Applied Parallel Computing LLC
http://parallel-computing.pro
+41 78 925 90 90
contact@parallel-computing.pro

2-day OpenACC training course at Universität Potsdam

(21–22 October 2019)

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. As a case study, the course presents a practical session on using OpenACC for solving 2D Poisson problem with FFT.

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 will be available to attendees in printed handouts.

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 (10:00-13:30)

10:00-11:15: lecture

- An overview of GPU performance in various applications
- Brief intercomparison of different types of accelerators
- Key programming principles to achieve high GPU performance

11:15-11:30: coffee break
11:30-12:30: lecture

- CUDA principles and CUDA implementation for C++
- Analogies between MPI+OpenMP and CUDA programming models
- The first CUDA program explained
- CUDA compute grid, examples
- Realistic CUDA application example (wave propagation code)
- Understanding GPU compute capabilities, *deviceQuery*
- Basic optimization techniques
- Overview of CUDA applications development on a cloud platform

12:30-13:30: lecture

- Advantages of OpenACC in comparison to CUDA
- 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, in comparison to CUDA

Afternoon (14:30-18:00)

14:30-16:00: hands-on session

- PGI compiler for NVIDIA GPUs
- Compilation and deployment of CUDA and OpenACC code examples

16:00-16:20: coffee break
16:20-18:00: 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
- Profiling GPU kernels in OpenACC application. *Time* option, *PGI_ACC_TIME* environment variable. Profiling with *pgprof*
- Data clauses: *deviceptr*, *copy*, *copyin*, *copyout*, *create*, *delete*, *present*, *present_or_(-copy, -copyin, -copyout, -create)*
- Organizing data persistence regions using *data* directives

Day 2: Advanced OpenACC programming

Morning (09:00-12:30)

09:00-10:00: hands-on session

- "Fill-in" exercise on implementing wave propagation stencil in OpenACC (*wave13pt*). Adding OpenACC directives step by step

10:00-10:30: coffee break

10:30-11:30: 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
- *Atomic* directive. Allowed operations. Restrictions

11:30-12:30: lecture

- Using non-structured data lifetime directives
- Example of OpenACC loop tiling, effect on performance
- Advanced code optimization practices, by examples. Restrictions, common mistakes, workarounds
- External dependencies in OpenACC kernels, functions inlining. OpenACC *routine* directive, separate compilation and procedure calls
- Accessing global variables

Afternoon (13:30-17:00)

13:30-14:30: lecture

- GPU-enabled scientific libraries (part 1)

14:30-15:30: lecture

- GPU-enabled scientific libraries (part 2)
- OpenACC interoperation with CUDA and GPU-enabled libraries
- OpenACC interoperation with Python using *pybind11*
- Using PGI OpenACC compiler as an OpenMPI backend
- Advantages of GPUDirect for a GPU-enabled MPI application

15:30-15:50: coffee break
15:50-17:00: hands-on session

- "Fill-in" exercise on reimplementing CUFFT + CPU version of 2D Poisson solver into efficient CUFFT + OpenACC version.

Prerequisites

- Beamer
- Client laptop or classroom computer with Chrome/Firefox browser