Comparing Android Runtime with native code:
Fast Fourier Transform on Android
Specification

André Danielsson
anddani@kth.se

Monday 13th February, 2017

Bontouch Supervisor :	Erik Westenius	erik.westenius@bontouch.com
CSC Supervisor :	Erik Isaksson	erikis@kth.se
Examiner :	Olle Bälter	balter@kth.se

Background and Objective

This thesis is a study that evaluates when and where there is a gain in writing a component of an Android application in code that is compiled into machine code. Android applications are mainly written in Java to ensure portability in form of architecture independence. By using a virtual machine to run a Java app, you can use the same bytecode on multiple platforms. To ensure efficiency on low resources devices, a virtual machine called Dalvik was developed. Applications on Android have been using the Dalvik virtual machine until Android version 5 [1] in November of 2014 [2]. Since then, Dalvik has been replaced by Android Runtime. Android Runtime, ART for short, differs from Dalvik in that it uses Ahead-Of-Time (AOT) compilation. This means that it compiles during the installation of the app. Dalvik, however, exclusively uses a concept called Just-In-Time (JIT) compilation, meaning that code is compiled during runtime when needed. ART uses Dalvik bytecode to compile an application, allowing most apps that are aimed at Dalvik Virtual Machine to work on devices running ART.

The reason you would want to write part of an application in native code is to potentially get better execution times of computational heavy tasks such as the Fast Fourier Transform (FFT). The FFT is an algorithm that computes the Discrete Fourier Transform (DFT) of a signal. It is primarily used to analyze the components of a signal. This algorithm is used in signal processing and has multiple purposes such as image compression (taking photos), voice recognition (Siri, Google Assistant), fingerprint scanning (unlocking device) to name a few. Another reason you would want to write native libraries is to reuse already written code in C or C++ and incorporate it into your project. This allows the app to become more platform independent. Shared code can be used in a computer app, Apple IOS app and more. This thesis will only cover the performance aspects of calling native code through the JNI instead of running it in Java.

One purpose of this thesis is to educate the reader about the process of porting parts of an app to native code using the Native Development Kit (NDK). Another is to explore the topic of performance differences between Android Runtime (ART) and native code compiled using a compiler called Clang/LLVM. Because ART is relatively new (Nov 2014) [2], this thesis would contribute with more information related to the performance of ART and how it performs compared to native code compiled by the NDK.

Some of the findings in this thesis might help decide which method of programming for Android that should be used for a given problem. For some problems, it is necessary to choose the appropriate programming method to ensure that an application is smooth and responsive. It is therefore important to know when and where it is necessary to optimize code. Further, when developing for Android there are multiple types of problems that occur and it is relevant to know which problems are worth solving in NDK rather than the SDK.

The objective of this project is to investigate when and where it would be beneficial to use the NDK instead of Java when developing for Android. This project will cover how an implementation of the FFT benefits from being compiled by the NDK with Clang compared to Java code compiled by Android Runtime. In Android development, you want to use Java when you control how the application should behave and call native functions when a lot of processing is needed. The overhead caused by calling a native function via the Java Native Interface (JNI) is something that must be taken into account when measuring the execution time of the programs.

The results from the experiments are expected to confirm or deny my problem statement and answer my research question. By thoroughly examining multiple methods and choosing the correct one, it is possible to correctly answer the research question. When constructing the experiments, a lot of time will be spent on finding the most appropriate approach of evaluating the FFT algorithm.

Research Question and Method

The research question of this thesis is the following:

Is there a significant performance difference between implementations of a Fast Fourier Transform (FFT) in native code, compiled by Clang, and Dalvik bytecode, compiled by Android Runtime, on Android?

The goal of this thesis is to examine the efficiency of ART and how it compares to native code written in C++ using the NDK in combination with the JNI. This thesis presents a study that investigates the relevance of using the NDK to produce efficient code. Further, the cost to pass through the JNI will also be a factor when analysing the code. A discussion about to what extent the simplicity of the code outweighs the performance of the code will also be present. For people who are interested in knowing about the performance impacts of implementing algorithms in C++ for Android, this thesis might be of some use.

