A System Context-Aware Approach for Battery Lifetime Prediction in Smart Phones

Xia Zhao, Yao Guo, Qing Feng, and Xiangqun Chen
Key Laboratory of High Confidence Software Technologies (Ministry of Education)
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{zhaoxia,yaoguo,cherry}@sei.pku.edu.cn，fengqing@os.pku.edu.cn

ABSTRACT
Energy is a bottleneck in smart phone systems, and knowing the status of the battery lifetime and being able to use it efficiently is an important requirement from users. We propose a system context-aware approach for predicting battery lifetime, which allows a user to know the accurate battery status and to utilize the power efficiently. We refer to a collection of system component states as system context and model the quantitative relation between system context attributes and the battery discharge rate by multiple linear regressions. When the user changes applications or operations, we can dynamically predict the remaining battery lifetime as well as its variations by monitoring system context attributes. We implement the CABLI system with our approach as on an HTC G1 smart phone running the Android operating system. Experiments show that our model describes how the changes of system component states affect the battery lifetime, and that it improves the accuracy of online battery lifetime prediction.

Categories and Subject Descriptors
D.4. [**Operating Systems**]: Organization and Design – *Real-time systems and embedded systems*, Performance – *Modeling and prediction*

General Terms
Management, Measurement, Experimentation, Human Factors

Keywords
Smart Phone, battery lifetime, energy consumption, system context-aware.

1. INTRODUCTION
A smart phone has extended its functionalities beyond the traditional role of a "phone" and become a pervasive computing device. The always-on background applications increase the complexity of the system environment as well as the power consumption. For smart phone users, battery lifetime is one of the primary usability concerns. Knowing the status of the battery lifetime and using it efficiently is an important requirement from users.

Traditional solutions take the form of a battery indicator, informing users the remaining battery charge level with four to seven bars. However, it is hard for users to know how long the battery lasts if they perform a variety of tasks, and how their changes of operations affect the battery lifetime. If the operating system can provide more accurate and quantitative information about remaining battery energy and lifetime, then the users can adjust the operations to extend the battery life and enjoy more quality time.

Prior researches tried to monitor the battery energy level during the execution of some target applications and then to predict the battery lifetime based on the battery discharge measurements of a past period [#,1]. These approaches work well when the target application is the only running application in the system and has a constant workload. However, in a multi-process OS environment with background concurrent applications, it is difficult to identify which application should account for energy consumption in a certain period. Furthermore, the assumption that the future energy consumption was the same as the historical measurement is in general not valid.

In this paper, we propose a system context-aware approach for battery lifetime prediction in smart phone systems. We obtain the critical system components that affect energy consumption of a smart phone and refer to a collection of their states as system context. We then build a quantitative model for the system context and the battery discharge rate by using multiple linear regressions. In addition, we monitor the system context and dynamically predict the remaining battery lifetime and its variation.

We implement this approach on an HTC G1 smart phone running the Android operating system. In our experiments, we analyze the prediction efficiency of the model and compare it with an existing approach [1]. The results show that our model describes how the changes of system component states affect the battery lifetime, and it improves the accuracy of online battery lifetime prediction.

We organize the rest of the paper as follows. Section 2 describes the related work. Section 3 presents the process of the approach and the system context-aware battery lifetime model. After presenting experimental results in Section 4, we conclude with Section 5.

2. RELATED WORK
There has been a lot of work on laptop battery lifetime prediction. Most of early researchers adopted electrochemical features to predict battery lifetime and optimize the energy usage [2, 3].

The battery lifetime research of mobile phones received a lot of attention over the years. Rahmati et al. studied human-battery interactions and improved the interaction between users and battery discharge of smart phones [4]. They pointed out that users need higher resolution battery indicators, which enable them to charge phones more conveniently. However, they did not discuss how to accurately predict battery lifetime.

In order to enhance user experiences of using smart phones, some researchers measured and analyzed energy consumption and

SAC'11, March 21–25, 2011, TaiChung, Taiwan.

battery lifetime under different applications and usage patterns [5][6][7]. They did not consider how the system components affect the battery lifetime and how to use this information to predict the battery lifetime.

