OpenCL 1.1 Enhancements for Multi-GPU Performance

Eric Young, NVIDIA

Overview

- NVIDIA Driver Availability
- Driver Performance Improvements
- OpenCL 1.1 Improvements
- OpenCL Multi-threading Example
- Summary

NVIDIA Driver Availability

- **NVIDIA R280 drivers now support OpenCL 1.1!**
- **Available on Desktop and Mobile**

NVIDIA Driver Version	Operating System	Available on NVIDIA.COM
280.26 Release WHQL	Vista and Windows 7	08/09/2011
280.13 Release	Linux	08/01/2011

NVIDIA Performance Improvements

- **Driver Optimizations:**
  - OpenCL driver adds multi-threaded optimizations.
  - Non-blocking API calls are enqueued and processed via worker threads.
  - Support for cross multi-GPU device synchronization.
    - CPU threads will not block during synchronization.
    - Less driver overhead with this type of synchronization.

- **Data I/O Improvements**
  - For data transfers between CPU and GPU with pageable memory buffers are asynchronous.

OpenCL 1.1 Improvements

- **Commands can be sent from different host threads**
  - APIs are thread safe except for `clSetKernelArg( … )`

- **Support for event callback functions on Host**
  - Enables heterogeneous computing (CPU + GPU) to be more efficient for both serial and parallel algorithms.
  - No explicit CPU/GPU synchronization required.

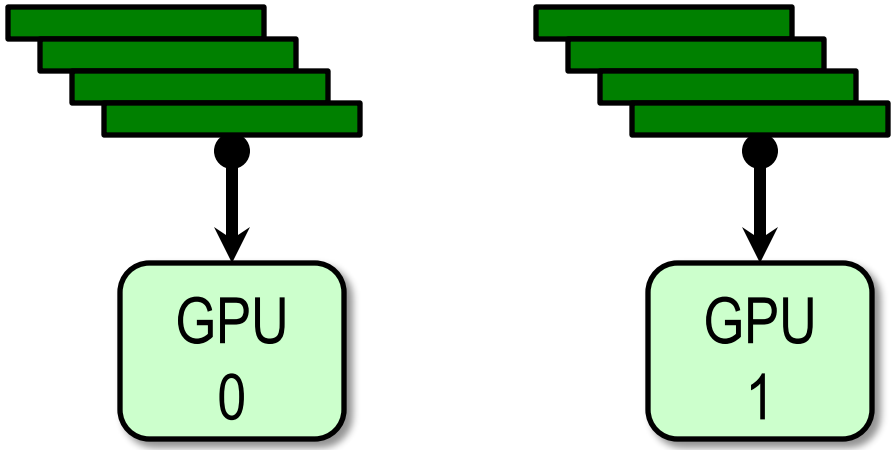
Callbacks and Events

- **OpenCL 1.0 has support events**
  - `clEnqueueBarrier()`, `clEnqueueMarker()`, `clEnqueueWaitForEvent()` for command-queue synchronization.
  - `clWaitForEvents()` for waiting on a event.

- **OpenCL 1.1 adds callback functions triggered by an event**
  - `clSetEventCallback()` passes a callback function that will be triggered by a user event. The callback function will get launched it a separate thread.
  - Use callback functions to distribute additional compute workloads on the GPU or CPU.

OpenCL Multi-Thread Example

- CPU thread initializes an array to be copied each GPU.
  - When each thread finishes, events (CPU Done Event) are set.
- 8 host threads run on 2 GPU devices.

