]; which is a system for elastic execution between mobile and cloud through dynamic application partitioning, where a thread of the application is migrated to a clone of the smartphone in the cloud. CloneCloud is like MAUI since it uses dynamic profiling and optimization solver, but CloneCloud goes a step further as partitioning takes place without the developer intervention; application partitioning is based on static analysis to specify the migration and re-integration points in the application. Unlike our proposed technique, neither MAUI nor Clonecloud take into consideration a rich set of contextual information during profiling or at the offloading decision making phase.

Kumar et al formulated an equation of several parameters to measure whether computation offloading to cloud would save energy or not in [18]. These parameters are network bandwidth, cloud processing speed, device processing speed, the number of transferred bytes, and the energy consumption of a smartphone when it's in idle, processing and communicating states. In this concept paper, the authors only discussed these various parameters and did not experiment their work in a real offloading framework. [19] discussed the energy efficiency of mobile clients and how there are several metrics for characterizing the energy consumption of communication; not only the amount of transferred data alone but also the energy characteristics of the device's wireless transfer, communication bit-rate, traffic pattern, and whether the user is near or far from the base station in case of cellular communication, they just discussed the various metrics that characterize computation offloading in this concept paper and did not experiment their work in a real offloading framework.

Our proposed technique takes into consideration all the parameters in [16, 17, 18, and 19]; additionally, we go step further by considering extra parameters that relate to user characteristics to enrich the offloading process. We tested the impact of each contextual feature on the offloading. Moreover, we experiment the usage of all these parameters at the profiling and decision making phases through real offloading of three applications.

More recently, [20] proposed ThinkAir which is a computation offloading system that is similar to MAUI and cloudclone; ThinkAir does not only focus on the offloading efficiency but also on the elasticity and scalability of the cloud side; it boosts the power of mobile cloud computing through parallelizing method execution using multiple virtual machine images. [21, 22] proposed SmartDiet which is a toolkit that helps the developers to identify the constraints that reduce offloading opportunities (e.g. hardware accessing) and calculates the energy-saving potential of offloading communication-related tasks; SmartDiet mainly focuses on analyzing the opportunities of offloading the communication-intensive tasks to save energy; unlike the earlier proposed work which focuses on offloading the compute-intensive tasks.

Besides computation offloading, other solutions have been proposed to make smartphone's battery lasts longer. Some are software-based solutions, such as applying power saving modes on processes, network interface cards, display, and sensors like GPS [23]. Other solutions are hardware-based such as adding low power processor to partition the processing between the main processor and the additional processor to save energy [24], developing rapid charging devices which cut the full charging time to half, or developing wireless chargers to facilitate the charging process. Furthermore, research has been carried out to enable charging the phone from different surrounding sources such as light, speech, movement, bacteria, and human heart's beats [25, 26].

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented Smartphone Energizer, a context-aware offloading technique that enables the smartphone to run computationally intensive applications while saving the smartphone's energy. The proposed technique uses SVR to predict energy consumption and execution time, and then takes the offloading decision that maintains the QoS and saves the device's energy. This paper presented how Smartphone Energizer's client works in two modes, *learning* and *prediction*. We evaluated the performance of this technique taking into consideration a variety of contextual information. Our results showed that Smartphone Energizer achieved a high accuracy (the model mean error is less than 9%) in predicting both energy consumption and execution time given different situations, which in turn helped in taking the right offloading decision.

In our future work, we are planning to extend the energy consumption model to include privacy and security features, investigate using fuzzy logic in partitioning the offloaded service execution, instead of the 0-1 model of cuckoo which either offloads the whole service or not. The current implemented technique assumes that only one instance of the offloaded applications is running; we plan to handle the multiple concurrent executions of the offloaded applications on smartphone at the profiling time, and investigate how to divide up the energy consumption across these concurrent applications. Additionally, in the current

implementation we are assuming the offloading server is one node with relatively high processing capabilities, however in the future work we are planning to inspect the architecture requirements of the server in order to leverage the server scalability, and find answers to questions like should each server keep a consistent model for remote energy and time consumption prediction? How will that scale?

REFERENCES

[1] Giurgiu.I, Riva.O, Juric.D, Krivulev.I, and Alonso.G. 2009. Calling the cloud: Enabling mobile phones as interfaces to cloud applications, Middleware '09, 83–102.