Ravi et al. proposed a battery management approach for mobile phones [8]. They used a base curve and the discharge speedup factor to predict the battery life. However, their approach can only be applied to a given set of applications observed in advance. Our approach is applied to the entire smart phone systems, not limited to some special applications.

Wen et al. proposed an online approach for predicting battery lifetime [1]. They assumed that the future energy consumption is the same as the historical measurements. This approach works on various applications, but it has relatively large mean errors for variable workloads. In our method, the states of system components are more crucial for battery lifetime prediction. Our experiment results show that our approach performs with higher accuracy and provides users better usage experiences than their approach.

Shye et al. studied mobile architectures by a logger application that collected real user activity and the traces of power consumption [9]. They adopted a linear regression analysis method and system parameters similar with ours. But they didn't consider the relationship between the battery lifetime and the system component states. We model the quantitative relationship among the battery discharge rate, battery lifetime and the system component states. One of our distinguished contributions is that we applied the model to battery lifetime prediction, and achieved much more accuracy than the existing approaches. Their work confirms our finding that the system component states are the key and straightforward indicators for battery energy consumption and lifetime prediction.

3. SYSTEM CONTEXT-AWARE BATTERY LIFETIME PREDICTION

Through extensive profiling, we found that the changes of system component states are driven by applications, and that the system component states are good indicators for workloads in the system. The battery lifetime is affected by the summation of energy consumption of all system components. Energy consumption of a component depends on its power state (which can be mapped to the operation state) and the duration it remains in that state. Therefore, we can use regression analysis to quantify the relation between system component states and battery energy consumption.

3.1 System Context

In a mobile phone, the major energy consumers are the CPU, LCD backlight, and network interface [9, 10]. However, no existing work has conducted the quantitative analysis on the relation between these component states and battery discharge rate.

We analyze a large amount of profiling data quantitatively, and find out that there are approximately linear relation between the battery discharge rate and some system component state attributes (Table 1). For some components, we use their resource utilizations to describe their states because the resource utilizations more accurately express the workload intensiveness and can be mapped to energy consumption of the components. For example, we use average *CPU utilization* during a time interval to describe CPU state, and use *data transfer rate* to describe the state of the network interface.

We refer to a set of system component states as *system context* and treat each state as an attribute of the system context. We denote a system context and its attributes as a tuple $C(brt, cpu, wifi, io, spd)$. Table 1 shows the system context attributes we use in our current prediction model. The values of the context attributes vary according to differently running applications and user's operations.

Table 1. System Context Attributes

Attribute	Description & Range	Example
CPU Utilization (*cpu*)	The ratio between the idle time to the total time of a interval, [0-1]	0.2
LCD Backlight Brightness (*brt*)	Range from [30,255] in HTC G1	255
Wireless State (*wifi*)	Disable or Enable [0, 1]	0
IO Idle Rate (*io*)	IO idle rate during a time interval	0.006
Data Transfer rate(*spd*)	Volumes of data transferred (KB)	0

It is well known that intensive usage of components tends to reduce battery lifetime. However, in order to accurately predict the battery lifetime, we need to present the quantitative relation between the system context and the battery discharge rate.

3.2 The Process of Our Approach

Our approach for battery lifetime prediction consists of two stages: modeling and predicting. Figure 1 shows the process of this approach.

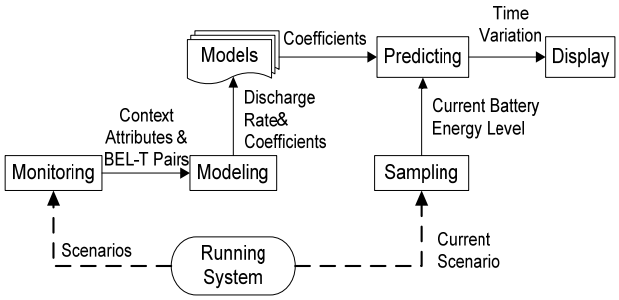
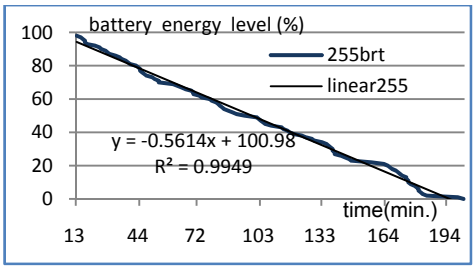


