Karl Rupp
Argonne National Laboratory, US

ViennaCL - Portable High Performance at High Convenience

Minisymposium Session PARA: Monday, 12:10 - 12:40, CO016

High-level application programming interfaces (API) are said to be the natural enemy of performance. Even though suitable programming techniques as well as just-in-time compilation approaches have been developed in order to overcome most of these limitations, the advent of general purpose computations on graphics processing units (GPUs) has lead to a renaissance of a wide-spread use of low-level programming languages such as CUDA and OpenCL.
Porting existing code to GPUs is, however, in many cases a very time consuming process if low-level programming languages are used. They require the programmer to understand many details of the underlying hardware and often consume a larger amount of development time than what is saved by a reduced total execution time. On the other hand, high-level libraries for GPU computing can significantly reduce the porting effort without changing too much of existing code.
ViennaCL is one of the most widely used library offering a high-level C++ API for linear algebra operations on multi-core CPUs and many-core architectures such as GPUs. In particular, we demonstrate that a high-level API for linear algebra operations can still be provided without sacrificing performance on GPUs. Furthermore, the generic implementations of algorithms such as iterative solvers allow for code reuse beyond device and library boundaries, making the transition from purely CPU-based code to GPU-accelerated code as seamlessly as possible. Also, we explain why and how ViennaCL manages different parallel computing backends and assess the role of autotuning for achieving portable performance. Benchmark results for GPUs from NVIDIA and AMD as well as for Intel's MIC platform are presented along with a discussion of techniques for achieving portable high performance.

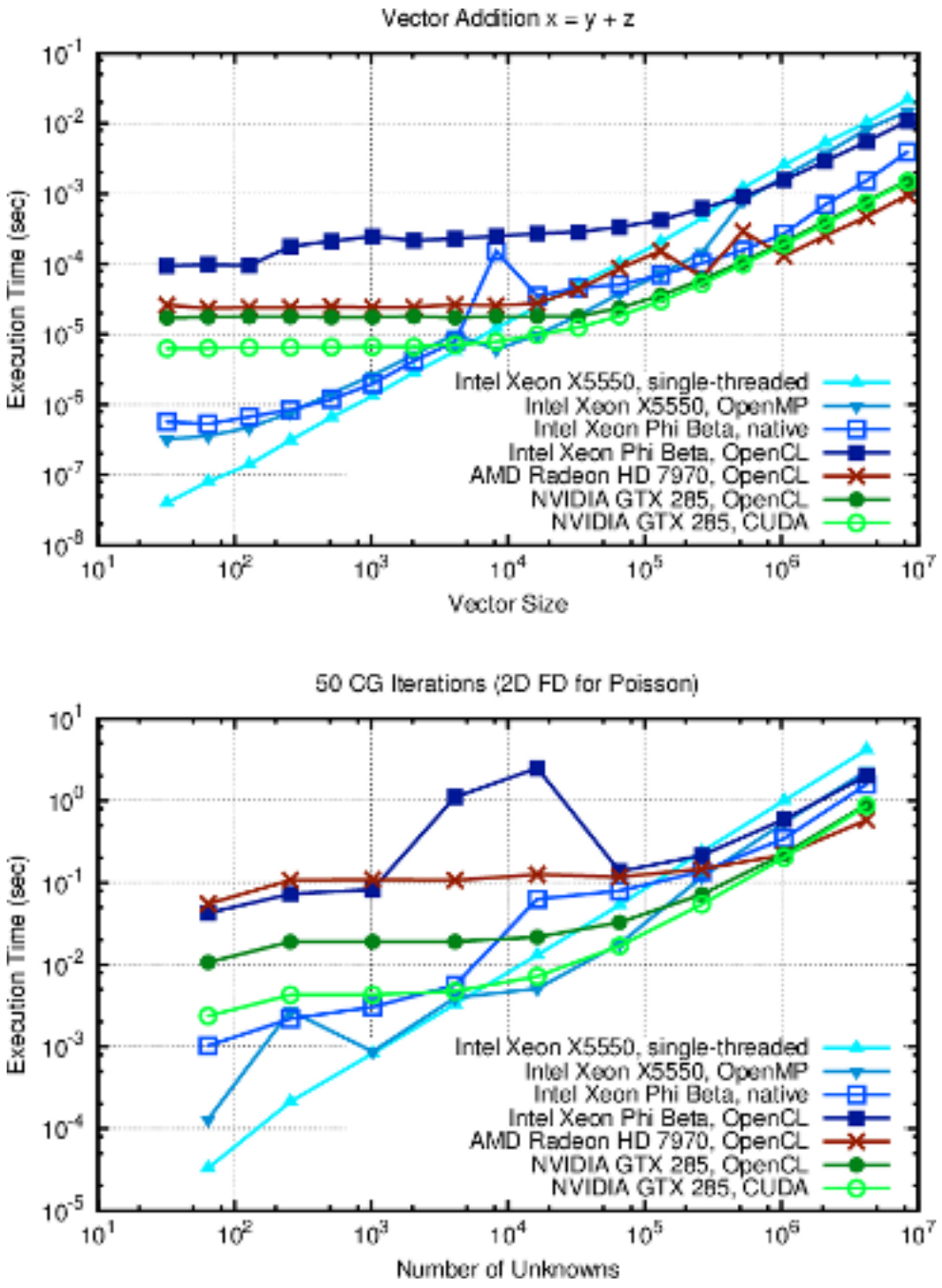

Figure 1: STREAM-like benchmark for the performance of vector additions (top) and performance comparison for 50 iterations of the conjugate gradient method (bottom) on different computing hardware.

Joint work with Philippe Tillet, Florian Rudolf, and Josef Weinbub.