History-based Dynamic Estimation of Energy Consumption for Mobile Applications

Kwang-Ho Lim, Byoung-Dai Lee*

Department Of Computer Science, Kyonggi University, Suwon Korea

khlim@kyonggi.ac.kr, blee@kyonggi.ac.kr

Abstract— **Mobile cloud computing is one of the primary research areas that aims to overcome the problem of limited mobile device energy. For this purpose, computational tasks of mobile applications that consume a high amount of energy are offloaded to the cloud. In order to perform effective computation offloading, the energy consumed by application programs in mobile devices and the cloud needs to be estimated first. Previous studies have proven that energy consumption estimators showed optimal performance with targeted mobile application programs. However, optimal performance has yet to be achieved with other mobile application programs. Thus, this paper proposes a dynamic energy consumption estimator in which a mobile application program selects an appropriate energy consumption estimator for its own benefit. In addition, performance and effectiveness of the proposed dynamic energy consumption estimator are validated through experiments.**

Keywords— **Mobile Cloud Computing, Mobile Application, Offloading Decision Making, Energy-Efficient, Energy Estimate**

I. INTRODUCTION

Mobile environments including mobile devices and mobile application programs have made astonishing advances. However, owing to the nature of the mobile environment, devices run on batteries, which is a limited energy source. Therefore, as mobile environments are dependent on energy consumption, it limits various services that can be provided in the mobile environment [1–2]. Various studies are under way to overcome this limit on energy [1–5]. Recently, aside from internal approaches to solving energy problems by improved battery technology or energy saving, new studies on mobile cloud computing investigating combinations of mobile devices and the cloud as an external resource have been attempted to improve energy efficiency and overcome resource limitations in mobile devices. In particular, a computation offloading technology that executes mobile application programs in the cloud and not in the mobile device has been highlighted as one of the main application areas of mobile cloud computing owing to the recent introduction of resource-intensive mobile application programs that require high performance specifications along with performance improvements of mobile devices. Computations involved in applications that require high energy consumption, or are of low importance can be offloaded to the mobile cloud computing environment for execution, thereby saving energy and reducing the execution time for mobile application programs. However, if the offloading cost to send jobs to the cloud is higher than the cost of execution when the jobs are processed in the mobile device, it increases execution time and energy consumption. Thus, offloading should be determined according to energy efficiency by analyzing the energy spent by mobile application programs in the mobile device and on the cloud.

Calculating the energy consumed by mobile application programs is highly complicated owing to various internal and external environment variables such as input from the user, internal logic, and the runtime environment (i.e., the maximum performance of the currently running CPU and usage); in some cases, execution is altogether impossible [4–5]. Thus, energy consumption is estimated based on the historical execution data. This method has been proven effective as seen in previous studies [5–6]. In addition, there already are various studies [7–10] on estimating and calculating energy consumption, and the suggested algorithms in each study show optimal performance for a specifically targeted mobile program. However, the downside is that this performance cannot be expected with mobile application programs other than the ones they target.

Thus, this study proposes a dynamic energy consumption estimator by which a mobile application program selects the appropriate energy estimator among various energy consumption estimators to overcome the limits of existing energy consumption estimators. Furthermore, an experiment was conducted to compare and analyze the estimation rate of the dynamic energy consumption estimator and other individual energy consumption estimators to validate the performance and effectiveness of the proposed dynamic energy consumption estimator.

II. RELATED WORK

From the energy consumption perspective, computation offloading is an energy-saving technology. Computation offloading is performed in the cloud if the energy consumed by offloading to the cloud is less that that consumed when programs are executed in the mobile device. Energy conservation studies are actively performed to evaluate such computation offloading.