Figure 1. Process of dynamic battery lifetime prediction

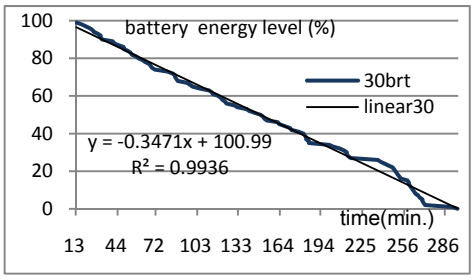
During the modeling stage, we custom specific scenarios that have stable system component states for a whole battery lifetime duration. For example, we run a video player application that plays a movie with the maximum LCD backlight brightness value 255. For the whole battery lifetime, the average *CPU utilization*, *wifi*, *io,* and *spd* are all approximately stable, as listed in the third volume in Table 1**.**

Then we profile the battery energy level vs. time under this scenario during the whole battery lifetime by using the API provided by the Java framework and the operating system. Figure 2(a) shows the battery discharge curves and battery lifetime of the *VideoPlayer* application under this scenario. The battery energy level is in terms of percentage. The slope of the fitted line of the curve is the discharge rate, which means the battery energy level decreasement per minute. In this example, its absolute value on average is 0.5174.

If we change the brightness to another value (for HTC G1, this value ranges between 30 and 255), such as 30, while keeping other component states fixed, we can get another discharge rate 0.3417 shown in Figure 2(b).



(a)



(b)

Figure 2. Battery discharge curve and battery lifetime of VideoPlayer application with the brightness of 255 and 30.

In this manner, we collect a series of discharge rates under different system contexts. The Table 2 shows a sample list of system context attributes and the battery discharge rate.

Table 2. A sample list of system context attributes and battery discharge rate

(*brt, cpu, wifi, io, spd*)	Battery Discharge Rate
255, 0.2, 0, 0.008, 0	0.56
192, 0.2, 0, 0, 0.008, 0	0.47
80, 0.2, 0, 0, 0.008, 0	0.41
30, 0.2, 0, 0, 0.008, 0	0.35
255, 1, 0, 0, 0	0.61
255, 0.91, 0, 0, 0	0.58
255, 0.73, 0, 0, 0	0.55
255, 0.63, 0, 0, 0	0.52
255, 0.38, 0, 0, 0	0.50
255, 0.3, 0, 0, 0	0.48
255, 0.2, 0, 0, 0	0.47
255, 0.1, 0, 0, 0	0.46
255, 0.06, 1, 0, 40	0.69
255, 0.04, 1, 0, 30	0.67
255, 0.03, 1, 0, 20	0.66
255, 0.02, 1, 0, 10	0.65

With the collected data, we conduct multiple linear regressions and achieve a quantitative model to describe how the discharge rate of the battery changes along with system component states. We save the model as coefficient sets. The modeling work is a one-time work for a smart phone battery, and the results built from the data can be used again for a long period before the battery ages.

In the predicting phase, we monitor the battery energy level and system component states under the current application scenario. Then, we use the model coefficients and the monitored attributes to compute the discharge rate of the battery. With the discharge rate and the current battery energy level, we can predict the remaining battery lifetime. Furthermore, by calculating the difference of the current and the last prediction, we can tell the variation of predicted battery lifetime caused by the changes of the system component states.

3.3 Regressions of Discharge Rate

In order to predict the remaining battery lifetime, we first need to estimate the discharge rate by using the system context attributes. As shown in Figure 2, the discharge rate is the energy consumption rate of the system. We build a quantitative model of system context and discharge rate by using multiple linear regression analysis.