Finding out how to accurately test the implemented algorithms is done by using knowledge gathered from literature and previous courses. The experiments will consist of programs, written in both Java and C++, that are compiled and executed on a mobile device. The execution time of these programs will be measured and compared with each other.

The execution time of the programs will vary because of factors such as scheduling, CPU clock frequency scaling and other uncontrollable behaviour caused by the operating system. To get accurate measurements, a mean of a large numbers of runs are calculated for each program. Additionally, it is also necessary to calculate the standard error of each set of execution times. With the standard error we can determine if the difference in execution time between two programs are statistically significant or not.

Evaluation

As mentioned previously, a statistical analysis will be carried out to ensure that the statements made about the data are reasonable. The discussion of the thesis will also contain some thoughts on some factors that might have influenced the result of the experiments. Furthermore, a discussion will be held about how the results would change by factors that are not reproducible for example which apps the users will have in the background while running your application.

The result of the thesis can be used to value the decision of implementing a given algorithm or other solutions in native code instead of Java. As previously stated, the FFT is an algorithm that is frequently used in app development. It is therefore valuable to know how efficient an implementation in native code is depending on the size of the data.

Pilot Study

During the first weeks of my project, I will focus on writing the introduction and start with the background to confirm that I have a clear view on how the project will progress. In this chapter I will go through all the relevant literature that is needed to advance. Some articles will be related to how Android Runtime performs and others about JNI and NDK related topics. There is a lot of change in the field of mobile software and some articles can be considered irrelevant for my thesis due to its age. I will ensure that the articles are still relevant before using them as a source.

Additionally, the details of the Discrete Fourier Transform will be expanded to give understanding about its details and where it can be applied. An article published by Cooley and Tukey on the Fast Fourier Transform algorithm [3] which computes the DFT in $O(n \log n)$ time instead of $O(n^2)$ (where n is the number of data points) will be of some use in the literature review.

There is already some previous work regarding how FFT performs on Android. The article "FFT benchmark on Android devices: Java versus JNI" [4] compares two implementations of FFT, one written in Java and one written in C++. This article does not go into detail on its subject although it does state that their Java library of FFT is faster than their C++ library of FFT for small data. This article was written in 2013 and versions of both GCC and Clang/LLVM compilers has been changed multiple times since then. With new versions of Android, updates to the kernel and software versions might change results found in previous studies. This is one reason why the subject of this project is relevant.

The general topics that are going to be covered in the literature review are:

- *Android Runtime*
- *Clang/LLVM*
- *Native Development Kit and Java Native Interface*
- *Discrete Fourier Transform and Fast Fourier Transform*

The literature that was chosen for the pilot study are the following:

- "Performance Analysis of Android underlying Virtual Machine in Mobile Phones" [5] discusses the differences between Dalvik bytecode and standard Java bytecode. This will give some interesting information about how Dalvik differs and how a register-based machine performs compared to a stack-based machine.
- "A comparison of data serialization formats for optimal efficiency on a mobile platform" [6] is an article that compares methods of serializing data. The result and discussion of this article might give some insight into why one method is faster than the other.
- "Analysis and evaluation of the android best practices impact on the efficiency of mobile applications" [7] measures the performance difference when coding in different styles in Java. This is something that is interesting to take into consideration when discussing the results of my findings.
- "Performance analysis for android runtime environment" [8] discusses the improvements of Android Runtime and has a performance comparison between DVM and ART.
- "Comparison and analysis of the three programming models in google android" [9] examines different aspects of programming in Java SDK, C++ NDK and Renderscript. It takes up aspects such as programmability, portability, security and performance.
- "Android App Energy Efficiency: The Impact of Language, Runtime, Compiler, and Implementation" [10] investigates performance differences between some algorithms written in Java and C++ (using NDK).
- "Analysis of Dalvik virtual machine and class path library" [11] is a technical report that goes into detail how the DVM works.
- "Benchmarking Java Application Using JNI and Native C Application on Android" [12] looks at how native code performs when compiled by the NDK and called from Java using the JNI compared to using an ARM cross-compiler.
- "Evaluating Performance of Android Platform Using Native C for Embedded Systems" [13] evaluates the difference between DVM and C code compiled with the NDK and called using JNI.