In a study by Miettinen et al. [7], a method for calculating energy consumption was proposed using the workloads of

* Corresponding author

mobile application programs. In their study, if the workload of a mobile application program exceeded 1,000 cycles in the target mobile device, it is better to offload the computation to reduce energy consumption. In the framework proposed by Cuervo et al. [8], a 0-1 integer linear programming (ILP) problem was formalized in which applications programs were divided into units of methods, and execution location was considered as 0 or 1. They then multiplied it by the energy consumption of each method to produce the lowest total energy consumption. In the framework of Kovachev et al. [11], the size of the code was considered as the number of commands to calculate the workload, thereby calculating energy consumption based on that number. Since the aforementioned study and frameworks calculated the workload and energy consumption of programs by the number of CPU cycles or the size of the code, it was limited in calculating the workload and energy consumption of application programs where CPU cycles or executing commands dynamically changes according to inputted data. For example, an accurate workload for programs that perform repetitive operations according to a user's input cannot be predicted, and energy consumption is hard to know without knowing how many repetitions will be made in the design time.

Hong et al. [10] proposed a formula to calculate energy consumption according to the number of images based on image extraction time, database fetch time, and image search time in content-based image retrieval (CBIR). Therefore, the advantage of their method is that energy consumption can be calculated dynamically during runtime regardless of changes in the number of images. However, their solution only applies to calculating energy consumption for CBIR, and it cannot be applied to other mobile application programs.

Thus, this study proposes a dynamic energy consumption estimator that can select an energy consumption estimator suitable for mobile application programs among many estimators as a method to overcome the drawbacks of previous studies and frameworks.

III. Dynamic Estimation of Energy Consumption

The dynamic energy consumption estimator proposed in the present paper selects the suitable energy consumption estimator for mobile application programs among various types of energy consumption estimators, and can add or remove various types of estimators to support the above operations. The problem with this structure is that the energy consumption estimator has to be selected from among various energy consumption estimators. In this study, an energy consumption estimator with the lowest error rate was deemed to be the suitable energy consumption estimator to solve this problem.

Figure 1 shows pseudo code of the algorithm for selecting the suitable energy consumption estimator for a mobile application program with dynamic energy consumption estimator. Lines 7 to 12 shows the process of calculating the error rate of each energy consumption estimator. The error rate is obtained using the energy consumption estimated by each energy consumption estimator and the actual energy consumption as well as runtime history. Each error rate is calculated based on the formulas shown in Table 1. Lines 14–17 indicate how an energy consumption estimator finds the lowest error rate from several error rate formulas. Lines 19 and 20 shows the last step in which the most frequently selected energy consumption estimator is selected as suitable.

```
1: /* ECEs: An array of energy consumption estimators
2:    ERs: An array of Error Rate
3:    realConsumedEnergy: The actual energy consumed
4:    estimatedEnergyConsumption: The energy consumption estimated
5:    pastRuntimeHistory: Past run-time history
6: */
7: for i = 0 ; i < ECEs.length ; ++i
8:    estimatedEnergyConsumption = ECEs[i].estimate()
9:       for j = 0 ; j < ERs.length ; ++j
10:         errorRateList[j][i] = ERs[j].calculate(
                                  realConsumedEnergy,
                                  estimatedEnergyConsumption,
                                  pastRuntimeHistory)
11:      end for
12: end for
13:
14: for j = 0 ; j < ERs.length; ++j
15:    minIndex = min(errorRateList[j])
16:    selectedECElist[minIndex]++;
17: end for
18:
19: maxSelectedECEIndex = max(selectedECElist);
20: bestECE = ECEs[maxSelectedECEIndex]
```

Figure 1. Pseudo code of the algorithm for selecting the suitable energy consumption estimator

TABLE 1. Formula for error rate calculation

Name	Formula
MSE (Mean-squared error)	$\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}$
MAE (Mean-absolute error)	$\frac{\sum_{i=1}^{n}\lvert(y_i-\hat{y}_i)\rvert}{n}$
MSEE (MSE Extend)	$\frac{\sum_{i=1}^{\alpha}(y_i-\hat{y}_i)^2}{n}$
MAE (MAE Extend)	$\frac{\sum_{i=1}^{\alpha}\lvert(y_i-\hat{y}_i)\rvert}{n}$
RSE (Relative-squared error)	$\sum_{i=1}^{n}\frac{(y_i-\hat{y}_i)^2}{(y_i-\bar{y})^2}$
RAE (Relative-absolute error)	$\sum_{i=1}^{n}\frac{\lvert(y_i-\hat{y}_i)\rvert}{\lvert(y_i-\bar{y})\rvert}$
RSEE (RSE Extend)	$\sum_{i=1}^{\alpha}\frac{(y_i-\hat{y}_i)^2}{(y_i-\bar{y})^2}$
RAEE (RAE Extend)	$\sum_{i=1}^{\alpha}\frac{\lvert(y_i-\hat{y}_i)\rvert}{\lvert(y_i-\bar{y})\rvert}$

In this study, eight formulas are proposed to calculate an error rate for selection of the energy consumption estimator with the lowest error rate by the dynamic energy consumption estimator, which are shown in Table 1.