We take system context attributes as independent variables and battery discharge rate as dependent variables. We use the multi-linear regression model shown in equation 1 to describe their relationship.

$$\mathbf{A} = \mathbf{cX} + \varepsilon \tag{1}$$

where,

$$\mathbf{A}=\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix},\ \mathbf{X}=\begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ 1 & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{pmatrix},\ \mathbf{c}=\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_k \end{pmatrix},\ and\ \boldsymbol{\varepsilon}=\begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

A is an $(n \times 1)$ dependent variable vector representing the absolute battery discharge rate, where n is the number of discharge rates. **X** is an $(n \times k)$ matrix of attribute values, where k is the number of attributes. **c** is a $(k \times 1)$ vector of regression coefficients. ε is an $(n \times 1)$ vector of random errors, and they account for derivations of the actual data from the predicted values. We usually think of ε as a statistical error and assume that it is normally distributed with mean zero and variance σ^2, abbreviated as $N(0,\sigma^2)$.

In order to find out how the system context attributes affect the prediction accuracy and identify the most proper set of system context attributes, we adapt a model family of six equations, which includes different attributes selected from (*brt, cpu, wifi, io, spd)*. Table 3 lists the names of the equations and their descriptions. The *dr1* equation only considers the LCD brightness, and other system contexts are treated as constants. The *dr2* equation only considers CPU utilization. The *dr3* equation considers the LCD brightness and CPU utilization. The subsequent equations incorporate more and more system context attributes in the model.

Table 3. Different System Context Attributes

Equation	*k*	*brt*	*cpu*	*wifi*	*io*	*spd*
dr2brt	1	√				
dr2cpu	1		√			
dr3	2	√	√			
dr4	3	√	√	√		
dr5	4	√	√	√	√	
dr6	5	√	√	√	√	√

The experiment results presented in the next section illustrate that different combinations of system context attributes result in different prediction accuracy, and there is a trade-off between the complexity and the prediction accuracy of the model.

We put the absolute values of battery discharge rate and system context attributes in the matrix and use the method of least squares to compute the coefficients. With different combinations of context attributes, we get different equations by the regression model. They are shown as a set of equations in formula 2.

$$
\begin{aligned}
a_1 &= 0.326 + 0.0008 * brt \\
a_2 &= 0.502 + 0.056 * cpu \\
a_3 &= 0.324 + 0.0006 * brt + 0.125 * cpu \\
a_4 &= 0.324 + 0.0006 * brt + 0.125 * cpu + 0.208 * wifi \\
a_5 &= 0.252 + 0.0008 * brt + 0.118 * cpu + 0.22 * wifi + 7.090 * io \\
a_6 &= 0.252 + 0.0008 * brt + 0.118 * cpu + 0.159 * wifi + 7.090 * io + 0.0017 * spd
\end{aligned} \quad (2)
$$

The above equations describe how the changes of system context attributes affect the variations of battery discharge rate and the battery lifetime. For example, suppose that the current battery energy level is 86. If a user changes the LCD backlight brightness value from 128 to 192 while keeping other contexts fixed, then with the *dr6* equation we calculate that the absolute discharge rate changes from 0.43 to 0.49. In addition, we can predict that the battery lifetime will shrink from 200 minutes to 175 minutes. This gives the user quantitative information about the change of the battery lifetime and the impacts of his operations on the battery lifetime.

3.4 Discharge Rate-Based Battery Lifetime Prediction

In this part, we will present how to predict the battery lifetime by using the estimated discharge rate. Refer to the battery discharge curve in Figure 2, we use formula 3 to describe the relationship between the battery energy level and the remaining time:

$$v = F_c(t) \quad (3)$$

where, v is the battery energy level in terms of percentage, c represents the context tuple, and t is the time in minutes. We write the linear regression function of the battery discharge curve under the context tuple c as in formula 4:

$$v = \beta_c - \alpha_c * t \quad (4)$$

where, β_c and (- α_c) are the intercept and slope of the trend line, respectively. In order to illustrate the prediction model more clearly, we plot the line with variable names in Figure 3. (α_c) measures the change in the mean of v for a unit change in t, which is the *discharge rate* of the battery.