- “An investigation into energy-saving programming practices for Android smartphone app development” [14] investigates how programming patterns can affect the energy efficiency of the device.
- “Benchmark Dalvik and native code for Android system” [15] is another comparison between Dalvik and Native code compiled using NDK.
- “Developing and benchmarking native linux applications on Android” [16] investigates performance difference between Java and C code between Android and PC. This way, they can compare how much the performance between the languages differs between platforms.
- “A hybrid just-in-time compiler for android” [17] explains how the just-in-time compiler and proposes an improved method of just-in-time compilation.
- “A method-based ahead-of-time compiler for Android applications” [18] compares AOT and JIT compiling methods.
- “On Tracking Information Flows through JNI in Android Applications” [19] is a study on how JNI passes information between Java code and native shared libraries.
- “Digital Signal Processing: Fundamentals and Applications” [20] will be used as a source for the literature review about the Discrete Fourier Transform.

Conditions and Schedule

Some literature that might be of interest are listed on the last pages of this document. All other resources such as Android device(s) used for testing are provided by Bontouch.

Due to time constraints, this project will only cover a performance evaluation of the FFT algorithm. The decision of choosing the FFT was due to it being a common algorithm to use for signal analysis. This project will not investigate the performance differences for FFT in parallel due to the complexity of the Android kernel. This is something that could be interesting to look at as an extension of this thesis.

Scheduling and power saving functionality will start come into play and those topics are large in scope by themselves. This would require more knowledge outside the scope of this project and would result in a more broad subject. Another limitation is to only use one phone to perform the experiments on. This is because we would otherwise just compare phones. As of now, I will limit myself to one version of the Android operating system. This is so that I can discuss more about the differences between the algorithms but also because there might be small changes in the OS affect the performance and skew the results.

My supervisor at Bontouch, Erik, will help me with technical questions as well as direct me throughout my project to ensure that I don’t deviate from the subject. We will also have regular meetings to discuss my work so far.

Week #	Goal
4	Pilot Study - Introduction
5 – 7	Pilot Study - Background/Literature review
8 – 10	Prepare experiment and Write on report, methodology
11 – 12	Write on report, result
13 – 14	Write on report, discussion
15	Write on report, conclusion
16 – 17	Write on report, fine-tune
18 – 19	Prepare for presentation
20	Hand in report
21 –	Presentation

Detailed Schedule

- **Week 4:** This week, I will try to familiarize myself with the NDK and start with the pilot study. My goal for the end of this week is to have a general view on the whole project. The first draft of the introduction chapter will also be completed.
- **Week 5:** For the first week of the Literature Review, I will spend time reading about the different topics and other articles to try to broaden my understanding of this area. My plan is to focus on how the NDK works and the process of calling native subroutines from Java using the JNI.
- **Week 6:** The second week will be dedicated to write about the FFT and look through different implementations of the algorithm.
- **Week 7:** The focus of this week will be to finish the first draft of the Background chapter.
- **Week 8:** The first part of the experiment will be to do research about different methodologies and of setting up the experiments as well as how to interpret the results. All of this will be added to the Methodology chapter.
- **Week 9:** The second week of the experiment block will be to continue with the Methodology chapter and start with the implementation of the benchmark program.
- **Week 10:** The goal of this week is to have completed most of the Methodology chapter as well as the implementation of the benchmark program.
- **Week 11:** This week will be dedicated to performing all the tests that will be the basis of the report. All the relevant data will be added to the report.
- **Week 12:** Extra week to finish with the result of the report.
- **Week 13:** Start with the discussion of the report, start with the abstract.
- **Week 14:** Finish the Discussion chapter.
- **Week 15:** Write the Conclusion chapter of the report and finish the abstract.

- **Week 16:** Extra time for report writing as well as fine-tuning of the report.
- **Week 17:** Extra time for report writing as well as fine-tuning of the report.
- **Week 18:** This week will be dedicated to prepare for presentation and extra time for report writing if needed. If done, hand in report before opposition.
- **Week 19:** This week will be dedicated to prepare for presentation and extra time for report writing if needed. Write the written opposition report
- **Week 20:** The goal of this week is to hand in the report, do the presentation, self-evaluation and other.

