1-512-256-0197
www.mindshare.com
training@mindshare.com

Comprehensive OpenCL Programming for GPU and Multicore Architectures

Let MindShare Bring "OpenCL Programming for GPU and Multicore Architectures" to Life for You

OpenCL allows parallel computing on heterogeneous devices such as combinations of multicore CPUs, GPUs, and other hardware accelerators (DSPs, etc) and is emerging as a primary programming framework for high-performance computing, mobile computing, and 3D graphics domains.

This course provides hands-on programming experience and explains the development environment and programming techniques required for developing general-purpose OpenCL software applications for GPU and multicore hardware.

This course introduces the programming techniques required to develop general-purpose OpenCL applications for GPU and multicore processors. It examines various components of the OpenCL development framework. Special attention is focused on understanding the performance characteristics of modern GPU/multicore platforms including architecture details: Intel Nehalem (Core i7), NVidia GT/GTX series, NVidia Fermi, AMD Fusion (APUs), and IBM Cell. The course organization provides theory and practice of parallel programming techniques with OpenCL. Elements and examples of NVidia CUDA programming model are included. The CUDA programming model and OpenCL share key abstractions of threads, thread blocks, grids of thread blocks, barrier synchronization, per-block shared memory, global memory, and atomic operations.

You Will Learn:

- How to design OpenCL applications for heterogeneous computing platforms.
- How to use OpenCL to develop portable code for co-processing devices such as GPUs and Multicore architectures

Course Length: 4 days

Who Should Attend?

The target audience is the advanced C/C++ developer with little or no knowledge of OpenCL programming, notions of multithreading programming or parallel hardware. This course is designed for library and middleware developers and performance-oriented application developers targeting heterogeneous computing architectures that utilize CPUs, GPUs and other co-processing devices.

Course Contents:

- **Introduction to the OpenCL Architecture**
  - Heterogenous programming
  - Multicore and GPU architecture models and trends
  - OpenCL platform, execution, memory, and programming model

- **OpenCL Framework: Platform, Runtime, and Compiler**
  - OpenCL terminology and OpenCL programming language, library API, and data types
  - Platform layer and device objects
  - Contexts - Environment for managing OpenCL objects and resources
  - Runtime: command queues, memory commands (data transfers: asynchronous/event-based), memory objects (buffers, images, program objects, kernel objects, executing

1-800-633-1440
www.mindshare.com
training@mindshare.com

kernels
- Event objects, out-of-order execution of kernels

- **Developing OpenCL Programs**
  - Data-level and task-level parallel problem domains
  - Threading (work-items) and blocking (work-groups), sharing data, barriers, and synchronization
  - Programming multiple devices
  - OpenCL numerical compliance and embedded specifications/profile
  - OpenCL extensions – AMD, NVidia
  -
- **GPU and Multicore Architecture Performance Analysis**
  - Case studies of specific devices provided: performance results
  - Memory (latency, registers, local, shared, global) and access patterns (bank conflicts, coalescing)
  - Warps/Wavefronts, scheduling, and GPU cores
  - Single and double precision support
  - Profiling and event collection/analysis
  - Code design guidelines

Laboratory Experience: Designing Efficient OpenCL Kernels

Hands-on examples are meant to provide deep understanding of parallel execution as well as advanced OpenCL concepts (multiple contexts & multiple devices). Learning outcomes include having participants able to develop efficient OpenCL modules, understand GPU/multicore resource limits/constraints, work with existing OpenCL infrastructure/libraries, measure OpenCL code performance, detect performance bottlenecks, and tune code solutions to specific hardware resources.

Three levels of ~12 custom OpenCL lab examples are delivered:
- Level1: Memory Bandwidth Analysis, Vector Addition, Parallel Reduction/Scan, Parallel Sort
- Level2: Convolution/Sobel-Filter, Matrix-Vector Product (GEMV), Matrix Product (SGEMM, DGEMM),
- Level3: FFT (Fast-Fourier Transform), N-Body Simulation, SIFT (Feature Transform), Image Analysis

Programming activities explored include:
- Compute-bound and memory-bound tasks
- Performance impact of device memory transfers and efficient/inefficient memory access patterns
- Performance impact of work-group specification
- Reformulation examples to expose/increase data-level parallelism
- Debugging tools and profiling techniques for performance/timing/event analysis
- Single context, multiple devices (standard way to work with multiple devices in OpenCL)
- Multiple contexts, multiple devices (computing on a cluster, multiple systems, etc).

The laboratory experience also includes guided coverage of the full OpenCL SDK (AMD, NVidia). Laboratory environment: Linux or Windows. Microsoft DirectCompute (DirectX) is not examined.

Select Examples of Laboratory Experiences Provided for Course Consideration

- Lab 1: Introduction to OpenCL Framwork and Kernel Design– Vector Addition
  - Setting up a Context: Platform, Device, and Command Queues
  - Context Memory Objects: Buffer, Transferring data
  - OpenCL compilation and runtime execution
  - Evaluating performance with timers, evaluating arithmetic intensity and memory bandwidth

1-800-633-1440
www.mindshare.com
training@mindshare.com

- Lab 5: OpenCL Specification of 2D Work-Groups with Shared Memory – Sorbel Filtering
    - Shared (local) memory allocation, reformulation of work, variation of workgroup assignment
    - Event/command synchronization

Recommended Prerequisites:

Previous programming experience with C/C++. Familiarity with parallel programming concepts such as task parallelism and domain decomposition.

Equipment:

Students must bring their GPU-based laptop running Linux to class.

Course Material:

Mindshare will supply electronic version of the presentation slides including the lab descriptions and source code referenced in the examples and lab exercises.