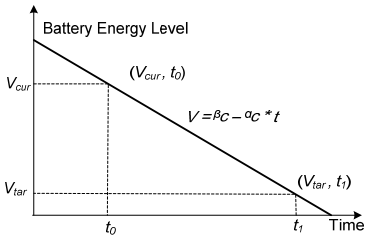


Figure 3. A trend line of a battery discharge curve

Suppose at the time t_0, the battery energy level is v_{cur}, and the target battery energy level is v_{tar} at t_1. Then, we calculate the battery lifetime from t_0 to t_1 by the formula 5.

$$T(v_{cur}, v_{tar}) = t_1 - t_0 = \frac{(v_{cur} - v_{tar})}{\alpha_c} \quad (5)$$

4. EXPERIMENTS AND EVALUATION

We implement the CABLI system using our approach in an HTC G1 smart phone running the Android operating system (Linux kernel 2.6.27). The phone has a 528MHZ Qualcomm MSM7201A ARM11 processor and 256MB flash memory. It is equipped with an 1150mAh/3.7V lithium-ion battery, and the capacity of the battery is 15318mJ in terms of energy**.**

We develop a set of tools in the CABLI system. A system context monitor service tool collects the profiling data in the running system. A modeling tool analyzes the data and achieves the coefficients of the model. A battery lifetime indicator monitors the system contexts and predicts the battery lifetime online by using the prepared model coefficients.

In order to achieve good results for the regression model, we collect a large amount of data from more than 40 different test scenarios. We select 16 group samples to build the model and use other data to evaluate the model. These samples are collected under the scenarios with stable system component states, which are CPU utilization, LCD backlight brightness, WiFi state, I/O idle rate, and network data transfer rate.

We use the benchmarks listed in Table 4 to test the prediction accuracy**.** For example, *VideoPlayer* provided by HTC G1 produces approximately constant workload; *Simulate-2* written by ourselves produces variable CPU utilization which we can control. Furthermore, we randomly run applications, such as *settings, contacts, notes,* etc., to produce variable workload, and name these scenarios as *MiscOpera*.

Table 4. Benchmark Descriptions

Benchmark	Workload	Description
VideoPlayer	Constant	An video player Java program
Ping	Variable	An operating system native utility, sends data to the server by WiFi
QuickSort	Constant	A sorting Java program with quick sort algorithm
Dijkstra	Constant	A graph search Java program that solves the single-source shortest path
BubbleSort	Constant	A sorting Java program with bubble sort algorithm
MiscOpera	Variable	Miscellaneous and applications, such as settings, contacts, notes
Simulate-1	Constant	A CPU-intensive Java program with constant CPU utilization
Simulate-2	Variable	A CPU-intensive Java program with variable CPU utilization

In the following sections, we first show an analysis on the energy consumption distributions of the smart phone. Then we evaluate the fitness of the regression model. Last, we present the comparison of our approach with a previous approach on efficiency and performance.

4.1 Energy consumption distributions

As described above, the battery discharge rate expresses the battery energy dissipation by the smart phone in the unit time. Based on the regressive model of the battery discharge rate, we

can understand the contribution of each system component to the discharge rate under a given certain system context.

We select four examples that are listed in Table 5, and analyze energy consumption distributions of the system components under these system contexts.

Table 5. System component states

Context Number	*brt*	*cpu*	*wifi*	*spd*	*io*
No.1	255	0.06	1	40	0.008
No.2	192	0.2	0	0	0.008
No.3	255	1	1	40	0.008
No.4	255	0.2	1	0	0.007

As shown in Figure 4, the energy consumption distributions of the system components are direct proportional to their states and resource utilizations, which is in conformity with our common sense. For example, the context No.1 has the same *brt, wifi, io* and *spd* attribute values with context No.3, but its *cpu* utilization is 0.6, which in less than that of No.3. Then, the CPU energy consumption of No.1 is less than that of No.3 by about 14%. This demonstrates that our model can be used to evaluate how the component states affect energy consumption of the smart phone system.

Our model also can be used to support the online dynamic power management. For example, during the system execution, we monitor the system component states and estimate energy consumption distributions of system components, and then we can adjust the workloads or states of the system components online to make trade-off between the performance and energy consumption.

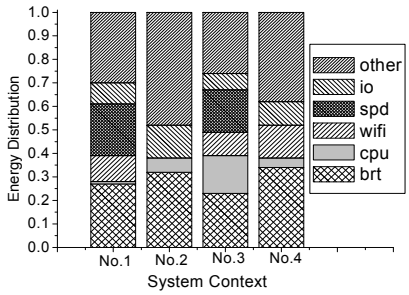


Figure 4. Energy consumption distributions of the system components

4.2 Model Evaluation

We evaluate the fitness of the prediction model by using residuals and prediction errors of the battery discharge rate. The difference between the sampled value which is used to build model and the estimated value is called a residual. The prediction error is the difference between the observed value and the estimated value. The residuals and prediction errors of the discharge rate are shown in Figure 5.