References

[1] Google, "Android 5.0 Behavior Changes – Android Runtime (ART)." `https://developer.android.com/about/versions/android-5.0-changes.html`. [Accessed: 24 January 2017].

[2] Google, "android-5.0.0_r1 - platform/build - Git at Google." `https://android.googlesource.com/platform/build/+/android-5.0.0_r2`. [Accessed: 24 January 2017].

[3] B. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation Complex Fourier Series," pp. 297–301, 1964.

[4] A. D. D. C. Jr, M. Rosan, and M. Queiroz, "FFT benchmark on Android devices : Java versus JNI," pp. 4–7, 2013.

[5] E. Azimzadeh, M. Sameki, and M. Goudarzi, "Performance Analysis of Android underlying Virtual Machine in Mobile Phones," *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin*, pp. 292–295, 2012.

[6] A. Sumaray and S. K. Makki, "A comparison of data serialization formats for optimal efficiency on a mobile platform," *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, vol. 8, no. 4, p. 1, 2012.

[7] A. R. Tonini, L. M. Fischer, J. C. B. De Mattos, and L. B. De Brisolara, "Analysis and evaluation of the android best practices impact on the efficiency of mobile applications," *Brazilian Symposium on Computing System Engineering, SBESC*, pp. 157–158, 2014.

[8] R. Yadav and R. S. Bhadoria, "Performance analysis for android runtime environment," *Proceedings - 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015*, pp. 1076–1079, 2015.

[9] X. Qian, G. Zhu, and X. Li, "Comparison and analysis of the three programming models in google android," *First Asia-Pacific Programming Languages and ...*, 2012.

[10] X. Chen and Z. Zong, "Android App Energy Efficiency: The Impact of Language, Runtime, Compiler, and Implementation," *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pp. 485–492, 2016.

[11] S. E. R. Group *et al.*, "Analysis of dalvik virtual machine and class path library," *Institute of Management Sciences, Peshawar, Pakistan*, 2009.

[12] Y.-j. Kim, S.-j. Cho, K.-j. Kim, E.-h. Hwang, S.-h. Yoon, and J.-w. Jeon, "Benchmarking Java Application Using JNI and Native C Application on Android," *12th International Conference on Control, Automation and Systems*, pp. 284–288, 2012.

[13] S. Lee and J. W. Jeon, "Evaluating Performance of Android Platform Using Native C for Embedded Systems," *International Conference on Control, Automation and Systems*, pp. 1160–1163, 2010.

[14] D. Li and W. G. J. Halfond, "An investigation into energy-saving programming practices for Android smartphone app development," *Proceedings of the 3rd International Workshop on Green and Sustainable Software - GREENS 2014*, pp. 46–53, 2014.

[15] C. M. Lin, J. H. Lin, C. R. Dow, and C. M. Wen, "Benchmark Dalvik and native code for Android system," *Proceedings - 2011 2nd International Conference on Innovations in Bio-Inspired Computing and Applications, IBICA 2011*, pp. 320–323, 2011.

[16] L. Batyuk, A. D. Schmidt, H. G. Schmidt, A. Camtepe, and S. Albayrak, "Developing and benchmarking native linux applications on Android," *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 7 LNICST, pp. 381–392, 2009.

[17] G. A. Perez, C.-M. Kao, Y.-C. Chung, and W.-C. Hsu, "A hybrid just-in-time compiler for android," *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems - CASES '12*, p. 41, 2012.

[18] C.-S. Wang, W.-C. Hsu, G. A. Perez, W.-K. Shih, Y.-C. Chung, and H.-R. Hsu, "A method-based ahead-of-time compiler for Android applications," *2011 Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, pp. 15–24, 2011.

[19] C. Qian, X. Luo, Y. Shao, and A. T. S. Chan, "On Tracking Information Flows through JNI in Android Applications," *Proceedings - 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014*, pp. 180–191, 2014.

[20] L. Tan and J. Jiang, *Digital Signal Processing: Fundamentals and Applications.* Elsevier Science, 2013.