



"APPLIED PARALLEL COMPUTING"
Education and Research Center



NVIDIA Advanced CUDA Programming Course

Description

The Institute of Aerodynamics and Fluid Mechanics at TU München announces in cooperation with NVIDIA (www.nvidia.com), FluidDyna GmbH (www.fluidyna.com) and the Applied Parallel Computing E&R Center (www.parallel-compute.com) a closed CUDA training for advanced software developers. The training is free of charge and dedicated to researchers with basic experiences in programming own CUDA codes. Main focuses will be the development of multi-GPU applications and the addressing of performance issues by debugging and optimizations techniques.

Organization

Training identifier:	TUM-AER-WB-2012-01
Requirements:	basic programming skills in CUDA
Participants:	max. 25
Date:	January, 16-18, 2012 (3 days)
Hours:	09:00am – 05.00pm
Place:	Technische Universität München Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Boltzmannstraße 1, 85748 Garching bei München, Germany, Room H.U.010
Costs:	none
Trainer:	Dmitry Mikushin
Registration:	via eMail to: Thomas.Indinger@tum.de

Course Plan

1. From GPU to GPGPU
 - 1.1 Performance and parallelism
 - 1.2 GPU evolution
 - 1.3 Parallel systems: multicore and clustering

2. CUDA programming model

- 2.1 Key principles
- 2.2 Threads and blocks
- 2.3 Language extensions
 - 2.3.1. Attributes
 - 2.3.2. Builtin types and variables
 - 2.3.3. Kernel invocation operator
- 2.4 CUDA runtime API
 - 2.4.1. Asynchronous execution
 - 2.4.2. Handling runtime errors in CUDA
 - 2.4.3. Querying GPU capabilities

3. Memory hierarchy

- 3.1 Global memory
 - 3.1.1. Example: matrix multiplication
 - 3.1.2. Optimizing global memory usage
- 3.2 Block-shared memory
 - 3.2.1. Example: matrix multiplication
 - 3.2.2. Shared memory access patterns
- 3.3 Constant memory
- 3.4 Texture memory
- 3.5 Unified virtual address space (UVA)

4. Implementing basic data processing

- 4.1 Parallel reduction
- 4.2 Prefix sum (scan)
 - 4.2.1. CUDA implementation
 - 4.2.2. CUDPP implementation

5. CUDA Libraries

- 5.1 CUBLAS
- 5.2 CUSPARSE
- 5.3 CUFFT
- 5.4 CURAND

6. Thrust

- 6.1 Thrust: transform for Vector Addition
- 6.2 SAXPY Implementation Using Functors
- 6.3 Reduction
- 6.4 Performance considerations
- 6.5 Thrust and CUDA C interoperability
- 6.6 Example: Total Rainfall at Each Site

7. Using multiple GPUs

- 7.1 CUDA context
- 7.2 fork
- 7.3 MPI
- 7.4 POSIX-threads
- 7.5 OpenMP
- 7.6 Boost.Threads

8. CUDA Streams

- 8.1 Example: concurrent kernels execution
- 8.2 Example: matrix multiplication
- 8.3 Example: Multi-GPU Async Copy

9. Debugging

- 9.1 Principles and terminology
- 9.2 gdb
- 9.3 cuda-gdb
- 9.4 CUDA (Visual) Profiler
- 9.5 cuda-memcheck

10. Optimization Techniques

Hands-on Session: basic CUDA programming, shared memory utilization, profiling, CUDA Libraries, Thrust, MultiGPU, CUDA Streams, debugging.