In the table, n is the total number of executed data in the past, α is the number of historical data that excludes 5% of the max/min data in the total executed data from the past, y is an actually measured value, $\hat{y}$ is an estimated value, and $\bar{y}$ is an average of measured values. All energy consumption estimators calculate error rates using the above formulas and select the energy consumption estimator with the lowest error rate per formula.

TABLE 2. EXAMPLE OF THE ALGORITHM FOR SELECTION OF AN ENERGY CONSUMPTION ESTIMATOR

	Estimator 1	**Estimator 2**	**Estimator 3**
MSE	12.66	12.77	13.14
MAE	12.21	12.33	12.11
MSEE	10.22	10.68	11.22
MAEE	11.01	12.28	10.55
RSE	3.34%	2.1%	2.48%
RAE	4.23%	2.71%	2.55%
RSEE	3.51%	1.81%	2.32%
RAEE	3.52%	2.76%	2.6%
Selected count	2 times	2 times	4 times

Table 2 shows an example of the algorithm operation that calculates an error rate of each energy consumption estimator according to the pseudo code in Figure 1, selects the energy consumption estimator with the lowest error rate, and finally shows how many selections each estimator received. In this example, Estimator 3 had the most selections at four. Thus, Estimator 3 is selected as the suitable energy consumption estimator.

IV. PERFORMANCE EVALUATION

A. Experiment Setup

In this paper, five energy consumption estimators were chosen to evaluate the performance of the proposed dynamic energy consumption estimator. To this end, a dynamic energy consumption estimator, in which the below five estimators are embedded, was used to estimate energy consumption in face detection, video transcoding, and a chess game, some of the popular programs in mobile applications, thereby comparing and analyzing the error rate of the estimation.

Five sets of a single energy consumption estimator were implemented based on the Waikato Environment for Knowledge Analysis (WEKA) [12], an open source project. WEKA is a set of machine learning algorithms for data mining, which includes methods for data pre-processing, prediction, classification, regression, clustering, and association rules. The five estimation algorithms were chosen in this study considering the mobile environment and frameworks. The characteristics of each energy consumption estimator are as follows:

- Linear Regression: Regression analysis is a mathematical tool used to estimate a single dependent variable out of a number of independent variables. In particular, linear regression is fast and has a low calculation cost. It is suitable for application programs with linear data distribution because it induces estimation values by finding approximate linearity using the method of least squares from data executed in the past.
- Decision Table: This is the most preferred classifier owing to easy implementation, fast speed, and high functionality, but it has low performance levels. It starts at the root node, answers a query, and moves to a child node according to the answer. This process is run recursively until it arrives at the last leaf node. It is suitable for application programs that perform estimation by real-time interaction with users owing to the above-mentioned operation process.
- LWL (Local Weighted Learning): This is an instance-based algorithm that estimates by assigning a probabilistic weight using the Naïve Bayes classifier, which is suitable for data types with normal distribution such as mail filtering or document classification.
- IBk: This is an instance-based algorithm that uses the KNN (K Nearest Neighbors) model. IBk is a method to measure similarity between data using the Euclidean distance. The Euclidean distance is the straight distance between two data and therefore does not consider weights or the influence of independent variables that they represent. Thus, it has a weakness of incorrect estimation if a unit or the influences of independence variables are different from one another.
- KStar: This is also an instance-based algorithm that uses the KNN model as in IBk. However, it measures the distance between data using the Entropy-based Distance Function that considers probabilistic similarity. Thus, it has the characteristic that the weight or units of independent variables are less influential on estimation.

