Advanced OpenCL by Example

Justin Hensley, Derek Gerstmann & Takahiro Harada
Course Agenda

OpenCL Review

OpenCL Development Tips

OpenGL/OpenCL Interoperability

Rigid body particle simulation

<15 minute Break>

Bullet cloth

Galaxy n-body simulation

Visualization with OpenCL
Slides from tutorial and course

http://sa10.idav.ucdavis.edu
GPU Basics

Architectures meant for **throughput**, not for **single thread**
- Not complexity of out-of-order issue
- Nor of multiple issue per core

Rendering is **latency tolerant**

Graphics generates many **parallelizable** tasks
- Therefore much easier to scale
The GPU is a multicore processor optimized for graphics workloads
Modern AMD GPU

Command Processor/Group Generator

Sequencer

GDS

Sequencer

Ins cache

Rd cache Crossbar

SIMD Engines 0-9

SIMD Engines 10-19

Ins cache

Write crossbar

Write combine caches
R/W cache for global atomics
Write combine caches
R/W cache for global atomics

Read L2 caches
8kB/mem channel

R/W cache for global atomics

8 channel memory controller

GDDR5  GDDR5  GDDR5  GDDR5  GDDR5  GDDR5  GDDR5  GDDR5  GDDR5
Industry Standard for Programming Heterogeneous Platforms

CPU:
- Multiple cores driving performance increases
- Multi-processor programming – e.g. OpenMP

GPU:
- Increasingly general purpose data-parallel computing
- Graphics APIs and Shading Languages

Emerging Intersection

OpenCL – Open Computing Language
- Open, royalty-free standard for portable, parallel programming of heterogeneous parallel computing CPUs, GPUs, and other processors
OpenCL Platform Model

- **One Host + one or more Compute Devices**
  - Each Compute Device is composed of one or more **Compute Units**
    - Each Compute Unit is further divided into one or more **Processing Elements**
An N-dimension domain of work-items

- Global Dimensions: 1024 x 1024 (whole problem space)
- Local Dimensions: 128 x 128 (work group ... executes together)

Synchronization between work-items possible only within workgroups: barriers and memory fences

Cannot synchronize outside of a workgroup

Choose the dimensions that are “best” for your algorithm
OpenCL Memory Model

- **Private Memory**
  - Per work-item
- **Local Memory**
  - Shared within a workgroup
- **Local Global/Constant Memory**
  - Visible to all workgroups
- **Host Memory**
  - On the CPU

- Memory management is explicit
  - You must move data from host -> global -> local *and* back
OpenCL C Language

- Derived from ISO C99
  - No standard C99 headers, function pointers, recursion, variable length arrays, and bit fields
- Additions to the language for parallelism
  - Work-items and workgroups
  - Vector types
  - Synchronization
- Address space qualifiers
- Optimized image access
- Built-in functions
OpenCL 1.1 - API

• Thread-safety
  • All API calls, except `clSetKernelArg`, are thread safe
• Sub-buffer objects
  • Create an object that represents a specific region in a buffer object
  • Easy and efficient mechanism to distribute regions of a buffer object across multiple devices
• OpenCL™ synchronization mechanism ensures modifications to sub-buffer object reflected in appropriate region of parent buffer object
OpenCL 1.1 - API

• User Events
  • **clEnqueue** *** commands can wait on event
  • In OpenCL™ 1.0, events can only refer to OpenCL™ commands
  • Need ability to enqueue commands that wait on an external, user defined, event

• Event CallBacks
  • **clSetEventCallbackFn** to register a user callback function
  • called when command identified by event has completed
  • Allows applications to enqueue new OpenCL™ commands based on event state changes in a non-blocking manner
**OpenCL 1.1 - Language**

- **Implicit Conversions**
  - OpenCL™ 1.0 requires widening for arithmetic operators
    ```
    float4 a, b;
    float c;
    b = a + c; // c is widened to a float4 vector
    // first and then the add is performed
    ```
  - OpenCL™ 1.1 extends this feature for all operators
    - relational, equality, bitwise, logical, ternary
OpenCL 1.1 - Language

• 3-component vector data types
  • And everyone applauds....well almost everyone

• cl_khr_byte_addressable as core feature

• Atomic extensions are now core features
  • cl_khr_global_int32_{base | extended}_atomics
  • cl_khr_local_int32_{base | extended}_atomics
OpenCL 1.1 - Language

• New built-in functions
  • `get_global_offset`
  • `clamp` for integer data types
  • `async_work_group_strided_copy`
  • strided async copy of data from global <--- local memory
  • `shuffle` - construct a permutation of elements from 1 or 2 input vectors and a mask
OpenCL 1.1 – OpenCL/OpenGL Sharing

• Improve performance of OpenCL/OpenGL interoperability
  • Portable OpenCL / OpenGL sharing requires
    • a glFinish before clEnqueueAcquireGLObjects
    • a clFinish after clEnqueueReleaseGLObjects
  • glFinish / clFinish are heavyweight APIs
OpenCL 1.1 – OpenCL/OpenGL Sharing

• Improve performance of OpenCL/OpenGL interoperability
  • Create a OpenCL event from an OpenGL sync object
  • Create a OpenGL sync object from a OpenCL event
  • Allows for a finer grained waiting mechanism
    • Use `event_wait_list` argument for events that refer to OpenGL commands to complete
    • Use OpenGL sync APIs to wait for specific OpenCL™ commands to complete