Parallel Computing 57 (2016) 25–36

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Extending Paralldroid with object oriented annotations

Alejandro Acosta*, Sergio Afonso, Francisco Almeida

Department of Computer Engineering, La Laguna University, La Laguna, Spain

ARTICLE INFO

Article history:
Received 18 April 2015
Revised 9 April 2016
Accepted 21 April 2016
Available online 22 April 2016

2010 MSC:
00-01
99-00

Keywords:
Renderscript
Source-to-source transformation
Android

ABSTRACT

The popularity of the handheld systems (smartphones, tablets , ...) and their computational capability open new challenges in terms of the efficient use of such devices. The heterogeneity of these SoCs and MPSoCs demands very specific knowledge of the devices, involving a very high learning curve for the programmers. To ease the development task we build Paralldroid, a framework oriented to general purpose programmers for mobile devices. Paralldroid unifies the Android programming models and allows for the automatic generation of parallel code. Sections of code to be optimized in a Java program can be annotated using Paralldroid annotations. Paralldroid automatically generates the native C or Renderscript code required to take advantage of the underlying parallel platform (GPU included). The code generated by Paralldroid offers a good performance with a very low cost of development, contributing to increased productivity when developing efficient code.

1. Introduction

The evolution of many of today's ubiquitous technologies, such as the Internet, mobile wireless technology, and high definition television has been possible due to the Systems on Chip (SoCs) technology. The information technology age, in turn, has fuelled a global revolution in communications. As a result of this revolution, the computing power of the mobile devices has been increased. The technologies available in desktop computers are now implemented in embedded and mobile devices. In this scenario, we can find that new processors integrating multicore architectures and GPUs are being developed for this market, i.e., the Nvidia Tegra, the Qualcomm snapdragon, the Samsung Exynos and the Texas Instruments OMAP 5.

Many frameworks have been built for software development on such devices, i.e., Windows phone from Microsoft, iOS from Apple and Android from Google. Developing applications for such devices is now easier. Besides to the problem of creating energy-efficient hardware, we stumbled on the difficult task of creating efficient, maintainable programs to run on them.

Conceptually, the architectural model can be viewed as a traditional heterogeneous CPU/GPU system where memory is shared between the CPU and GPU and acts as a high bandwidth communication channel. Memory performance continues to be outpaced by the ever increasing demands of faster processors, multiprocessor cores and parallel architectures.

We find a strong separation among traditional mobile software developers and parallel programmers, the first tend to use high level frameworks like Eclipse or Android Studio for the development of Java programs, without any knowledge of parallel programming (Android: Eclipse + Java, Windows: Visual Studio + C#, iOS: XCode + Objective C), and the latter that use to work on Linux, doing their programs directly in OpenCL closer to the metal. The first take advantage of the high

* Corresponding author.
E-mail address: aacostad@ull.edu.es (A. Acosta).

http://dx.doi.org/10.1016/j.parco.2016.04.003
0167-8191/

level expressiveness while the latter assume the high performance programming challenge. Paralldroid tries to help bring together these two worlds, having in mind the importance of creating a unified programming model as stated in [1].

We propose the Paralldroid system [17,18], a framework that allows the automatic development of Native C and Renderscript applications, sequential and parallel, for mobile devices. Paralldroid contributes to unify the different programming models in Android. Under Paralldroid, the developer fills and annotates, using a sequential high level language, the sections on a template that will be executed using a Native and Renderscript language. In this paper we present the Paralldroid object oriented version that, keeping a unified development model, provides annotation directives for the object oriented paradigm. This simplifies the definition of annotations and eases the use for Java programmers. The generation of code based on annotations directives is well-known [2,3] and in some cases can be used in object oriented languages, like C++. However, the directives used are not well integrated in the object oriented paradigm. Paralldroid can be seen as a proof of concept where we show the benefits of using generation patterns to mask the complexity inherent to parallel programs.

We summarize the main contributions of our proposal:

- To unify the different programming models of Android. Paralldroid generates code from Java to other target languages like Native C or Renderscript. The generation is hidden from the programmer.
- The generation of code for different programming models. The heterogeneity of the Android programming models allows the programmer to obtain the best performance, implementing each section of the application using the programming model that better fits their code. Paralldroid allows to generate code for each programming model, making easier the development of efficient heterogeneous applications.
- The methodology used in Paralldroid to adapt some of the OpenMP directives to the object oriented paradigm is a contribution by itself. Most of the Android developers are Java programmers that apply the object oriented paradigm. Classical OpenMP annotations for C++, for example, are applied to structured blocks inside the sequential algorithm code. Paralldroid introduces the annotations to classes and methods trying to be closer to the object oriented programming paradigm.
- To adapt techniques used in supercomputing systems to handheld systems. The use of compiler directives to generate parallel code is a well-known technique to ease the development of applications in supercomputing systems. In this context we can find some standards like OpenMP or OpenACC. Paralldroid extends the OpenMP standards and generates parallel code for handheld systems.

The paper is structured as follows: Section 2 collects related work and points out issues where Paralldroid could constitute a contribution, in Section 3 we present the Paralldroid Framework and Section 4 analyses the performance obtained with Paralldroid through different use cases and applications from the Renderscript image processing benchmark [8]. The computational results prove the increase of performance provided by Paralldroid at a low cost of development. We finish the paper with some conclusions and future lines of research.

2. Related works

Related work is analyzed in order to evaluate our contributions. In [9] the authors transform Java bytecode to improve the performance of Java code executions. This transformation could be combined with Paralldroid to take advantage of Renderscript executions. Some translators from Java applications to C/C++ code are presented in [10,11]. Paralldroid does not translate a Java application to other languages. Paralldroid generates sections of code annotated by the user integrating it into the Java execution flow as an automatic process hidden to the user. In [12,13] the authors present some OpenMP implementations for Java modifying the Java code to obtain parallel Java Threaded code. The directives used follow the OpenMP C++ specification. They are not well integrated in the object oriented programming paradigm (OOP) and are just applied in an imperative way to the implementation of methods. The directives used by Paralldroid are integrated in the OOP paradigm. Moreover, Paralldroid can be easily extended to generate parallel code using Java Threads or any other target language. The annotation based parallelism is a well-known technique ([2–7]), but annotations are usually applied to the algorithm's code. That approach is more suited to imperative and structured programming, but object-oriented languages offer other possibilities. Paralldroid annotations are applied to the main components of a class and its methods, making the meaning of annotations clearer.

In [14] the authors generate code from OpenCL to Renderscript. In this case the user has to integrate the Renderscript generated code into the Java application. Paralldroid makes this integration automatically and the code is generated from the main Android language: Java. In [15] the authors present a Domain-Specific Language (DSL) to generate Renderscript code. This DSL is specific for image processing algorithms. Paralldroid is not restricted to image processing algorithms. Moreover, Paralldroid is based on the main language of Android, so the user does not need to learn a new language, in contrast to the approach presented in [15].

3. Paralldroid

Android is a Linux based operating system mainly designed for mobile devices such as smartphones and tablet devices, although it is also used in embedded devices like smart TVs and media streamers. It is designed as a software stack that