Exploiting Parallelism in Mobile Devices

Arghya Chatterjee,Timothy Newton, Tom Roush, Hunter Tidwell, Vivek Sarkar
Department of Computer Science, Rice University, USA
(arghya, kgmstwo, twr1, wht1, vsarkar) @rice.edu

Abstract

The computational heart of modern mobile devices such as smartphones, tablets, and wearables is a powerful system-on-chip (SoC) with rich parallelism and heterogeneity. While the hardware parallelism of these mobile systems continues to increase year-over-year, they remain resource constrained with respect to power consumption and thermal dissipation. Efficient use of multi-core processors in mobile devices is a key requirement for improving performance, while staying within the power and thermal limits of mobile devices.

Categories and Subject Descriptors D.1.3 [*Programming Techniques*]: Concurrent Programming - Parallel Programming

Keywords Mobile devices, parallel programming, multi-core, Habanero Java library, abstraction

1. Motivation

The state of the art approach is to re-use or develop custom computing libraries such as OpenIMAJ [3] and OpenCV [6] with their own specific parallel programming abstractions. However, this approach suffers from the fact that the programmers need to understand the library specific abstractions for development, tune their applications accordingly, and learn different approaches to parallelism in different libraries.

To address this challenge, we explore parallel programming for mobile devices by using the Habanero Java parallel programming library (HJlib) [7]. HJlib offers a wide variety of general structured parallel programming abstraction, that make extensive use of Java 8's lambda expressions and can be composed with each other with well-defined semantics. This single-library approach shortens the learning curve for adding parallelism to a wide range of Android applications, since the same set of HJlib primitives can be used in different application domains. Prior to this work, it was unknown if HJlib would work on Android's Dalvik VM. This work demonstrates that HJlib provides a viable approach for application developers to exploit the multi-core architecture of modern mobile devices more productively than with existing approaches.

SPLASH Companion'15, October 25–30, 2015, Pittsburgh, PA, USA
ACM. 978-1-4503-3722-9/15/10
http://dx.doi.org/10.1145/2814189.2817273

2. Habanero Java Library (HJlib)

This work uses HJlib, developed at Rice University as a part of the Habanero Extreme Scale Software Research Project [2], as our research vehicle. HJlib is a JVM-based parallel programming library that integrates a wide range of parallel programming constructs (`async, finish, forall, isolated, futures, actors, phasers`) into a single programming system, there by enabling unique and rich combinations of these constructs. As a high-level parallel programming model, HJlib makes it easier to focus on general parallel programming concepts without getting into the low-level details. We choose HJlib for two primary reasons: *a*) its library-based implementation in Java 8, and *b*) its ease of programmability on Android platforms.

3. Evaluation

As part of the exploration of HJlib for parallelism on Android phones, we investigated the *Android Image Manipulation*, the *Android Image Filtration* and the *Face Replace* applications. These applications are parallelized using the Habanero programming model constructs (such as `async, finish`) by adding HJlib as one of the third-party libraries to the development flow. Our evaluation uses a Nexus 4 smart phone with a quad-core Qualcomm Snapdragon S4 Pro (APQ8064) processor [1] on Android 4.1. The remainder of this section summarizes the characteristics and performance evaluation of each of these applications.

3.1 Android Image Manipulation

Android Image Manipulation is an existing application built with OpenCV, a well-known computer vision library [5], to apply various filters to a real-time image using the phone's camera. The most time-consuming task in the application is applying a *sepia filter* on the image. We parallelized the *sepia filter* module using the async and finish constructs available in HJlib.

Figure 1 shows the performance (speedup relative to sequential) of the parallelized application while applying the *sepia filter*. As can be seen from the figure, the parallel version of the application on four worker threads performs 1.89× faster relative to the sequential version. This speedup is reflected in a visible improvement in the responsiveness of the application running on the phone.