From the results, we can find that, the mean residual of *dr6* is less than 0.2%, and the mean prediction error is less than 1%. The *dr2brt* model has the worst absolute residual of not more than 2% and the *dr2cpu* has the worst absolute error of not more than 3%. This means that the model with context attributes including LCD backlight, CPU utilization, WiFi state, I/O idle rate and network data transfer rate performs the best, and that the too few system context attributes affect the performance of the regression model.

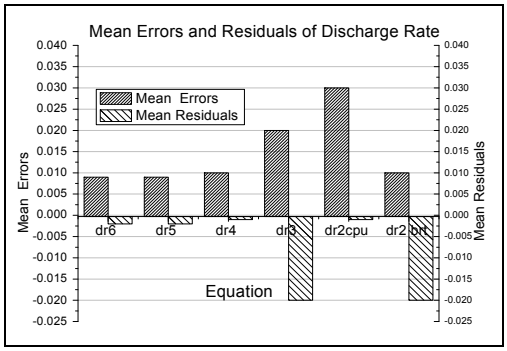


Figure 5. Prediction errors and residuals of discharge rate

4.3 A Comparison with Wen's Approach

First, we compare Wen's approach [1] (denoted as HBI) with ours (denoted as CABLI). We execute the above benchmarks and monitor the battery energy level vs. time. We predict the battery lifetime with two different approaches at every battery energy level and get the prediction errors. Because of the limited space, we take a *VideoPlayer* scenario and a *ping* wireless data transfer scenario as examples to illustrate the efficiency of the approaches. From the results in Figure 6, we find that the prediction error of HBI is about -35%~55%, while that of CABLI is only about -10% ~10%. The reason is that HBI assumes that the future battery power drainage tends to be the same with the history. In contrast, CABLI thinks that the system components are the major power consumers and predicts the battery power drainage based on the current values of their states. Therefore, it can reflect the variation of the remaining battery lifetime more accurately.

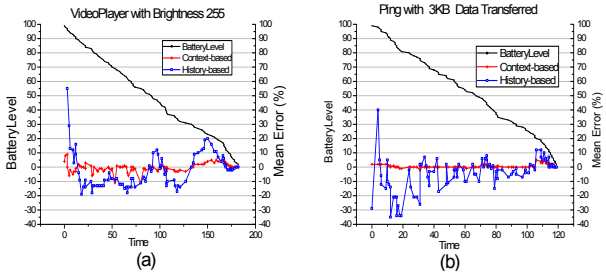


Figure 6. Prediction errors of the approaches.

4.4 Prediction Errors of the Equations

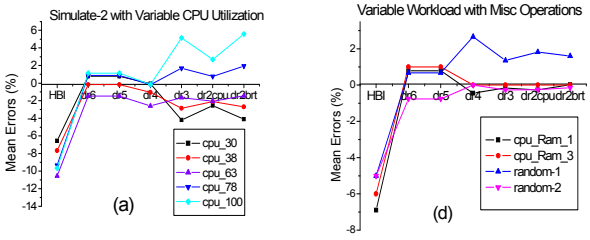


Figure 7. Mean prediction errors of different equations

In CABLI, different model equations represent different complexity levels of the model. We compare the prediction results

of battery lifetime obtained by different equations. In Figure 7, we take *Simulate-2* and M*iscOpera* as the examples. The results show that, the mean error of *dr6* is the smallest and is within -2% ~ 2%, and *dr6* performs best among the six equations.

4.5 Prediction Errors of Variations of the Battery Lifetime for Changed Workloads

When a user changes the workload, which approach can accurately notify the user the variations of the battery lifetime caused by the changes? In order to answer this question, we compare the prediction errors of variations of the battery lifetime given by the two approaches.

