Open Source Mobile Virtual Machines: An Energy Assessment of Dalvik vs. ART

Anton B. Georgiev, Alberto Sillitti, and Giancarlo Succi

Free University of Bozen Bolzano (UNIBZ) Piazza Domenicani 3
39100 Bolzano, Italy
{anton.georgiev}@stud-inf.unibz.it,
{Alberto.Sillitti,Giancarlo.Succi}@unibz.it

Abstract. Dalvik Virtual Machine is Open Source Software and an important part of the Android OS and its better understanding and energy optimization can significantly contribute to the overall greenness of the mobile environment. With the introduction of the OSS solution, named Android Runtime (ART) an attempt of performance and energy consumption optimization was made. In this paper we investigate and compare the performance of the Dalvik virtual and ART from energy perspective. In order to answer our research questions we executed a set of benchmarks in identical experimental setup for both runtimes, while measuring the energy spent and percentage of battery discharge. The results showed that in most of the use case scenarios Ahead-Of-Time compilation process of ART is overall more energy efficient than the Just-In-Time one of Dalvik.

1 Introduction

Dalvik was the virtual machine that Google created to use in Android, its mobile Operating System. For 7 years it evolved alongside with other modules of Android. Dalvik made use of Just-In-Time compiler as a suitable solution for devices with space limitations, characteristic of the first generation devices equipped with Android OS. With the growth of computational power, storage space, display size and embedded sensors, these limitations become less restrictive. This enabled the Operating System developers to seek performance optimizations and better UI responsiveness by introducing new virtual machine. ART stands for Android runtime and it was introduced as a new feature available in version 4.4 of the Android Operating System. It was an optional feature in that release because it was in experimental phase, and its inclusion was triggered by the need of having feedback from both users and developers.

In this paper we designed and executed an experiment to analyze how the new ART affects the overall energy consumption. We based our study on the execution of several benchmarks.

The rest of this paper is organized as follows: section 2 introduces Android OS virtual machine; section 3 comments related work; section 4 describes the methodology and environment created to carry out our experimentation; section 5 presents and analyzes the data collected during the benchmarks' executions and section 6 identifies directions for future work and draws the conclusions.

2 Android OS and Its Virtual Machines

The Android Operating System is an open source operative platform for managing mobile devices. Its open nature allows researchers to analyze its structure and internal organization, facilitating a richer analysis and offering better development possibilities [1,2].

Android is organized in different layers to place the services and applications according to specific requirements and necessities. It is based upon a customized version of the Linux kernel, which acts as the abstraction layer between the hardware and the software. On top of the kernel there is a set of Open Source C/C++ libraries. In the topmost part of the stack the Android apps reside [3]. These apps are programmed in Java programming language. Such programs are compiled into .dex (Dalvik Executable) files, which are in turn zipped into a single .apk file on the device. Android's dex files is created by automatically translating compiled applications written in the Java programming language; the Java compiler (javac) then produces Java bytecode. The dex compiler (dex) then translates java bytecode to a Dalvik bytecode; and Dalvik bytecode is then executed by a virtual machine called Dalvik (DVM) [3].

With the advancement of the mobile devices, it was reached a level where the storage space is relatively big Android developers decided to introduce new virtual machine called Android runtime (ART). It is an alternative of the Just-In-Time compiler, which was part of the Android OS since version 2.2 [4].

With the release of ART Google aimed to improve the GUI responsiveness, to shorten the loading time of the mobile applications and to optimize the execution time and energy consumption of its system. However it has not been discussed what would be eventual implication at user level, for instance an additional use of resources or and additional energy tall

The goal of this work is to analyze the performance of Dalvik and ART from energy point of view.

With the purpose to reach this goal, we defined two research questions:

R.Q.1 Are there any differences between Dalvik and ART in terms of energy efficiency?

R.Q.2 How big is the difference between the two FOSS technologies from the energy perspective?

In order to answer our research questions we executed a set of benchmarks in the same experimental setups for both Dalvik and ART, measuring the energy spent and percentage of battery discharge.

3 Related Work

The Open Source nature of the Android OS in combination with mobile area triggered the interest of many enthusiasts to benchmark and measure the performance of both Dalvik and ART. Since ART was introduced in version 4.4 of the Android OS there are just a few articles [5, 6]. Based on their results one can conclude that even in experimental phase, there are use cases (e.g. floating point calculations) where ART performs better than Dalvik.

In contrast with ART, Dalvik was part of Android since the beginning. During the years of Android OS evolution there were series of studies analyzing its virtual machine. Some of them offer a special focus on assessing the performance of Java and C implementations, conduct a comparison between them [7, 8]. Other work [9] presented a review of the energy consumption of an Android implementation in comparison with the same routine in Angstrom Linux.

