Optimising Energy Consumption Heuristically on Android Mobile Phones

Mahmoud Bokhari
Computer Science School
University of Adelaide
Adelaide, Australia

Computer Science Department
Taibah University
Medina, Kingdom of Saudi Arabia

Markus Wagner
Optimisation and Logistics,
University of Adelaide
Adelaide, Australia

ABSTRACT

In this paper we outline our proposed framework for optimising energy consumption on Android mobile phones. To model the power usage, we use an event-based modelling technique. The internal battery fuel gauge chip is used to measure the amount of energy being consumed and accordingly the model is built on. We use the model to estimate components' energy usages. In addition, we propose the use of evolutionary computations to prolong the battery life. This can be achieved by using the power consumption model as a fitness function, re-configuring the smartphone's default settings and modifying existing code of applications.

CCS Concepts

•Computer systems organization → Embedded systems; •Computing methodologies → Heuristic function construction; Randomized search;

Keywords

Power Consumption Modelling, Energy Optimisation, Genetic Improvement, Search Based Software Engineering.

1. INTRODUCTION

Cellphone usage has grown significantly in the recent years and has expanded from only voice services to other sophisticated services such as social networking, entertainment and education. Despite the powerful capabilities of the current smartphones, their availability relies on the limited battery life. A smartphone needs a collection of applications that use its hardware to provide such services to its users. In order to satisfy a user's needs, applications need to run semi-constantly in the background or utilise hardware components frequently. In addition, the running applications might have energy bugs that intensely affect the energy consumption [8][9]. As a result, the operation time reduces notably which is a usability issue.

At the present time, it seems that engineers are incapable of increasing the amount of energy created by the chemical reactions in relatively small size batteries. Apparently, enlarging the battery size is the only way to increase its capacity. Nevertheless, this conflicts with the evolution of smartphones, where lighter and thinner devices are more desirable. It is worth mentioning that nanotechnology is being explored to build batteries that are able to generate ten times the amount of energy of the current lithium-ion batteries [2]. To the best of our knowledge, the mobile industry has yet to adopt this new technology.

An effective software solution starts with estimating the factors that affect the battery life which can be achieved through: measuring the current and voltage for particular time periods; generating models using power measurements to identify the factors that influence energy usage; and estimating the amount of energy used by the components using the generated model. External power and smartphones' built-in meters can be used to measure the current and voltage. The former method requires opening the device and connecting the meter to the battery interface. Building a model using this method restricts its accuracy to laboratory conditions. Using the built-in meter allows constructing models that can be applied under varying conditions.

Modelling software energy consumption is well studied in existing literature. Some works use utilisation-based energy modelling [4, 7, 11], where the main assumption of energy usage is correlated to the utilisation of a hardware component (e.g., GPS, disk and NIC). Any change of utilisation causes the power state change of the component. However, these models are deficient in capturing and modelling non-utilisation behaviours such as opening a file and tail states, which may last several seconds [10].

To overcome these issues, event-based modelling can be used. It correlates the energy usage to events that trigger power state changes of a component. For example, the authors of [10] represent the system as finite state machine (FSM) and associate a fixed energy consumption cost to each state. In addition, they utilise system calls to trace which state a component is in to compute its power usage. Parameters within system calls include the application name, the requested hardware component and the level of utilisation help to obtain fine-grained energy consumption estimation.

In order to prolong the battery life, the default settings of the smartphone have to be optimised. One solution is re-configuring the mobile phone's settings dynamically depending on the user usage behaviour. This method requires exploring a massive number of potential solutions as each smartphone consists of several components and each component has a number of settings. As a result, conventional

GECCO'16 Companion, July 20-24, 2016, Denver, CO, USA

DOI: http://dx.doi.org/10.1145/2908961.2931691

software engineering methods are insufficient to solve such a problem [3]. Forming this problem into an optimisation problem makes it suitable for genetic algorithms (GA) [3].

Besides parameter tuning using GAs, structural optimisations are sought to reduce high energy usage of an application. The authors of [6] suggest employing genetic improvement (GI) as a solution technique to modify an existing source code. For example, the work in [1] applies GI to the MiniSAT solver for a specific problem domain, which eventually consumes 25% less energy. They run the experiment on an Apple laptop and use the Intel Power Gadget API to estimate energy consumption. In contrast to this, we use our own model for estimation purposes, and more importantly our targeted optimisation platform is hand-held devices, where the available computatio power is limited.

