GPU Computing Training
8 Aug.–2 Sep., 2011
Instructor: Christopher Cooper, PhD candidate at Boston University
Organizers: Prof Luis Salinas (UTFSM), Prof Lorena Barba (Boston University)

Week 1
Class 1:
- Understand the need for multi-core in applications
- Manycore architecture:
  - GPU vs CPU chip design
  - Data parallelism
  - Concepts behind a CUDA-friendly algorithm
- Basic CUDA:
  - C-like language
  - Threads
  - Launching a CUDA kernel

Lab 1:
- Device query
- Familiarize yourself with CUDA
- Launch a simple vector add
- Implement an axpy
- Implement a matrix matrix multiplication

References:
- Kirk, D. and Hwu, W. Programming Massively Parallel Processors. Ch. 1-4

Class 2:
- Fundamentals of the finite difference method
- Programming model: mapping the discretized model to the GPU threads
- Multilevel memory hierarchy
  - Shared, global, registers, textures, constant, texture memories
  - Sizes and latency
  - Blocks
- Finite difference implementations using CUDA:
  - Global memory
  - Texture
  - Shared and global
  - Only shared

Lab 2:
- Implementation of 2D explicit heat transfer with global memory
- Implementation of 2D explicit heat transfer with texture memory
- Comparison of each: timings vs. programming effort

References:
- Kirk, D. and Hwu, W. Programming Massively Parallel Processors. Ch. 5
- Micikevicious P. 3D Finite Difference Computations on GPUs using CUDA
Class 3:
- Efficient memory usage:
  - Warp and SIMD
  - Warp divergence: if statement
  - Kernel memory access
  - Coalescing
  - Memory bandwidth: measuring effective performance
- Reducing global memory access:
  - Effective use of shared memory and registers
  - Tiling: shared memory as cache
  - Using FD examples to explain
  - Discussion on implementations in 1D, 2D and 3D

References:
- Kirk, D. and Hwu, W. Programming Massively Parallel Processors. Ch. 5
- Micikevicious P. 3D Finite Difference Computations on GPUs using CUDA

Week 2
Class 1:
- Holiday

Lab 1:
- Using shared memory as cache:
  - Implement 2D explicit heat transfer using shared memory
  - Comparison of each: timings vs. programming effort

References:
- Kirk, D. and Hwu, W. Programming Massively Parallel Processors. Ch. 5
- CUDA C Best Practices Guide. Ch. 2-6.

Class 2:
- Efficient use of shared memory:
  - Bank conflicts
  - Occupancy
  - Further recommendations:
    - Blocks per thread heuristics
    - Latency hiding
  - Discussion on memory as limiting factor to parallelism
  - Instruction optimization

Lab 2:
- Matrix transpose example

References:
- Kirk, D. and Hwu, W. Programming Massively Parallel Processors. Ch. 5
- CUDA C Best Practices Guide. Ch. 2-6.

Class 3:
- Further optimization techniques:
  - Streaming multiprocessor partitioning
  - Data prefetching
  - Instruction mix
  - Thread granularity
  - Loop unrolling
- Parallel algorithms: scan, reduce

References:
- Kirk, D. and Hwu, W. Programming Massively Parallel Processors. Ch. 5

Week 3
Class 1:
- Thrust library: STL for CUDA
  - Overview of features:
    - Transformations
    - Reduction
    - Prefix sums
    - Reordering
    - Sorting
    - Iterators
  - Using Thrust in your CUDA kernels: raw pointer cast

Lab 1:
- Do examples using thrust:
  - Sum, sort, fill, sequence, axpy, etc.
- Change a code done in week 1 to use Thrust
- A useful profiling tool: Valgrind

References:
- http://code.google.com/p/thrust/
- http://valgrind.org/

Class 2:
- Cusp library: Sparse matrix algebra in CUDA
  - Overview of sparse matrices
  - Matrices in Cusp: types and formats
  - Solvers: CG, BiCG, GMRES, etc.
  - Preconditioners

Lab 2:
- Implement a 2D Poisson solver with Cusp
- Generating a Cusp matrix - show example
- Compare timings with different:
  - Solvers
  - Preconditioners
  - Matrix formats
  - Memory space use (CPU vs GPU)
- Implement a sparse matvec

References:
Class 3:
- Good practices within Thrust and Cusp
- Using Cusp and Thrust for “cleaner” codes -> raw pointer cast

Week 4
Class 1:
- The role of sparse matrices in CFD:
  - Importance of implicit schemes
- Challenges of GPU implementations in CFD codes
  - New CUDA codes need to replace old finely tuned CPU implementations
- Example using implicit FD: from the mesh to a matrix in a Poisson problem
- Explanation of the 2D implicit heat transfer using Crank Nicolson in CUDA

Lab 1:
- 2D implicit heat transfer (Crank Nicolson)
- Use 2D stencil codes and Cusp

References:
- IBM on GPU paper

Class 2:
- N-body simulation with GPU
- Revisit mat-vec on GPU and relate it to the physical problem
- Example of applications:
  - Astrophysics
  - Surface reconstruction with RBF
  - Others (MD)
- Revisit optimizations:
  - Shared memory use and tiling
  - Loop unrolling and other optimization techniques

Lab 2:
- Implementation of a fast N-body simulation

References: