MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Framework for Parallel Kernels Autotuning

MASTER'S THESIS

Bc. Filip Petrovič

Brno, Fall 2017

Declaration

Bc. Filip Petrovič

Advisor: RNDr. Jiří Filipovič, Ph.D.

Acknowledgements

I would like to thank my supervisor Jiří Filipovič for his help and valuable advice. I would also like to thank my family for their support during my work on the thesis.

Abstract

The result of this thesis is a framework for autotuning of parallel kernels which are written in either OpenCL or CUDA language. The framework includes advanced functionality such as support for composite kernels and online autotuning. The thesis describes API and internal structure of the framework and presents several examples of its utilization for kernel optimization.

Keywords

autotuning, GPU programming, OpenCL, CUDA, kernel, optimization

Contents

1 Introduction

In recent years, acceleration of complex computations using multi-core processors, graphics cards and other massively parallel devices has become much more common. Currently, there are many devices developed by multiple vendors which differ in hardware architecture, performance and other attributes. In order to ensure portability of code written for particular device, several software APIs such as OpenCL or CUDA were designed. Code written in these APIs can be run on various devices, while always producing the same result. However, there is a problem with portability of performance. For example, code which was optimized for a GPU may run poorly on a regular multi-core processor. The problem also exists among multiple generations of devices developed by the same vendor, even if they have comparable parameters and theoretical performance.

A costly solution to this problem is to manually optimize code for each utilized device. This has several significant disadvantages, such as a necessity to dedicate large amount of resources to write different versions of code and test which one performs best on a given device. Furthermore, new devices are released frequently and in order to efficiently utilize their capabilities, it is often necessary to rewrite old versions of code and repeat the optimization process again.

An alternative solution is a technique called autotuning, where code includes parameters which affect performance depending on their value, for example a parameter which affects length of a vector type of a particular variable. Optimal values of these parameters might differ for various devices based on their hardware capabilities. Parametrized code is then launched repeatedly using different combinations of parameters to find out the best configuration for a particular device.

In order to make the autotuning easier to implement into applications, several frameworks were created. However, large number of these are focused only on a very small subset of computations. There are some frameworks which are more general, but their features are limited and only support simple usage scenarios. The aim of this thesis was to develop autotuning framework which would support more complex use cases, such as situations where computation is split into several smaller programs. Additionally, the framework should be written in a way which would allow its easy integration into existing software stacks and possibly combine autotuning with regular computation.

The thesis is split into four main chapters. <Todo: add short description of each chapter.>

2 Autotuning in parallel compute APIs

This chapter serves as an introduction to autotuning technique and includes description of parallel compute APIs which are utilized by the framework developed in practical part of this thesis - OpenCL and CUDA.

2.1 Parallel compute API

Todo...

3 Conclusion

Todo...

A Appendix

Todo...