2. POWER PROFILING

The proposed power profiler is based on event-based power modelling and internal power measurement techniques. We adopt the modelling methodology in [10] to avoid the mentioned issues related to the utilisation-based modelling. However, our method of constructing the system's FSM is different. Unlike the work in [10] where the FSM is pre-built for the tested devices, our profiler collects information regarding the hosting mobile's components as a first step. It then examines each component to derive its states.

To assign power usage values to power states, we use mobile phone self-monitoring capabilities. Modern smartphones are equipped with battery fuel gauge chips that report the voltage, current and remaining energy within the battery. Accessing these values can be done through the battery API, such as Android's BatteryManager class. The API periodically broadcasts these values. Using the internal measurement obviates the need of an external hardware power meter which depends on laboratory conditions. In addition, it allows dynamic model generation using information obtained from the device.

We use Android studio version 1.5 to build the profiler. It uses the BatteryManager API to collect power measurements from the battery fuel gauge. Initially, the target device for our experiments is HTC Nexus 9. It is equipped with the Maxim MAX17050 fuel gauge.

3. GENETIC ALGORITHMS

Since the issue of optimising the power consumption depends on each user's preferences (e.g., see [5, 11]), we propose a methodology that dynamically adjusts mobile configurations using GA and collected usage data.

Our framework keeps track of the user's usage behaviour. It records usage data for a pre-defined period of time. It then analyses the collected data to extract operational patterns. The goal of this step is to identify and rank the essential set of objectives to be used in the optimisation process. Ranking an objective is based on the frequency and/or the duration of the pattern that is represented by that objective.

In our optimisation approach each solution is represented as a set of settings for each component and the fitness function is to minimise the required energy for the desirable objectives. For instance, a solution contains the screen brightness level, GPS mode, network mode, and synchronisation frequency for applications such as the Gmail app. A predefined power consumption value is assigned to each setting and therefore the sum of these values represents the quality of the solution. A fitter solution within a population consumes less power in total.

4. GENETIC IMPROVEMENT

We use GI to optimise existing applications. We develop a test suite that has several energy-demanding applications. The GI runs on the source code of each test case to generate a new version of the original test case. In the next step, the framework compares the energy consumption of the modified test case, either through an actual execution or through our power model. Indeed, a fitter solution consumes less power. To preserve the functionality of the main software, its source code is used as a test oracle [1].

5. REFERENCES

[1] B. R. Bruce, J. Petke, and M. Harman. Reducing energy consumption using genetic improvement. In *Genetic and Evolutionary Computation Conference*, pages 1327–1334. ACM, 2015.

[2] C. K. Chan, H. Peng, G. Liu, K. McIlwrath, X. F. Zhang, R. A. Huggins, and Y. Cui. High-performance lithium battery anodes using silicon nanowires. *Nature Nanotechnology*, 3(1):31–35, 2008.

[3] J. Clarke, J. J. Dolado, M. Harman, R. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd. Reformulating software engineering as a search problem. *IEE Proceedings - Software*, 150(3):161–175, 2003.

[4] M. Dong and L. Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Mobile Systems, Applications, and Services*, pages 335–348. ACM, 2011.

[5] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.

[6] M. Harman, Y. Jia, W. B. Langdon, J. Petke, I. H. Moghadam, S. Yoo, and F. Wu. Genetic improvement for adaptive software engineering (keynote). In *Software Engineering for Adaptive and Self-Managing Systems*, pages 1–4. ACM, 2014.

[7] R. Murmuria, J. Medsger, A. Stavrou, and J. M. Voas. Mobile application and device power usage measurements. In *Software Security and Reliability*, pages 147–156. IEEE, 2012.

[8] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. In *Embedded Networked Sensor Systems*, pages 10:1–10:14. ACM, 2013.

[9] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with Eprof. In *Europ. Conf. on Computer Systems*, pages 29–42. ACM, 2012.

[10] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *European Conference on Computer Systems*, pages 153–168. ACM, 2011.

[11] A. Shye, B. Scholbrock, and G. Memik. Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures. In *Symposium on Microarchitecture*, pages 168–178. ACM, 2009.