In a wider scope there are several research publications than were found during our literature survey dealing with the problem of power consumption in a smartphone. The work of Olsen and Narayanaswami [10] presents and Operating System's power management scheme, named PowerNap, which aims to improve the battery life of mobile device by putting the system in more efficient low power state. The authors of another research publication [11] analyzed the power consumption of a mobile device and measured not only the overall energy spent, but also the exact break down by the device's main hardware components. In another track, analyzing component specific energy characterization is an aim for the authors [12], which used a software tool named CharM to survey and collect parameters, exposed by the Operating System. The presented results showed that CPU, the OLED display and the WiFi interface are the elements that are taking the highest tall from the device battery when stressed.

For the purposes of our open source experimental setup, we needed a state of the art analysis of the mobile software based energy measurement. After surveying the literature in the area we discovered a set of applications which implements profiling techniques that analyze the source code. With these tools at hand you can obtain very detailed information about what are the routines and system calls that have a major contribution to the energy utilization [13]. Building an energy model using hardware measurement is another approach that is used in this area [14, 15, 16, 17, 18].

4 Data Collection

4.1 Benchmarks

To explore all the differences between the two runtimes we selected several Benchmarking apps from Google play. These software products will guaranteed a fair comparison, because they are executing the same set of instruction in identical sequence each time they perform their benchmark. The benchmark applications utilized on our experimentation were:

- *AndEBench* [19] – it is an app providing standardized method, which is industry-accepted for evaluating Android OS performance.
- *Antutu Benchmark* [20] – is a benchmark tool, which includes – CPU, GPU, RAM, I/O tests with the option to compare your results with popular Android OS devices
- *GFXBench 3.0* [21] – is a comprehensive benchmark and measurement tool, with cross-platform and cross-API capabilities. It measures graphics performance alongside with render quality and long-term performance stability of the device. We selected to include two of the available Low-Level tests – basic ALU and its 1080p off-screen variation.
- *Pi Benchmark* [22] – calculates Pi up to the 2.000.000 digits of precision. For our experiment we were execution this benchmark with load of 500.000 digits.
- *Quadrant Standard Edition* [23] – is an application, which requires Internet connection to compute the results and doesn't work on device without GPU. It stresses the CPU, GPU and performs I/O operations.
- *RL Benchmark: SQLite* [24] – determines how much time you need to process a number of SQL queries.
- *Vellamo Mobile Benchmark* [25] – evaluates the web browser performance of the device including scrolling and zooming, 3D graphics, video and memory read/write performance.

With these benchmarks we also made sure that we exercise different components that may impact the execution and the overall user experience while using an app.

4.2 Experimental Setup

Our experimental setup used two software tools to collected information regarding the energy consumption of the executed benchmarks:

- *A Custom version of PowerTutor* [13] to measure the energy spent in Joules. The samples were collected after every single benchmark execution.
- *CharM* [charm] – to read the battery percentage spent. CharM is a tool that surveys battery discharge cycle. It collects the date in a background process, and is implemented with negligible energy footprint. We used it to collect date more relevant to the end user, as one would normally check the remaining battery percentage on his device. The readings were acquired after the complete cycle of benchmark executions was over.

Before each measurement phase we were resetting the PowerTutor and CharM instances to assure that the benchmark application will start its energy consumption from 0. The battery of the phone was fully charged and left to cool down to assure that the settings for each test are as identical as possible. For reproducibility purposes we repeated each test 30 times.

Our test bed options were limited, because ART was experimentally introduced only for Android 4.4 compatible devices. We selected a Google® HTC Nexus 4™ device and all the results presented in this paper are according to its capabilities and specification. The phone has the latest stock Android version (4.4.2), without any account information in order to prevent background synchronization with remote servers, which can affect our measurements. The device is equipped with a quad-core Snapdragon S4 Pro APQ8064 processor running at 1.5 GHz, 2 GB RAM and 2100 mAh battery.

5 Data Analysis

5.1 Data Interpretation

After executing all the benchmark series we collected more than 500 samples for both virtual machines. Table 1 shows the average energy consumption in Joules of 30 cycles for each benchmark. Column two contains readings associated with Dalvik virtual machine, while column three contains the ART ones.