B. Experimental Result

Because resolution and color depth affects image size, in race detection the independent variables are correlated, and therefore data are highly similar leading to good performances with IBk and KStar as shown in Figure 2. Good performance was achieved by selecting KStar, which showed the best prediction rates, as the dynamic energy consumption estimator.

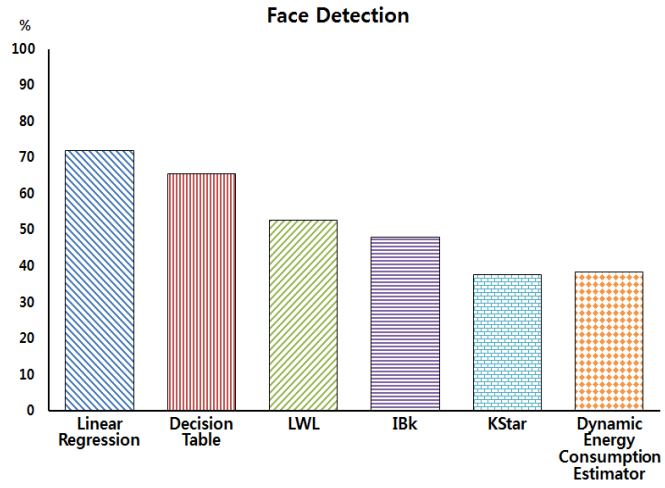

Figure 2. Average error rate of energy consumption estimation for face detection

Figure 3 shows the average error rate of energy consumption estimation for video transcoding. Due to the characteristic of video transcoding which has linear distribution in terms of energy consumption, linear regression as well as the dynamic energy consumption estimator using linear regression had the best estimation rate.

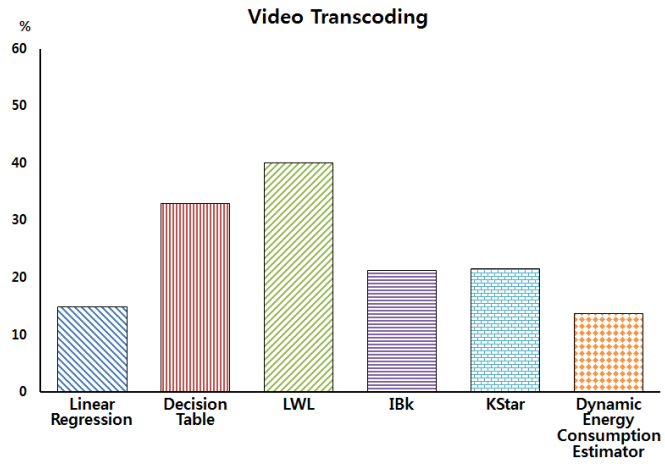

Figure 3. Average error rate of energy consumption estimation for video transcoding

Finally, Figure 4 shows the average error rate of energy consumption estimation for a chess game. In chess, values of independent variables required to determine the difficulty of the game are correlated, indicating similarity between the data. However, it has a highly complicated distribution because units or influences of independent variables are different from one another. KStar had the best estimation rate using the Entropy-based Distance Function to alleviate the effect of different influences and units between independent variables compared to other energy consumption estimators. Nonetheless, KStar did not show the best estimation rate in all applications we tested. However, dynamic energy consumption estimators showed the lowest error rate by selecting energy consumption estimators deemed suitable for all estimations.

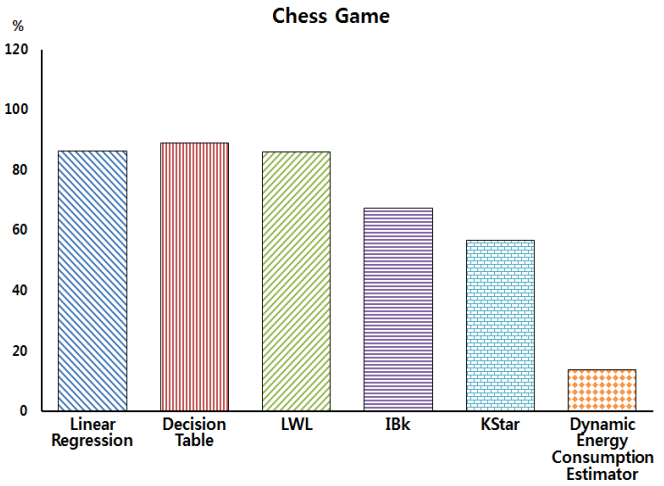

Figure 4. Average error rate of energy consumption estimation for a chess game

In conclusion, we validated the effectiveness of dynamic energy consumption estimators by experiments which showed that dynamic energy consumption estimators displayed good performance by dynamically selecting energy consumption estimators according to each mobile application program.

