PORTABLE PERFORMANCE FOR HETEROGENEOUS PARALLEL ARCHITECTURES

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF MASTER OF SCIENCE IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2015

By
Arturo Isaí Castro Pérpuli
School of Computer Science
Abstract

Programming is difficult in a general sense. Tuning for parallel performance adds more complexity to a program. Complexity means more potential bugs and greater difficulty to introduce changes to existing code, reducing its maintainability. Coupling these observations with the fact that most programs today are designed for a particular architecture means that current applications have a limited lifespan.

As hardware continues to evolve, different products will continue to emerge. The current trends in computer architectures dictate that future hardware configurations will contain different kinds of chips, mixing their individual performance capabilities, both for maximising performance for existing programs and for widening the range of applications that can tackle problems where high performance is a decisive factor.

However, creating programs for these heterogeneous architectures with current programming models represents a formidable task. New performance portability techniques are needed. While custom versions of programs for specific architectures might yield better results than programs made with these new techniques, the savings in terms of human effort are undeniable and could mean a much more viable application development life-cycle, with longer intervals between new versions.

The aim of this dissertation is to explore cost-effective (in terms of time and human effort) ways to exploit parallelism targeting heterogeneous architectures with the same code base. A case for portable performance is made and current approaches are evaluated. The key experiments presented are concerned with researching whether it is possible to find an optimal local work size for an OpenCL program that minimises performance loses across heterogeneous architectures, and whether this optimal configuration can be found quickly.

Results show that the methods employed can be used to achieve portable performance on a program without modifying the application’s logic, and that it is possible to reduce the search space required to find an optimal local work size.