Table 1. Summary of the energy spent per benchmark application in Joules

Benchmark Name	Dalvik	ART
AndEBench [b1]	88.36 J	115.18 J
Antutu Benchmark [b2]	265.66 J	265.18 J
GFXBench 3.0 [b3]	90.50 J	90.50 J
Pi Benchmark [b4]	217.40 J	92.68 J
Quadrant Standard Edition [b5]	77.56 J	67.62 J
RL Benchmark:SQLite [b6]	51.46 J	29.30 J
Vellamo Mobile Benchmark [b7]	293.18 J	284.10 J

The energy spent to complete each computational job varies largely due to the nature of the selected benchmarks. Some of them like Antutu Benchmark [20] and Vellamo Mobile Benchmark [25] considerably higher energy consumption than the rest of the experiments.

By the graphics shown in Figure 1, with the exception of AndEBench, ART performed equally or better in comparison with Dalvik. We have the biggest difference in the Pi Benchmark where the newer virtual machine was more than two times efficient in comparison with its predecessor.

In Table 2 is summarized the average of the percentage that was discharge from the battery during one benchmark series (i.e. the execution of a benchmark for 30 times). The readings were collected using CharM application.

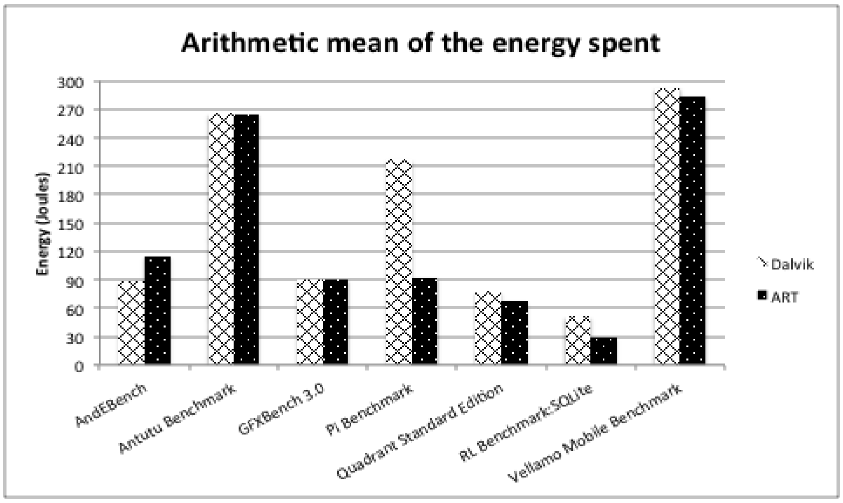

Fig. 1. Arithmetic mean of the energy spent

Table 2. Summary of the percentage of battery discharge per benchmark application

Benchmark Name	Dalvik	ART
AndEBench	32 %	45 %
Antutu Benchmark	57 %	57 %
GFXBench 3.0	30 %	30 %
Pi Benchmark	55 %	36 %
Quadrant Standard Edition	35 %	31 %
RL Benchmark:SQLite	11 %	6 %
Vellamo Mobile Benchmark	74 %	61 %

The plots of the remaining battery level after each benchmark execution per selected experiment can be seen on Figure 2. The figure presents information about the battery level indicator that the end user will see on his screen after the completion of each experimental task.

5.2 Research Questions Revisited

R.Q.1 Are there any differences between FOSS virtual machines Dalvik and ART in terms of energy efficiency?

After analyzing the collected results it is evident that in almost all cases we have difference in the energy spent between the two virtual machines. The only benchmark that we had exactly the same reading was *GFXBench 3.0* with 90.50 Joules for both Dalvik and ART.

R.Q.2 How big is the difference between the two FOSS technologies from energy perspective?

Readings in Table 1 show that out of seven benchmarks only in one, namely *AndEBench*, we have advantage of 26.82 Joules for the Dalvik virtual machine. In the remaining 6 benchmarks, excluding *GFXBench 3.0,* we always have better energy efficiency for the ART virtual machine with average savings from 0.48 Joules (for *Antutu Benchmark*) to 124.72 Joules (for *Pi Benchmark*).