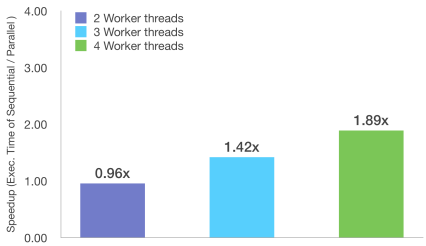

Figure 1: Speedup of the Image Manipulation with diff numbers of worker threads. Seq. execution time is 3.897 frames per sec

3.2 Android Image Filtration

Android Image Filtration is another application that applies various effects such as adding different filters, adjusting color, and adjusting contrast. It is written in pure Java, and it loops through all pixels one after another to alter an image. As updating a pixel is independent of other pixels, we parallelize the application by spawning a task for each row of pixels in the image using the parallel programming constructs (`async, finish`) provided by HJlib.

As can be seen from Figure 2, we achieve a maximum speedup of 2.61× with the parallel version on four worker threads relative to the sequential version of the application.

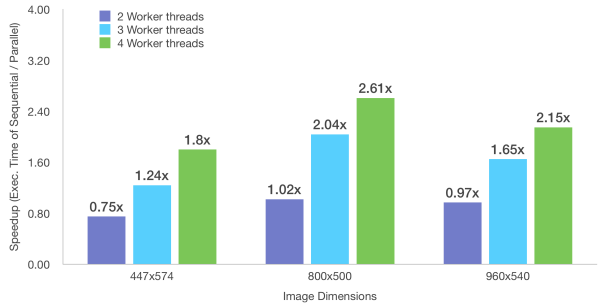

Figure 2: Speedup of the Image Filtration with different numbers of worker threads and varied image size

3.3 Face Replace

FaceReplace is another existing application which considers two input image files (*source, target*) and matches a face (smaller sub image) from *source* to the face on the *target*. The original application was implemented in the earlier Habanero Hava language based on Java 5, as a part of the Parachute Project [4]. In this work, we rewrote the application using HJlib. Figure 3 shows 3× speedup of the application written using the `actors` constructs from HJlib with four threads.

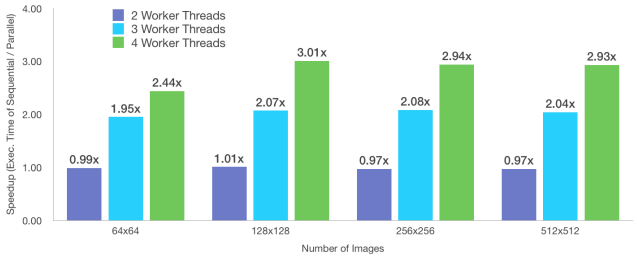

Figure 3: Speedup of the Face Replace application with different numbers of worker threads

4. Conclusions and Future Work

This work is motivated by the observation that modern mobile devices are equipped with heterogeneous and multi-core processors for rich parallelism. In this work, we enable application developers to effectively exploit the parallel computing capabilities of modern mobile platforms while reducing programmer burden. The benefits of our approach are expected to increase as the number of cores in mobile devices increases. In the future, we like to explore using some advanced constructs offered by HJlib such as `barriers, phasers and data-driven tasks` for better performance and scalability of the mobile applications. Exploring heterogeneous workloads, usage of our custom runtime/thread scheduler, and portability of the applications on different mobile architectures are also of future interest.

Acknowledgments

We would like to thank Shams Imam, Max Grossman, and Prasanth Chatarasi for discussions on HJlib, and for their feedback on earlier drafts of this paper.

References

[1] LG Nexus 4 E960 - Full Phone Specs. `http://www.gsmarena.com/lg_nexus_4_e960-5048.php`, 2012.

[2] Habanero extreme scale software research project. `https://wiki.rice.edu/confluence/display/HABANERO/Habanero+Extreme+Scale+Software+Research+Project`, 2014. Accessed: 2014-07-25.

[3] OpenIMAJ:Intelligent Multimedia Analysis in Java. `http://www.openimaj.org/tutorial/parallel-processing.html`, 2014.

[4] Parachute Project. `http://formalverification.cs.utah.edu/parachute`, 2014.

[5] Android Image Manipulations using OpenCV. `http://opencv.org/platforms/android/opencv4android-samples.html`, 2015.

[6] OpenCV. `opencv.org`, 2015.

[7] S. Imam and V. Sarkar. Habanero-Java Library: a Java 8 Framework for Multicore Programming. In *PPPJ'14*. ACM, 2014.