[2] Kemp.R, Palmer.N, Kielmann.T, Seinstra.F, Drost.N, Maassen.J, and Bal.H.E. 2009. EyeDentify: Multimedia Cyber Foraging from a Smartphone, ISM'09, 392-399.

[3] Goyal.S and Carter.J. 2004. A lightweight secure cyber foraging infrastructure for resource-constrained devices, Sixth IEEE Workshop on Mobile Computing Systems and Applications'04, 186–195.

[4] http://www.layar.com/. Last visited: March 29, 2013.

[5] Palacin.M.R. 2009. Recent advances in rechargeable battery materials: a chemists perspective, Chemical Society Reviews, volume 38, issue 9, 2565–2575.

[6] User satisfaction with smartphone survey.2010. http://whatjapanthinks.com/2011/02/15/battery-dissatisfaction-high-amongst-smartphone-users/. Last visited: March 29, 2013.

[7] Zhang.L, Tiwana.B, Qian.Z, Wang.Z, Dick.R, Mao.M, and Yang.L. 2010. Accurate online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones, IFIP'10, 105-114.

[8] QRDroid. http://www.qrdroid.com/. Last visited: March 29, 2013.

[9] PhotoTranslator. http://www.smartmobilesoftware.com. Last visited: March 29, 2013.

[10] Kemp.R, Palmer.N, Kielmann. T, and Bal. H. 2010. Cuckoo: a Computation Offloading Framework for Smartphones, MobiCASE'10, 59-79.

[11] Lovell.B.C, and Walder.C.J. 2006. Support Vector Machines for Business Applications, Business Applications and Computational Intelligence Idea Group, 267-290.

[12] Collobert. R, and Bengio. S. 2001. SVMTorch: Support Vector Machines for Large-Scale Regression Problems, Journal of Machine Learning Research, 143-160.

[13] OCR Test. http://www.rmtheis.com. Last visited: March 29, 2013.

[14] Mansoon Power Monitor. http://www.msoon.com/LabEquipment/PowerMonitor. Last visited: March 29, 2013.

[15] Kemp.R, Palmer.N, Kielmann.T, and Bal.H. 2010. Opportunistic Communication for Multiplayer Mobile Gaming: Lessons Learned from PhotoShoot, MobiOpp'10, 182-184.

[16] Cuervo. E, Balasubramanian. A, Cho. D, Wolman. A, Saroiu. S, Chandra. R, and Bahl. P. 2010. MAUI: Making smartphones last longer with code offload, MobiSys'10, 49-62.

[17] Chun.B.G., Ihm.S, Maniatis.P, Naik, M, and Patti, A. 2011. CloneCloud: Elastic Execution between Mobile Device and Cloud, EuroSys'11, 301-314.

[18] Kumar.K,and Hsiang Lu.Y. 2010. Cloud computing for mobile users, Computer'99, volume 43, issue 4, 51-56.

[19] Miettinen.A.P, and Nurminen.J.K. 2010. Energy efficiency of mobile clients in cloud computing, 2nd USENIX conference on Hot topics in cloud computing, 4-4.

[20] Kosta.S, Aucinas.A, and Hui.P. 2012. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, INFOCOM'12, 945–953.

[21] Saarinen.A, Siekkinen.M, and Xiao.Y. 2012. SmartDiet: offloading popular apps to save energy, SIGCOMM'12, 297–298.

[22] Saarinen.A, Siekkinen.M, and Xiao.Y. 2012. Can offloading save energy for popular apps?, MobiArch'12, 3-10.

[23] Lin.K, Kansal.A, Lymberopoulos.D, and Zhao.F. 2010. Energy accuracy trade-off for continuous mobile device location. MobiSys'10, 285–298.

[24] Priyantha.B, Lymberopoulos.D, and Liu.J. 2011. LittleRock: Enabling Energy Efficient Continuous Sensing on Mobile Phones. IEEE Pervasive Computing Magazine, volume 10, issue 2, 12-15.

[25] http://www.seas.harvard.edu/news-events/press-releases/gates-grant/?searchterm=MFC. Last visited: March 29, 2013.

[26] American Chemical Society press. Mar 2011.http://portal.acs.org/portal/acs/corg/content?_nfpb=true&_pageLabel=PP_ARTICLEMAIN&node_id=222&content_id=CNBP_026949&use_sec=true&sec_url_var=region1&__uuid=3d60a60f-1722-46c3-b6a3-d08cf4b7e46. Last visited: March 29, 2013.