An analysis of power and performance of applications for mobile devices with Android OS

Andrws Vieira, Daniel Debastiani, Luciano Agostini, Felipe Marques, Júlio Mattos
{aavieira,dsdebastiani,agostini,felipem,julius}@inf.ufpel.edu.br

Group of Architectures and Integrated Circuits - GACI
Federal University of Pelotas - UFPEL

Abstract

Develop complex applications for mobile platforms can become trivial for the Android platform. With all of the facility offered to the developer, it is necessary to evaluate the performance, the execution time and the power as the application is dissipating. The focus of this work is the analysis of performance (execution time) and energy consumption of applications for Android in recursive and iterative algorithmic versions in the same computational complexity. The result of this analysis suggests the version of the algorithm that provides the best compromise between performance and power consumption for the platform.

1. Introduction

The cellphone market is rapidly developing. Studies show that today more than 3 billion consumers have a mobile phone, and this corresponds to more or less half of the world population [1]. In this context, the Android is recently on the spotlight for mobile devices developers, because of its software development kit (SDK), which enables a modern and agile development of applications and user interfaces that are visually elegant and easy to navigate. Android is a development platform for mobile applications based on Linux operating system [1] derived from an open source project led by Google. The fact the Android is an open source project, allied with its Apache 2.0 license, makes it flexible, allowing developers and companies to make customizations without need of sharing such changes. The Android application development is simplified by its SDK that provides tools and APIs needed to develop applications, favoring an easy integration with many resources available on the device. During the development of mobile applications, it is necessary to analyze non-functional requirements, such as performance, power consumption and energy, since these applications run on battery-based devices.

Performance analysis of an Android application can be done with Traceview [2], and power and energy consumption estimations can be obtained with PowerTutor [3]. Traceview is a graphical viewer for execution logs that you creates by using the Debug class to log tracing information in your code. Traceview can help you debug your application and profile its performance [2]. With the help of this tool, it is possible to find bottlenecks in an application, allowing developers to make modifications that increase the application's performance as much as possible. PowerTutor is an application for Google phones that displays the power consumed by major system components, such as CPU, network interface, display, GPS receiver and different applications. This application allows software developers to see the impacts of design changes on power efficiency. Application users can also use it to determine how their actions are impacting battery life. PowerTutor uses a power consumption model built by direct measurements during careful control of device power management states. This model generally provides power consumption estimates very close to actual values, with a 5% error margin. A configurable display for power consumption history is provided. It also provides users with a text-file based output containing detailed results. One can use PowerTutor to monitor the power consumption of any application [3].

The focus of this paper is to investigate applications for Android regarding the performance and power consumption in different versions of recursive and iterative algorithms for the same computational complexity. In doing so, it is possible to produce the best version of a given algorithm for a particular purpose.

2. State of the art and related works

Extending battery life in mobile devices has always been a challenge. Many researchers have as object of study methods of optimization and energy efficiency [4]. The complexity of modern devices is fast growing and such devices cannot be considered as simple devices. It is predicted that by 2013 mobile devices like smart-phones and tablets will overtake PCs as the most widely used devices for Internet access [5]. The most modern smart-phones are capable of running video and audio in high definition, providing access to high speed internet, allow pictures and videos of high quality - HD - besides having interface with sensors such as GPS and accelerometer.

The focus of this session is to describe and introduce systems that perform real-time estimates of energy consumption on mobile devices Android. PowerRunner and PowerTutor are results of research related to this work.

PowerRunner is automated management software that provides energy and optimizes energy consumption on mobile devices, using learning Linux kernels and a low-level API for power management at runtime [6]. Modern Linux kernels define a structure for the code to facilitate power management off and on an individual component of the system at runtime. Through careful monitoring and learning the behavior of the system components used by various user activities, PowerRunner analyzes and estimates the resources required by an activity on Android. In order to maximize the user experience, minimize power requirements and maximize energy savings by applying various combinations of optimizations at runtime. The project has proven to save up to 25% of energy consistently.

PowerTutor is a system for estimating energy consumption in real time implemented for the Android platform smart-phone. PowerTutor provides accurate estimates of energy consumption in real time to the hardware components, including CPU and LCD display, as well as GPS, Wi-Fi, audio and mobile network interfaces.

The main objective of this work is to use the PowerTutor to determine the impact of software design changes in energy consumption. This tool was chosen because it provides estimates of energy consumption by hardware components, allowing an analyze the behavior of energy consumption in different components of the device.

