Sustainable Computing: Informatics and Systems 7 (2015) 11–23

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

A study of energy-aware implementation techniques: Redistribution of computational jobs in mobile apps

Luis Corral, Anton B. Georgiev*, Alberto Sillitti, Giancarlo Succi

Free University of Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy

ARTICLE INFO

Article history:
Received 28 February 2014
Received in revised form 5 November 2014
Accepted 16 November 2014

Keywords:
Android
Energy
Green
Mobile
Offloading
Reallocation

ABSTRACT

As devices like smartphones and tablets have been adopted by millions of users, the offer of mobile software products and services has as well evolved significantly. Current mobile software products require hardware resources, computer job and network connectivity that is reflected in increasing power needs. This power demand represents a major challenge to the strong autonomy requirement of mobile devices, powered by batteries. As a consequence, one of the most important qualities of mobile software development is the ability to produce applications that consume energy resources wisely. To accomplish this goal, different approaches have been proposed, including energy-aware software design and implementation techniques. The energy aware software design techniques considered in this work are two: Method Reallocation, which refers to the placement of pieces of code in different execution scopes within a single target (e.g., kernel space, application space, shared library space), and Method Offloading, which refers to the placement of pieces of code in external resources in different scopes (for instance a remote server). Both techniques aim to economize resources like processing power, and memory usage, but always upon the expenses carried by interfacing, communication and network overhead. Our goal is to investigate how each one can contribute to reduce the overall energy consumption of a mobile software application. As an experiment, we utilized a mobile application that runs software benchmarks coded in Java and C. We exercised the benchmarks in different execution scopes within the handheld target and a remote server counterpart, measuring the amount of energy required to complete each job. After determining the energy consumed by each routine for each execution scope, we identified in what cases it is convenient to reallocate the processing job, and when it is advisable to offload it to an external execution environment.

1. Introduction

Energy consumption is understood as the amount of power that a device takes to carry out an operation. In the mobile execution environment, energy consumption is an important characteristic that developers and end users deeply care about. Mobile devices are carried by their owners, most of the time away from a permanent power source like an electrical power outlet. Current mobile software products keep up an increasing requirement of hardware resources, computer job and network connectivity that is reflected in increasing power demand. This need represents a major challenge to the strong autonomy requirement of mobile devices. Modern mobile applications take full advantage of the device's features like sensors, antennas and hardware components that, together with their strong computing capabilities require a higher amount of battery resources and trigger the need of recharging the device constantly. These conditions pose on developers an important concern: how to design and implement mobile applications that help the end user to accomplish his goals, but at the same time provide an efficient, energy aware operation that will not consume the energy resources of the target device.

The importance of an effective use of resources in a software system is considered both by developers and users as an important quality. In particular, in the mobile execution environment, the efficient utilization of resources, including energy consumption, represents a fundamental property and an imperative to evaluate the overall quality and usefulness of the final product [1].

In this paper, this work extends the results we got in the paper [2] analyzing the following data. Rerun the data collection, extends the data, we investigate the impact that the allocation of a method has in the overall energy consumption of a mobile application. We used two implementation techniques to distribute the

* Corresponding author. Tel.: +39 0471 016245.
E-mail addresses: luis.corral@unibz.it (L. Corral), anton.georgiev@unibz.it (A.B. Georgiev), alberto.sillitti@unibz.it (A. Sillitti), giancarlo.succi@unibz.it (G. Succi).

http://dx.doi.org/10.1016/j.suscom.2014.11.005
2210-5379/

computational job for energy efficiency: Method Reallocation and Method Offloading. These techniques aim to reduce the energy consumed by processing load in the mobile device by sending the tasks to an external target through a network. However, one should bear in mind that the use of interfacing devices (such as WiFi, 3G/2G antennas) may require a considerable energy toll, and that the payoff can turn into a backfire. This motivates the need of conducting an experiment that provides empirical evidence about the behavior of this situation in different settings, which require more or less computing power, and more or less network traffic. To this end, we implemented an experiment utilizing a mobile application that runs software benchmarks coded in Java and C. We exercised the benchmarks in different execution scopes, within the mobile target and using remote server counterpart, measuring the amount of energy required to complete each job. After determining the energy consumed by each routine for each execution scope, we identified in what cases it is convenient to reallocate the processing job, and when it is advisable to offload it to an external infrastructure. Our objective is to shed light on energy-efficient approaches for designing and implementing a mobile application.