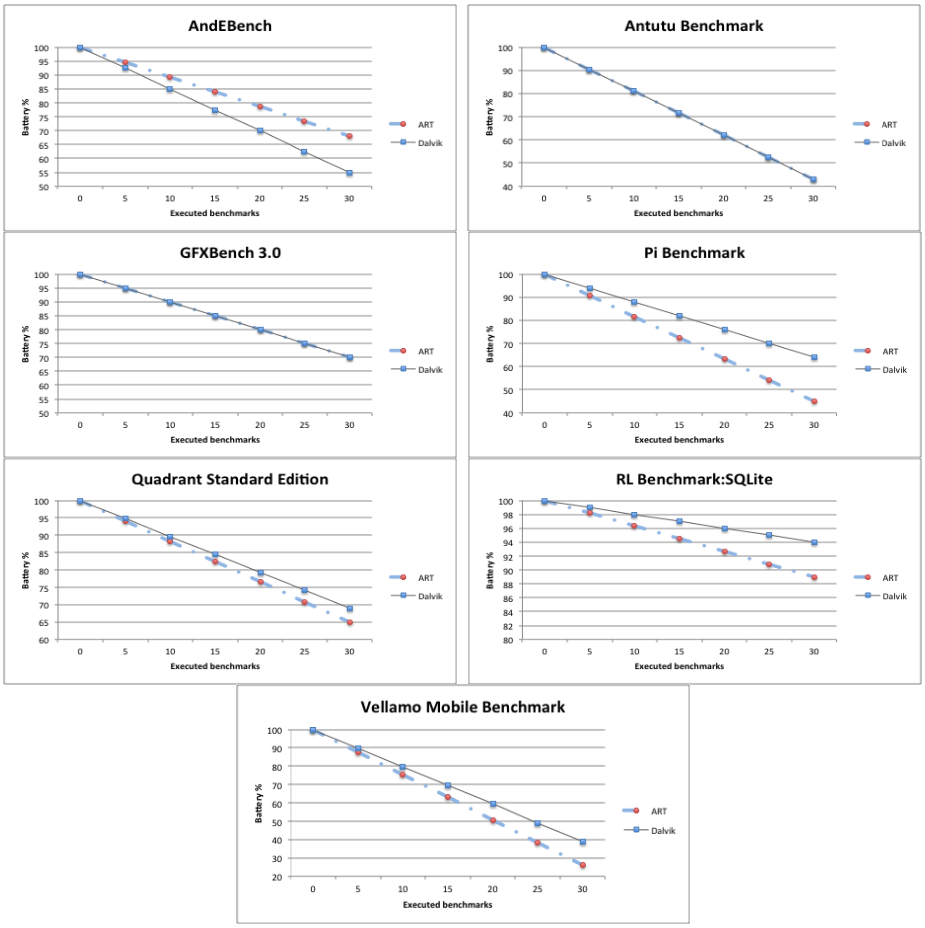

Fig. 2. Battery discharge level after benchmark execution in Dalvik and ART

6 Future Work and Conclusions

6.1 Directions for Future Work

Possible continuation of the work presented in this paper is to replicate the benchmark executions on different test beds. Collecting samples not only from smart phones, but also from tablets, smart TVs and others Android embedded device will deepen our understanding of the energy performance in the Operating System. Reproducing the described experiments using hardware measurement tools instead of software will provide additional verification of the previously generated data. Moreover we also foresee to increase the granularity of the collected measurements [26], so in the future we can analyze in greater details the relationship between the code execution and the corresponding energy consumption.

6.2 Conclusions

In this work we presented and analyzed data collected from set of selected benchmarks in order to analyze how the new Android runtime affects the overall energy consumption of the device under test. These benchmarks were selected with the goal of stressing different components that may impact the execution and the overall user experience while using an app. The results showed that replacing a Just-In-Time compiler with virtual machine with Ahead-Of-Time compilation process would optimize the energy utilization for five out of seven scenarios exercised using our experimental setup.

Optimizing the performance of a mobile device and by doing so reducing the energy consumption is of a great importance for the future development of the mobile environment. Having a strong requirement for autonomy provokes not only the mobile hardware specialists, but also software developers to seek for a solution that will increase the battery life. If they omit the energy consumption factor while designing their products, they could not only decrease the user satisfaction, but in a long term they might significantly contribute to environmental pollution by increasing the battery garbage.

References

[1] Di Bella, E., Sillitti, A., Succi, G.: A multivariate classification of open source developers. Information Sciences 221, 72–83 (2013)

[2] Kovács, G.L., Drozdik, S., Succi, G., Zuliani, P.: Open source software for the public administration. In: Proceedings of the 6th International Workshop on Computer Science and Information Technologies (2004)

[3] Ehringer, D.: The Dalvik Virtual Machine Architecture (March 2010), http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf (retrieved November 4, 2013)

[4] Dalvik (software). In Wikipedia (software) (December 1, 2013), http://en.wikipedia.org/wiki/Dalvik_(software) (retrieved December 2, 2013)

[5] Toombs, C.: Meet ART, Part 2: Benchmarks - Performance Won't Blow You Away Today, But It Will Get Better (November 12, 2013), http://www.androidpolice.com/2013/11/12/meet-art-part-2-benchmarks-performance-wont-blow-away-today-will-get-better/ (retrieved December 1, 2013)

