Performance-portable algorithmic skeletons for heterogeneous systems

1 Introduction

Computing **systems** have become increasingly **complex** and difficult to program with the emergence of **heterogeneous** hardware. For instance, it is now common to see GPUs (Graphic Processing Units) used for general purpose computation in personal computers, data centres or even supercomputers. As a result, achieving high performance and energy-efficiency for such complex systems is an extremely challenging task.

This project aims at providing **performance portability** for often recurring **parallel patterns** (i.e. *algorithmic skeletons*) on heterogeneous platforms such as GPUs, APUs, or multicore CPUs. To achieve this goal we plan to (1) **automatically generate** implementations for different hardware platforms; (2) explore how **pattern-specific optimisations** can improve performance across different devices and (3) use **machine-learning** models to automatically select the best implementation depending on the hardware characteristics.

2 Programme

To achieve performance portability for algorithmic skeletons we will extend our existing high-level programming model SkelCL[1]. The application developer customizes an skeleton for his application by providing a function which is executed in parallel as defined by the semantic of the skeleton. For example the reduce skeleton performs a parallel reduction of an array to a scalar value by repetitively applying the given customizing binary function. SkelCL provides six skeletons: map, zip, reduce, scan, mapOverlap and allpairs. SkelCL is implemented using OpenCL and currently the skeleton implementations are written manually and then combined with the customizing function provided by the application developer to form an OpenCL kernel. The performance of skeleton implementations is currently not portable across devices, as optimizations are done manually for a specific device architecture. By automatically generating different optimized versions of the skeleton implementation we want to explore the possible implementation design space and, thus, generate versions which perform well on each target hardware platform. To this end, we will combine SkelCL with the LLVM-based OpenCL source-to-source compiler developed by the CArD group at Edinburgh.

By using algorithmic skeletons we have **semantic information** about the implementation available which will **guide our optimisations**. For example the semantic knowledge of a parallel reduction can be used to apply more aggressive optimizations than would not otherwise be possible. Another example is the newly introduced allpairs skeleton: it performs computations on all pairs of vectors of two matrices and can be used for matrix multiplication or n-body simulations. Information about the memory access pattern can be exploited in its implementation to automatically utilize faster shared memories if available. Other possible optimisations include: vectorisation, granularity of parallelism or loop unrolling.

In general it is unclear which version performs best on a given hardware platform, therefore, we will employ **machine learning** techniques **to automatically select the best** performing version. We will extract characteristics from the source code and hardware, and use them as features into the machine learning model to predict the best performing implementation. We will benefit from our previous experience in this field (see Dubach et. al MICRO'09).

3 Potential Output

We want to directly integrate all work in the existing SkelCL library and, thereby, provide a useful contribution to research and practice. All work performed during the project will be available as open source software, just as the SkelCL library is already. For this three month project we will deliberately restrict ourself to develop the techniques for two algorithmic skeletons: reduce and allpairs. After the three month period we will prepare a publication of our results in a joint publication. We target a publication at the PLDI conference (deadline November 2013).

[1]`http://skelcl.uni-muenster.de`