3. Description of Developed Work

3.1. Choice of Algorithms

In this work, it was used algorithms that have the same computational complexity in the iterative and recursive versions. The algorithms were implemented: InsertionSort and Factorial, which have the same asymptotic complexity as much as in the iterative version of recursive version, with O (n^2) and O (n) respectively according to [7]. Test cases were established for analysis of performance and power of these algorithms on the Android OS. In recursive versions with entries high values sometimes cause stack overflow in Android, complicating the analysis and comparison of results. To solve this problem, the method that executes the algorithms is called multiple times and the arithmetic mean of the results is calculated. For generating an input with large size to be ordered by the Insertion Sort algorithm, was described a script using the Python language [8], which generated random numbers positive integers in the range of zero to ten thousand. The same input was used to perform the recursive and iterative versions of the algorithm.

3.2. Evaluating Runtime with Traceview

From the implemented algorithms, it was necessary to import the library Debug and insert the source code of the application the method calls necessary for the implementation of Traceview. Apart from indicating the point of beginning and end of the trace, you must grant the application permission to record the trace on the SD Card. To make this possible, it is necessary to add the following line in the file AndroidManifest.xml, "<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />", giving permission for the program to write the data generated by Traceview tool in the external storage device of mobile [2]. After these steps, you run the command "pull" on the SD Card, so extracting the data, so that can be analyzed. This data contains information of how long each method or function call took to execute, these data are discussed in section four.

3.3. Measuring Power with PowerTutor

The power that an application/program dispels can be evaluated by applying PowerTutor that provides accurate, real-time power consumption estimates for power-intensive hardware components including CPU and LCD display as well as GPS, Wi-Fi, audio, and cellular interfaces, As can be seen in Figure 1 [9]. The data provided by the tool PowerTutor have a ratio of time, in seconds, of the dissipated power in milliwatts. These data sets were analyzed with the data extracted by the tool TraceView to analyze the power peaks that occur in a particular method or function call. These relationships are described in the following section.

Fig. 1 – PowerTutor running

4. Achieved Results

In both algorithms the execution time and power dissipation was higher in recursive algorithms, thus consuming more energy, but with some to highlighted to be make, especially in the factorial algorithm. Figure 2 demonstrates the behavior idle of the iterative factor to a high value input, although with some peak power greater than the recursive version, recursive noticed that the factor is maintained in practically all its entirety execution dissipating more power and a greater run time, taking approximately 11% more. From this result does not notice is a big difference between the algorithms, but it is worth highlighting that the recursive version of the maximum input size is limited to around 300 times smaller than the iterative version. Soon the iterative version of factorial is to be the best option to be implemented in applications for Android.

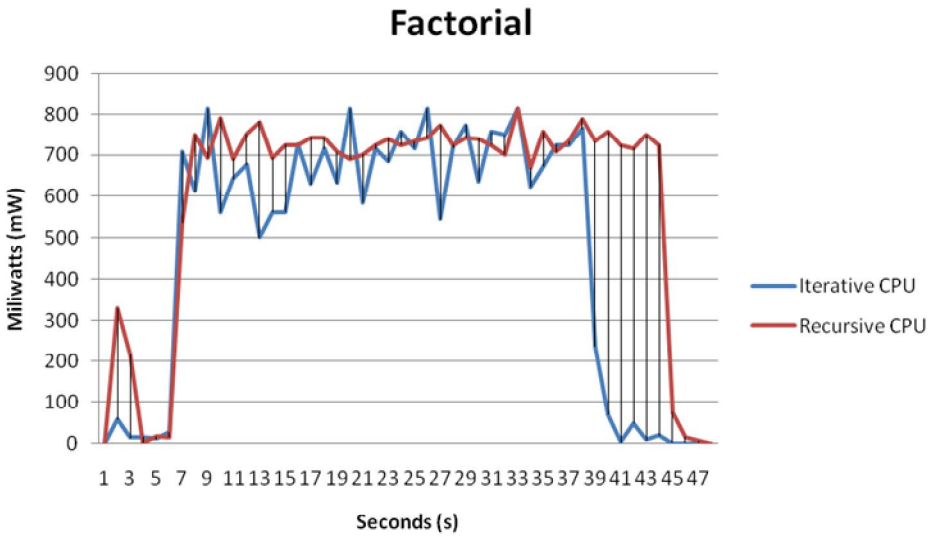

Fig. 2 – Runtime and power dissipated by the iterative and recursive Factorial.