[6] Toombs, C.: Meet ART, Part 3: Battery Life Benchmarks - Not Good, But Not Too Bad (January 22, 2014), `http://www.androidpolice.com/2014/01/22/meet-art-part-3-battery-life-benchmarks-not-good-but-not-too-bad/` (retrieved January 26, 2014)

[7] Cargill, D.A., Radaideh, M.: A practitioner report on the evaluation of the performance of the C, C++ and Java compilers on the OS/390 platform. In: Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, pp. 40–45 (2000)

[8] Corsaro, A., Schmidt, D.: Evaluating Real-Time Java Features and Performance for Real-time Embedded Systems. In: Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 90–100 (2002)

[9] Kundu, T.K., Kolin, P.: Android on Mobile Devices: An Energy Perspective. In: Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology, pp. 2421–2426 (2010)

[10] Olsen, C.M., Narayanaswami, C.: PowerNap: An efficient power management scheme for mobile devices. IEEE Trans. on Mobile Computing 5(7), 816–828 (2006)

[11] Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: USENIX Annual Tech. Conf., pp. 21–34 (2010)

[12] Corral, L., Georgiev, A.B., Sillitti, A., Succi, G.: A Method for Characterizing Energy Consumption in Android Smartphones. In: Proceedings of the 2nd International Workshop on Green and Sustainable Software (GREENS 2013), in connection with ICSE 2013, pp. 38–45 (2013)

[13] Zhang, L., Tiwana, B., Qian, Z., Wang, Z.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: 8th Intl. Conf. on HW/SW Codesign and System Synthesis, pp. 105–114 (2010)

[14] Wonwoo, J., Chulko, K., Chanmin, Y., Donwon, K., Hojung, C.: DevScope: a nonintrusive and online power analysis tool for smartphone hardware components. In: 8th Intl. Conf. on HW/SW Codesign and System Synthesis, pp. 353–362 (2012)

[15] Yoon, C., Kim, D., Jung, W., Kang, C., Cha, H.: AppScope: application energy metering framework for Android smartphone using kernel activity monitoring. In: USENIX Annual Technical Conference (2012)

[16] Pathak, A., Hu, Y., Zhang, M.: Where is the energy spent inside my app? Fine grained energy accounting on smartphones with Eprof. In: 7th ACM European Conference on Computer Systems, EuroSys 2012, pp. 29–42 (2012)

[17] Corral, L., Sillitti, A., Succi, G., Strumpflohner, J., Vlasenko, J.: DroidSense: A mobile tool to analyze software development processes by measuring team proximity. In: Furia, C.A., Nanz, S. (eds.) TOOLS 2012. LNCS, vol. 7304, pp. 17–33. Springer, Heidelberg (2012)

[18] Corral, L., Sillitti, A., Succi, G., Garibbo, A., Ramella, P.: Evolution of mobile software development from platform-Specific to web-Based multiplatform paradigm. In: Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, pp. 181–183. ACM (October 2011)

[19] EEMBC, AndEBench (Version 1605) (2014), `https://play.google.com/store/apps/details?id=com.antutu.ABenchMark&hl=en` (retrieved)

[20] AnTuTu, Antutu Benchmark (Version 4.1.7), `https://play.google.com/store/apps/details?id=com.eembc.coremark` (retrieved)

[21] Kishonti Ltd. GFXBench 3.0 3D Benchmark (Version 3.0.4) (2014), from `https://play.google.com/store/apps/details?id=com.glbenchmark.glbenchmark27` (retrieved)
[22] Markovic, S.D.: PI Benchmark (Version 1.0) (2014), `https://play.google.com/store/apps/details?id=rs.in.luka.android.pi` (retrieved)
[23] Aurora Softworks, Quadrant Standard Edition (Version 2.1.1) (2014), `https://play.google.com/store/apps/details?id=com.aurorasoftworks.quadrant.ui.standard` (retrieved)
[24] RedLicense Labs, RL Benchmark: SQLite (Version 1.3) (2014), `https://play.google.com/store/apps/details?id=com.redlicense.benchmark.sqlite` (retrieved)
[25] Qualcomm Connected Experiences, Inc., Vellamo Mobile Benchmark (Version 2.0.3) (2014), `https://play.google.com/store/apps/details?id=com.quicinc.vellamo` (retrieved)
[26] Scotto, M., Sillitti, A., Succi, G., Vernazza, T.: A non-invasive approach to product metrics collection. Journal of Systems Architecture 52(11), 668–675 (2006)