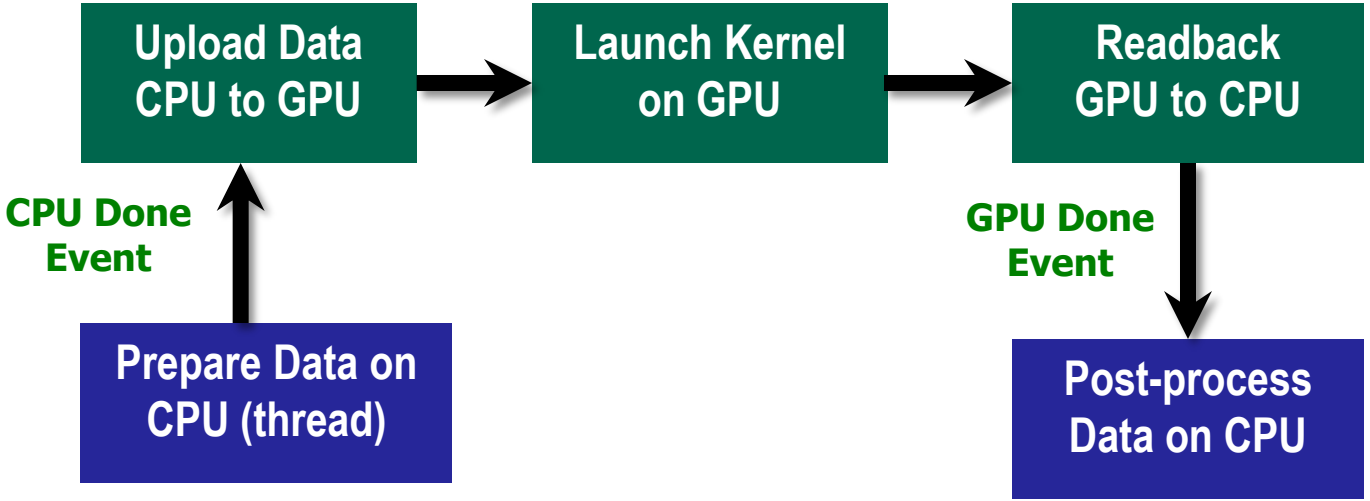


OpenCL Multi-Thread Example

- CPU is not blocked during enqueuing of commands. Each host thread has an OpenCL command queue and event.
- Triggering an event trigger will launch the callback function in a separate thread.

