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

3-day Course on GPU Computing at Rohde & Schwarz GmbH

(03–05 July 2017)

All corresponding presentations and code samples will be available to attendees in printed handouts.

Day 1: Introduction to CUDA and GPU libraries

Morning (09:00-12:30)

09:00-10:30: 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 using Visual Studio 2015

10:45-12:15: Hands-on session

- Example of *vector addition* in CUDA, compared to OpenACC implementation
- **Hands-on:** Write & deploy a simple CUDA program
- **Hands-on:** More control on CUDA compute grid

12:30-14:00: Hands-on session

- **Hands-on:** Write & deploy a meaningful image processing tool in CUDA

14:00-15:30: Lunch

15:30-16:30: GPU-enabled libraries

- Thrust – the C++ library of GPU-enabled parallel algorithms
- CUBLAS, MAGMA, CUBLAS-XT, CUSPARSE, CUFFT and CURAND
- CUSP and AmgX – Krylov and mutigrid solvers
- CUDNN – Deep Neural Network library

16:45-18:00: Hands-on session

- **Hands-on:** solving Poisson equation with CUFFT

Day 2: GPU memory hierarchy, advanced CUDA, optimization & profiling

09:00-10:30: GPU memory hierarchy

- GPU memory types
- Shared memory
- GPU caches hierarchy and mode switches
- Automatic texture cache (Kepler GK110)
- Unified virtual address space (UVA) in CUDA 7.5
- Streams and asynchronous data transfers

10:45-12:15: Hands-on session

- **Hands-on:** "fill-in" exercise on reduction with and without shared memory
- **Hands-on:** getting additional performance using automatic texture cache

12:30-14:00: Advanced CUDA

- Dynamic parallelism
- Dynamic memory allocation in CUDA threads
- Compiling & linking relocatable device code
- CUDA C++ compiler pipeline, PTX assembler, SASS
- Understanding "-Xptxas -v" reports

14:00-15:30: Lunch

15:30-16:30: GPU code optimization

- PCI-E optimizations: streams, asynchronous data transfers
- An overview of Fermi, Kepler and Maxwell GPU architectures
- GPU optimizations: compute grid, coalescing, divergence, unrolling, vectorization, maxrregcount, aligning, floating-point constants
- Overview of *NVIDIA Visual Profiler*
- Overview of *nvprof* (command line profiler)
- Common practices of identifying performance hazards in GPU application using NVIDIA Visual Profiler

16:45-18:00: Hands-on session

- **Hands-on:** profile and optimize the bilinear interpolation kernel

Day 3: GPU debugging & Message Passing Interface (MPI)

09:00-10:00: GPU debugging

- Principles and terminology
- GNU Debugger (*gdb*)
- CUDA-enabled GNU Debugger (*cuda-gdb*)

- GPU memory checker (*cuda-memcheck*)
- Debugging SASS without the source code

10:00-10:30: Hands-on session

- **Hands-on:** live demonstration of *cuda-gdb* debugger on a sample application

10:45-12:15: MPI Overview

- The Message-Passing Programming Paradigm
- Data and Work Distribution
- MPI messages
  - Access and addressing in message passing system
  - Point-to-Point Communication
  - Collective Communications

12:30-14:00: Process model and language bindings

- MPI Forum
- Goals and Scope of MPI
- MPI Header files and MPI Function Format
- Initializing MPI
- Starting the MPI Program
- Communicator *MPI_COMM_WORLD*
- Handles identifying MPI objects
- MPI rank and communicator size
- Exiting MPI

14:00-15:30: Lunch

15:30-16:30: Messages and point-to-point communication

- MPI messages, basic and derived datatypes
- MPI Basic Datatypes
- The concept of point-to-point communication
- Sending a Message: *MPI_Send*
- Sending a Message: *MPI_Ssend*
- Receiving a Message: *MPI_Recv*
- Requirements for Point-to-Point Communications
- Wildcarding receiver
- Communication Modes

16:45-18:00: Hands-on session

- **Hands-on:** Using CUDA in MPI applications: single and multiple GPUs
- **Hands-on:** Inter-GPU data message passing with CUDA-aware MPI