In Figure 8, we present the errors and the percentage errors of the predicted changes under five scenarios. The labels "255->80", "255->30","30->255" represent three scenarios of the *VideoPlayer* application. Under these scenarios, the user changes backlight brightness from 255 to 80, 255 to 30, and 30 to 255, respectively. The label "Video->sort" represents a scenario, under which, the user changes *VideoPlayer* to *Quicksort*. The label "cpu0.8->0.5" represents a scenario of the *simulate-2* program, under which, the user changes CPU utilization from 0.8 to 0.5. From the Figure 8, we can find that the maximum percentage error of CABLI is less than 6%, while that of HBI is close to 40%. CABLI predicts the battery lifetime changes more accurately than HBI, and performs better than HBI.

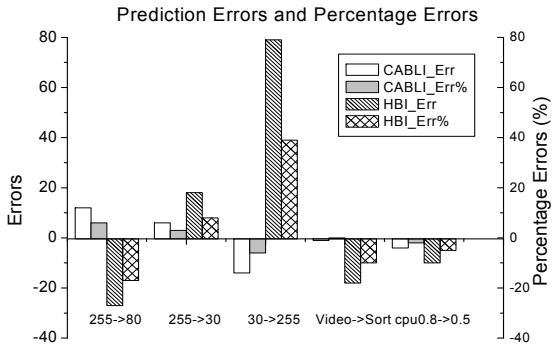


Figure 8. Prediction Errors and Percentage Errors under changed workloads

5. CONCLUSION

We propose a system context-aware approach for online battery lifetime prediction. We uses multiple linear regressions to build a quantitative battery lifetime prediction model for smart phones. The model describes how the changes of system context attributes affect the variations of energy consumption and battery lifetime. Using this approach, we dynamically predict the remaining battery lifetime based on monitored system context attributes. We implement our approach in the HTC G1 smart phone running the Android operating system. Experiments show that our approach predicts battery lifetime with higher accuracy than prior works. The accurate prediction of remaining battery lifetime can provides smart phone users with better usage experiences.

6. ACKNOWLEDGEMENT

This work was supported by the National High Technology Development Program of China (863) under Grant No. 2008AA01Z133, the National Basic Research Program of China (973) under Grant No. 2009CB320703, the Science Fund for Creative Research Groups of China under Grant No. 60821003, and the China Postdoctoral Science Foundation under Grant No. 20090450234.

7. REFERENCES

[1]Y. Wen, R. Wolski, C. Krintz, and R. Krintz, "Online Prediction of Battery Lifetime for Embedded and Mobile Devices," in *Issue on Embedded Systems: Springer-Verlag Heidelberg Lecture Notes in Computer Science*, 2004, p. 2004.

[2]D. U. Sauer and H. Wenzl, "Comparison of different approaches for lifetime prediction of electrochemical systems--Using lead-acid batteries as example," *Journal of Power Sources,* vol. 176, pp. 534 - 546, 2008.

[3]L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Discrete-time battery models for system-level low-power design," *IEEE Trans. Very Large Scale Integr. Syst.,* vol. 9, pp. 630--640, 2001.

[4]A. Rahmati, A. Qian and L. Zhong, "Understanding human-battery interaction on mobile phones," in *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, New York, NY, USA, 2007, pp. 265--272.

[5]A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX Annual Technical Conference*, Boston, MA, USA, 2010.

[6]J. Kang, C. Park, S. Seo, M. Choi, and J. W. Hong, "User-Centric Prediction for Battery Lifetime of Mobile Devices," in *APNOMS '08: Proceedings of the 11th Asia-Pacific Symposium on Network Operations and Management*, Berlin, Heidelberg, 2008, pp. 531--534.

[7]N. Balasubramanian, A. Balasubramanian and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, New York, NY, USA, 2009, pp. 280--293.

[8]N. Ravi, J. Scott, L. Han, and L. Iftode, "Context-aware Battery Management for Mobile Phones," in *PERCOM '08: Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, Washington, DC, USA, 2008, pp. 224--233.

[9]A. Shye, B. Scholbrock and G. Memik, "Into the wild: Studying real user activity patterns to guide power optimization for mobile architectures," in *Proceedings of the International Symposium on Microarchitecture (MICRO 2009)*, 2009.

[10]H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services*, New York, NY, USA, 2010, pp. 179--194.