On behalf of DoWiR and ITMC we would like to announce a 2-day training course "OpenACC" at the TU Dortmund (9-10 May 2017)

CUDA and OpenCL are two programming languages that allow to develop GPU applications with great level of detail, but require advanced knowledge of GPU architecture and extensive code re-writing. Many cross-platform or prototype applications do not actually require deep GPU porting, if they perform data processing using standard algorithms, e.g. reduction. In such cases, fair performance numbers could be achieved with simplified directive-based language extensions. OpenACC is de facto an industry standard for C/Fortran directive extensions for porting code to GPUs. This training course walks through OpenACC programming technology, from basics to advanced practices used in real applications.

Hands-ons: All discussed topics will be accompanied with practical sessions, using PGI OpenACC compiler with NVIDIA backend. Exercises will be conducted either on the provided remote GPU server or on the customer’s local system.

All corresponding presentations and code samples will be available to attendees from the beginning of each training day.

Applied Parallel Computing LLC is delivering GPU training courses since 2009. Several dozens of courses have been organized all over Europe, both for commercial and academic customers. We work in close partnership with NVIDIA, CUDA Centers of Excellence and Tesla Preferred Partners. In addition to trainings, our company provides GPU porting/optimization services and CUDA certification.

Day 1: Introduction to OpenACC

Morning (09:00-13:30)

09:00-10:00: introduction (Dell & DoWiR)
10:00-11:30: lecture

- Advantages of OpenACC.
- Execution model: gangs, workers and vectors – three levels of coarse-grain and fine-grain parallelism. SIMD instructions.
- OpenACC memory model: host and accelerator address spaces.
- OpenACC directive syntax in C and Fortran. Main directives: parallel and kernels – offloading code regions to accelerator, loop – detailed parallelization parameters for each loop.
- Examples of vector addition and reduction explained.

12:00-13:30: hands-on session

- PGI compiler for NVIDIA GPUs. GCC open-source OpenACC compiler.
- Compilation and deployment of OpenACC code examples.

Afternoon (14:30-18:00)

14:30-16:00: lecture

- Understanding PGI OpenACC compiler output. Compiler flags and environment variables for detailed analysis and performance reports. Compiler limitations in dependencies tracking, enforcing data independence.
- Data clauses: deviceptr, copy, copyin, copyout, create, delete, present, present_or_(-copy, -copyin, -copyout, -create).
- Organizing data persistence regions using data directives.

16:30-18:00: hands-on session

- Profiling GPU kernels in OpenACC application. Time option, PGI_ACC_TIME environment variable. Profiling with pgprof.
Day 2: Advanced OpenACC programming

Morning (10:00-13:30)

10:00-11:30: lecture

- Non-structured data lifetime with enter data and exit data directives.
- Additional data management directives: cache, update, declare.
- Expressing locality in loop nest with tile directive.
- Organizing asynchronous execution using async clause and wait directive.
- Comparison of OpenACC and OpenMP 4 main directives: parallel and kernels vs parallel and target
- Comparison of OpenACC and OpenMP 4 data clauses: copy, copyin, copyout vs map and other directives
- Comparison of OpenACC and OpenMP 4 data persistence directives: data vs target data and declare target
- Getting LLVM-based OpenMP 4 compiler to work on NVIDIA GPUs
- Side by side comparison of vector addition, reduction and wave13pt examples in OpenACC and OpenMP 4.

12:00-13:30: hands-on session

- Using non-structured data lifetime directives
- Example of OpenACC loop tiling, effect on performance
- Advanced code optimization practices, by examples. Restrictions, common mistakes, workarounds.
- OpenACC interoperation with CUDA C, CUDA Fortran and GPU-enabled libraries.

Afternoon (14:30-18:00)

14:30-15:30: lecture

- External dependencies in OpenACC kernels, functions inlining. OpenACC routine directive, separate compilation and procedure calls.
- Accessing global variables.

15:30-16:30: lecture

- Scaling OpenACC across multiple GPUs
- GPU context switching in OpenACC: acc_set_device_num
- Accessing multiple GPUs from a single process/thread with OpenACC
- Accessing multiple GPUs from OpenMP application with OpenACC
- Accessing multiple GPUs from MPI application with OpenACC

17:00-18:00: hands-on session

- Reimplement CUFFT + CPU version of 2D Poisson solver into efficient CUFFT + OpenACC version.

17:30-18:00: Q & A

Prerequisites

- Beamer
- Chrome or Firefox web browser
- (optional) Open (unfirewalled) connection to farm.parallel-computing.pro (188.44.42.33) over port 22
- (optional) PuTTY SSH client