V. CONCLUSIONS

Computation offloading-based mobile application programs require estimation of energy consumption for offloading. The previously proposed energy consumption estimation methods showed suitable performance only for specific mobile application programs. Therefore, this study proposed a dynamic energy consumption estimator by which the most suitable energy consumption estimator for a mobile application program was selected and used dynamically from a number of energy consumption estimators to overcome the drawbacks of individual energy consumption estimators. This study also developed a selection algorithm by which the dynamic energy consumption estimator can select the suitable estimator suitable for a given mobile application program.

In order to validate the performance and effectiveness of the proposed dynamic energy consumption estimator, three mobile application programs used widely in mobile devices were implemented, and the energy consumption for execution of the applications was estimated to compare the performance between the dynamic energy consumption estimator and individual energy consumption estimators. Through the experiments, the dynamic energy consumption estimator showed high estimation accuracy and effectiveness as it selected the suitable energy consumption estimator for various mobile application programs.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2011-0008552).

REFERENCES

[1] Byoung-Dai Lee, "A Framework for Seamless Execution of Mobile Applications in the Cloud," Recent Advances in Computer Science and Information Engineering. Springer Berlin Heidelberg., vol. 3, pp. 145-153, 2012.

[2] Karthik Kumar and Yung-Hsiang Lu, "Cloud computing for mobile users: Can offloading computation save energy?," Computer., vol. 43, pp. 51-56. April. 2010.

[3] Fox Armando, et al, "Above the clouds: A Berkeley view of cloud computing," Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Tech. Rep. UCB/EECS 28 2009.

[4] Jacob R. Lorch and Alan Jay Smith, "Improving dynamic voltage scaling algorithms with PACE," SIGMETRICS/Performance., June. 2001.

[5] Wanghong Yuan and Klara Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," ACM SIGOPS Operating Systems Review., vol. 37, pp. 149-163, December. 2003.

[6] Dirk. Grunwald, Charles B. Morrey III, Philip Levis, Michael Neufeld, and Keith I. Farkas, "Policies for dynamic clock scheduling," Proceedings of the 4th conference on Symposium on Operating System Design & Implementation., vol. 4, pp. 6-6, 2000.

[7] Antti P. Miettinen, And Jukka K. Nurminen, "Energy efficiency of mobile clients in cloud computing," Proceedings of the 2nd USENIX conference on Hot topics in cloud computing., pp. 4-4, June. 2010.

[8] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl, "MAUI: making smartphones last longer with code offload.", ACM. Proceedings of the 8th international conference on Mobile systems, applications, and services., 2010, p 49-62.

[9] Changjiu Xian, Yung-Hsiang Lu, and Zhiyuan Li, "Adaptive computation offloading for energy conservation on battery-powered systems," Parallel and Distributed Systems. 2007 International Conference on. vol. 2. IEEE, 2007.

[10] Yu-Ju Hong, Karthik Kumar, and Yung-Hsiang Lu, "Energy efficient content-based image retrieval for mobile systems," Circuits and Systems. 2009. ISCAS 2009. IEEE International Symposium on., May. 2009,

[11] Kovachev. D, Tian Yu, and Klamma. R, "Adaptive Computation Offloading from Mobile Devices into the Cloud," Parallel and Distributed Processing with Applications (ISPA). 2012 IEEE 10th International Symposium on. IEEE, July 2012.

[12] Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, "WEKA---Experiences with a Java Open-Source Project," The Journal of Machine Learning Research 9999., vol. 11, pp. 2533-2541, March. 2010.

Kwang-Ho Lim is a master's degree at the department of Kyonggi University, Korea. His current research interest includes mobile cloud computing, open mobile platform, mobile application, and mobile framework. After earning his M.S., he plans to start software engineer.