OpenCL™ Programming Heterogeneous Systems

Course Description

OpenCL programming heterogeneous systems training provides a comprehensive introduction to the OpenCL standard for heterogeneous parallel computing.
The training offers a detailed walk-through of the different models defined by the OpenCL standard interleaved with specific examples to demonstrate the concepts.
The focus is on GPU devices, and best practices will be discussed such as how to best suit an algorithm to GPU and optimization guidelines.

The training is based on the OpenCL 1.2 specification.

Course Duration

2 days (3 with hands-on labs)

Target Audience

Software engineers with no, or little, OpenCL background, which about to embark on the development of an OpenCL project.

Course Material

- Course book

Agenda

Main Topics:

- Introduction to Parallel Computing
- Introduction to OpenCL
- The Anatomy of OpenCL Applications
- Platforms & Devices
- Data Parallelism
- The OpenCL Memory Model
- Command Synchronization & Profiling
- The OpenCL-C Language
- OpenCL Images
- Best Practices

Day #1

- Parallel computing – introduction
  - Flinn's taxonomy
  - Parallel vs. Concurrent
  - Task and Data parallelism
  - What type of a processor is a GPU
  - Mapping tasks to a suitable processor

- Introduction to OpenCL
  - What is OpenCL
  - How OpenCL came to be
  - OpenCL models

- First code – The anatomy of OpenCL applications
  - The platform model
    - Schematic view of an heterogeneous system
    - Querying for platforms and devices
  - The execution model
    - The structure of an OpenCL application

        - Creating a context
        - Creating command-queues
        - Creating OpenCL programs and kernels
            - Writing OpenCL-C code
            - Building the code
        - Creating memory objects
        - Data transfers between host processor and OpenCL device
        - Executing kernels on the device
        - Cleanup

- Platforms and devices
    - Mapping available OpenCL platforms and computation devices
    - Retrieving information about the devices
    - Survey of interesting device information

- Data-parallelism (and some complimentary information)
    - Spawning an ND-Range for parallel execution of work-items
        - What is an ND-Range
        - What are work-items and work-groups
        - Indexing work-items and work-groups
    - Mapping work-items and work-groups onto GPU shader cores
    - Parallelizing example from first code
    - Execution model concepts re-visited
        - Context creation and information retrieval
        - Command-queue types
        - Types of commands
        - The OpenCL-C compiler
            - Compiler flags
            - Resolving compilation issues
            - Retrieving compiled binaries
    - New example: large matrix transpose

- The OpenCL memory model
    - Memory regions and accessibility
    - Mapping the OpenCL memory model to GPU architecture

Day #2

- Command synchronization and profiling
  - Synchronization levels
  - Coarse synchronization
    - Queue flushing
    - Barriers
  - Fine-grained synchronization
    - Creating event objects
    - Querying for event information
    - Markers
    - Synchronizing host application with command-queue
    - User events
    - Event callbacks
  - Profiling commands using events

- The OpenCL-C language
  - Restriction and extensions over C99
  - Data-types
  - Working with vector literals and elements
  - Host-side type referencing
  - Qualifiers
    - Function qualifiers
    - Address space qualifiers
    - Accessing qualifiers
    - Type qualifiers
  - Pre-processor directives overview
  - Overview of built-in functions
    - A word on math functions with emphasis GPU

- OpenCL images
  - What are images
  - Specialized hardware facilities in GPUs
    - Spatial locality aware cache
    - Linear interpolation of adjacent values
    - Out of bounds accesses
  - Working with images
    - Creating images

- Image element type and arrangement
- Writing/reading data to/from images on the host
- Image sampler objects
- Writing/reading data to/from images in the kernel code

- Best practices
  - What algorithms best suit GPU implementation
  - Optimization guidelines
    - Establishing an optimization strategy
      - Profiling and analyzing kernels
      - Load balancing between GPU and host processor
      - Memory sharing between GPU and host processor
    - Optimization checklist
      - General optimizations
      - Memory optimizations
      - ND-Range optimizations
      - Kernel code optimizations
      - Host application execution optimizations