Vrije Universiteit Brussel

Proposal bachelor thesis

Title: Analyzing Android Applications to Match Code for Acceleration

Promotor: Professor Jennifer B. Sartor

Includes preparation course: **XX Yes XX** / No (Select the correct option)

Context

Nowadays most software can be run in the palm of your hand – i.e. on your smartphone. The development of software for Android phones has boomed, with novice and expert programmers are creating new interesting apps everyday. These apps include media programs, such as watching video or hearing sound, scientific tasks, and navigating the web, just to name a few examples. In these days, it is important not only to make these kinds of software run fast, and be efficient for the user, but also to not to use too much power, which is synonymous with battery life. We want to look at ways to accelerate these applications to achieve better performance, while not taking too much energy so that the phone does not overheat, or so that your battery does not drain as fast.

Proposal bachelor thesis

We have already built a framework to analyze how to accelerate media applications written in the C programming language. We now want to bring this framework to the mobile world and see the potential to make Android applications faster and minimize their power output. This Bachelor's thesis goal would be to take the infrastructure that another student has built in the LLVM compiler and re-implement it in the Dalvik virtual machine (for Android). One initial task is to find good benchmarks that run on the Android platform to use throughout the project. Then, you will follow the toolchain we already have (developed within the LLVM compiler) to first found out where the applications spend most of their execution time, and then find sequences of statements that perform mathematical operations that could be accelerated by specialized hardware. You will primarily work in the DalvikVM compiler to analyze sequences of code from different applications, find similar sequences from different applications, and then estimate how much performance you could save if you were to have a special piece of hardware that could accelerate these sequences. Your contribution will be to then analyze the similarity between applications, such as how long the sequences of code are that match between applications, and find programs that would be best grouped together to be run on the same hardware. Thus, your project will look towards the future evolution of smartphones, and how to make mobile applications as efficient -- with regards to performance and energy -- as possible.

Preparatory course bachelor thesis

(only if applicable)

This research project is based on a lot of prior work that the student needs to get familiar with. First, the student needs to understand the current implementation of our framework in LLVM, and understand the Intermediate Representation (IR) used in this compiler. This is where code analysis is done. Second, the student should familiarize him/herself with pattern matching techniques and the code diagrams on which they are used during compiler code generation. We use a special diagram called a Taylor Expansion Diagram that allows us to identify similar sequences of mathematical operations across different applications. We have already built a framework to actually analyze code from different programs and do the pattern matching. The point of this preparatory course is for the bachelor's student to become familiar with the flow of our existing framework, with the help of the doctoral student who built it. After that, the student should study the corresponding IR and compiler structure in the Android compiler that is included in the Dalvik virtual machine. Finally, the student can find some interesting apps, such as the research benchmark suite 0xbench, that run on the DalvikVM on top of an Android phone to use for this project. After doing this project, the student will have gained expertise on several compilers, and on code analysis and matching. The research output of this preparatory course would be a comparison of the IR of two highly-used compilers, enumerating their similarities and differences.