The rest of this paper is organized as follows: Section 2 discusses diverse energy-aware software design techniques; Section 3 describes the selection and implementation of some energy-aware software design techniques in the form of an experiment on Android; Section 4 describes the methodology and environment created to carry out our experimentation; Section 5 presents the results obtained and the discussion on the results; Section 6 comments the related work; finally Section 7 identifies directions for future work and draws the conclusions.

2. Energy-aware software implementation techniques

The need of energy-efficient applications rises before the beginning of the smartphone era. The impact of the energy consumption has always affected mobile devices, but became a bigger issue with the introduction of modern mobile operating system such as iOS or Android, which promoted the utilization of hand held devices from simple communication and gaming features to more comprehensive computing applications. A vast variety of applications promoted a higher utilization of the mobile device and a higher demand of hardware features that take a significant toll from the battery reserves. Device manufacturers strive to progress in providing better battery technologies but it is difficult to keep up with the evolution of the computing capabilities and the demand of energy required to operate modern smartphones. From the software point of view, large and complex applications tend to stress heavily different components of the device like CPU, memory, storage, etc. There are different approaches to accomplish mature energy aware applications. From the research and practitioner point of view, several have taken the effort to lower the demand of energy, from a software point of view, in an effort to increase battery lifetime.

2.1. Energy aware frameworks

After surveying the literature we found frameworks for energy-aware mobile application design. They are based on the dynamic adaptation through an energy profiler module, energy policies and energy adaptation APIs [3]. The energy profiler module recognizes all the system states and estimates the energy demanded by an application; the energy policies module executes the energy adaptation invoking finally the energy adaptation APIs. With this implementation, when energy adaptation occurs, several parameters related to the quality of the service change. As consequence, this improvement compromises the software product in terms of quality, usability or user-friendliness. Another disadvantage is that they need to implement only an energy aware framework compatible with the developed application.

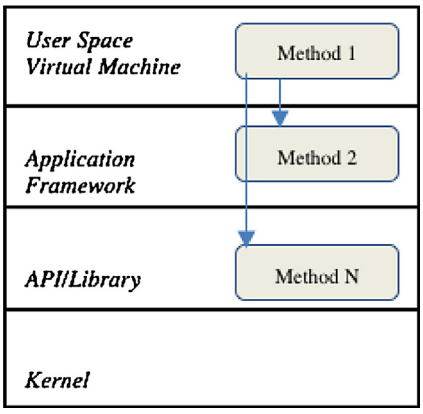

Fig. 1. Method reallocation.

2.2. Method reallocation

Another way of optimizing the energy invested to execute computational job is method reallocation [2]. Method reallocation refers to the analysis of a software product (i.e., an application) as means of a stack (for instance, kernel level, library level, API level, native code level, virtualized level, etc.) Having this structure facilitates the analysis of the energy costs associated to the components of each level of the stack (Fig. 1).

With this analysis at hand, it is recommended the reorganization of the classes and methods through the different levels of the software stack, in a way in which the classes and methods can be placed in the level where the energy consumption is minimal. As a limitation, this technique can be utilized only if the operating system and the software development environment allow developer to go through the different levels of the software stack.

2.3. Method offloading

Process offloading (also known as cyber foraging or remote execution) [4–7] is a technique in which heavy computation jobs are sent to a remote computer; after the end of the remote execution, the result is sent back to the local machine (Fig. 2).

The aim of offloading is to improve the performance of applications running under the low computational capabilities of laptops, moving the largest computations to more powerful surrogate computers. The application of the offloading technique became highly relevant with the advancement of smartphones aiming to extend their battery lifetime. This reorganization of the system aims to economize processor and memory, upon expenses of and network usage and communication overhead [8–13]. In addition, method offloading can be orchestrated by self-adapting algorithms that may decide whether to or not to offload a computational job.

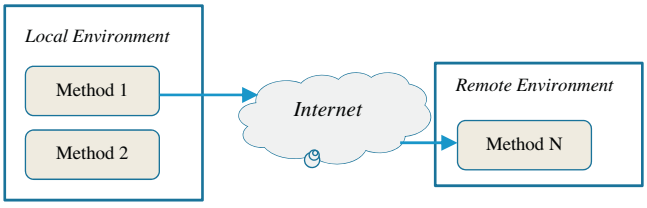

Fig. 2. Method offloading.