On the algorithm Insertion Sort, the result obtained was within expectations. The time to exhaustion of the maximum CPU where they made the comparisons and exchanges of positions in the array, ignoring the startup time of the program such as loading the data and graphical interface was about 1 second then fell soon after processing, different from the version recursive which was maintained for 5 seconds with processing for high as can be seen in Figure 3. With a shorter time by about 40% and dissipating less power when in any interval of time, so the iterative process has a better performance and also consumes less power. In this algorithm even if it were despised the fact of the recursive version cannot order a large set of numbers, it would still be in fact the iterative version the best choice. From the data obtained shows that the versions of the iterative algorithms have always superior performance and a lower power

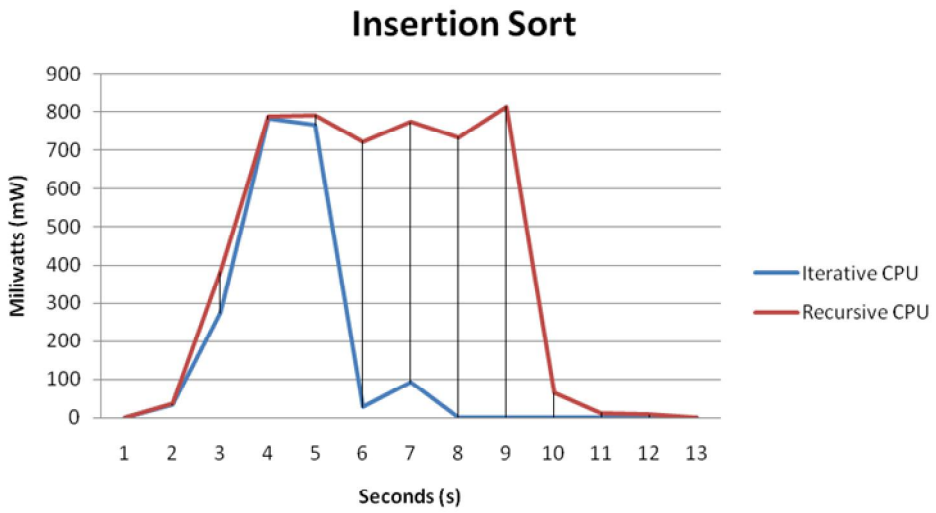

Fig. 2 – Runtime and power dissipated by iterative and recursive Insertion Sort.

5. Conclusion and Future Works

Modern mobile devices are able to play high definitions video and audio, provides access to high-speed Internet, allow shooting and record high quality videos, and also has interfaces with GPS and sensors such as accelerometer. This paper presented a performance evaluation and energy consumption of applications for the Android mobile platform. The evaluated applications running algorithms in iterative and recursive versions of the same computational complexity. Such algorithms are: Insertion Sort and Factorial.

The main objective of this study was to use the PowerTutor tool, which provides estimates of power dissipation for hardware components, allowing analyzing the behavior of energy consumption in different components of the device [9]. These data were analyzed together with data obtained from the TraceView tool [2], which provides a report of the methods implemented and the runtime of the application.

Crossing the data of the two tools we get a detailed list of performance and power dissipation of each method executed by the applications evaluated. This allowed us to suggest the version of an algorithm that provides the best compromise between performance and power consumption for the platform. The importance of the results obtained in this study lies in the fact that we can determine the impact of design changes in software power dissipated by mobile applications for the Android platform.

6. References

[1] R. R. Lecheta, “Google Android: Aprenda a criar aplicações para dispositivos moveis com o Android SDK”. 2nd ed. São Paulo: Novatec Editora, 2010.

[2] Google Android. “Android Developers”, Oct. 2011; http://developer.android.com/index.html.

[3] Mark Gordon, Lide Zhang, et. al. “PowerTutor”, Nov. 2011: http://powertutor.org/

[4] C. Thompson, J. White, B. Dougherty, D. Schmidt. “Optimizing Mobile Application Performance with Model–Driven Engineering”, Lecture Notes in Computer Science, vol. 5860, pp. 36-46, 2009.

[5] Gartner Inc. “Gartner Highlights Key Predictions for IT Organizations and Users in 2010 and Beyond”. Oct. 2011; http://www.gartner.com/it/page.jsp?id=1278413

[6] E. Kreiman. “Using Learning to Predict and Optimise Power Consumption in Mobile Devices”. Oct. 2010: http://www.doc.ic.ac.uk/teaching/distinguished-projects/2010/e.kreiman.pdf

[7] T. Cormen, C. Leiserson, et al. “Algoritmos Teoria e Prática”. 2nd ed. Rio de Janeiro: Elsevier, 2002.

[8] Python Software Foundation. “Python Programming Language”. Oct. 2011: http://www.python.org/.

[9] L. Zhang, L. Yang, et. al. “Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones”. October 2010, Scottsdale, Arizona, USA.