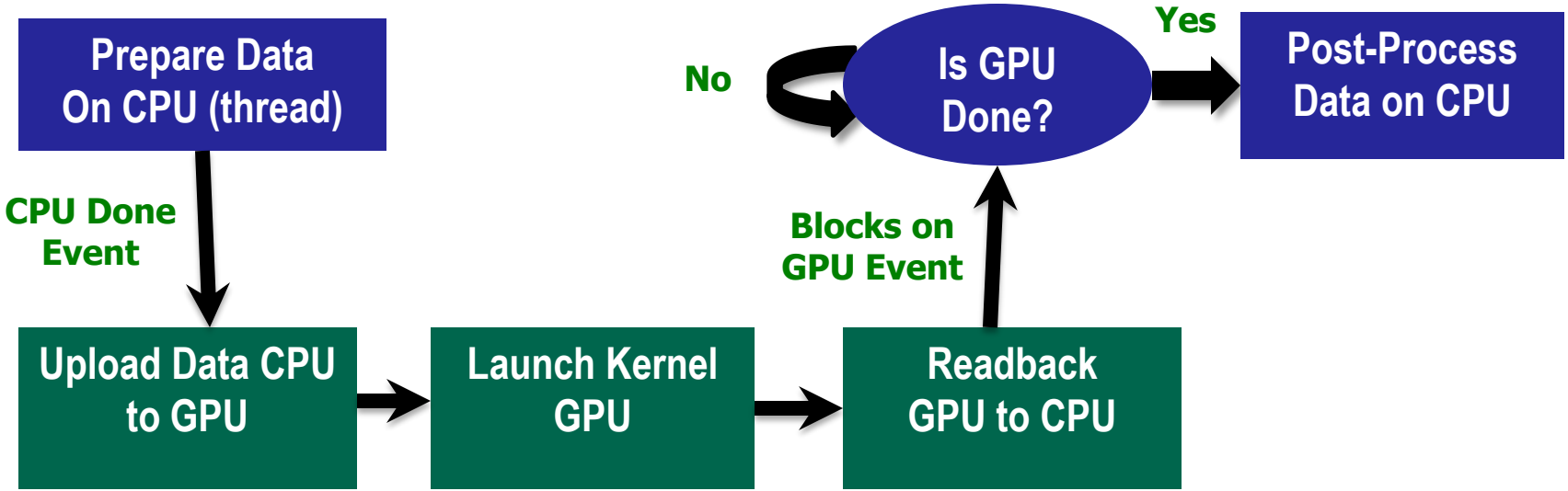
With OpenCL 1.1

- Data upload `clEnqueueWriteBuffer(…)` waits for CPU Done Event.
- `clSetKernelCallback(…)` - CPU post-process function gets triggered by the GPU Done Event.



With OpenCL 1.0

- `clWaitForEvents` (GPU Done Event) before GPU->CPU readback and launching CPU Post-Process.
- Explicit CPU/GPU synchronization required.



oclMultiThreads

- **Example of oclMultiThreads with multiple GPUs using OpenCL 1.1.**

```
[oclMultiThreads.exe] starting...
Detected 2 OpenCL devices of type CL_DEVICE_TYPE_GPU
> OpenCL Device #0 (Tesla C2050), cl_device_id: 5445872
> OpenCL Device #1 (Quadro 4000), cl_device_id: 5445928
Tesla C2050: simpleIncrement(), cl_device_id: 44505176, event_id: 0
Quadro 4000: simpleIncrement(), cl_device_id: 44535184, event_id: 1
Tesla C2050: simpleIncrement(), cl_device_id: 44505176, event_id: 2
Quadro 4000: simpleIncrement(), cl_device_id: 44535184, event_id: 3
Tesla C2050: simpleIncrement(), cl_device_id: 44505176, event_id: 4
Quadro 4000: simpleIncrement(), cl_device_id: 44535184, event_id: 5
Tesla C2050: simpleIncrement(), cl_device_id: 44505176, event_id: 6
Quadro 4000: simpleIncrement(), cl_device_id: 44535184, event_id: 7
```

oclMultiThreads (contd.)

Timing results from each thread and event

> event_callback() event_id=0, kernel runtime 49.389344 (ms)
> event_callback() event_id=1, kernel runtime 82.967264 (ms)
> event_callback() event_id=2, kernel runtime 91.719552 (ms)
> event_callback() event_id=4, kernel runtime 43.965088 (ms)
> event_callback() event_id=3, kernel runtime 26.245088 (ms)
> event_callback() event_id=5, kernel runtime 65.455936 (ms)
> event_callback() event_id=6, kernel runtime 59.854240 (ms)
> event_callback() event_id=7, kernel runtime 30.516416 (ms)

Summary

- **NVIDIA OpenCL driver is multi-threaded:**
  - Enqueuing commands and Data I/O are asynchronous to the application thread (non-blocking).
    - Improves overlap of the CPU and GPU workloads.
- **OpenCL 1.1 adds**
  - Event triggered callback functions:
  - Callback functions are non-blocking and run in a separate thread.
- **Take advantage of these features to achieve:**
  - Higher Compute throughput (CPU + GPU efficiency).
  - Higher GPU utilization (less time idle).
  - Lower API overhead with single and multiple GPUs.

Thank You!

Contact:

Eric Young

eyoung@nvidia.com

OpenCL Programming Guide

By Aaftab Munshi, Benedict R Gaster, Timothy G. Mattson, James Fung, Dan Ginsburg

Key Features (includes)

- Understanding OpenCL's core models, concepts, terminology, goals, and rationale
- Discovering and preparing available resources
- Programming with OpenCL C, its built-in functions, and runtime API
- Using buffers, sub-buffers, images, samplers, and events
- Sharing and synchronizing data with OpenGL and Microsoft's Direct3D
- Simplifying development with the C++ Wrapper API
- Using OpenCL Embedded Profiles to support devices ranging from cellphones to supercomputer nodes
- Building complete applications: image histograms, edge detection filters, physics simulations, Fast Fourier Transforms, optical flow, and more
- Using OpenCL with PyOpenCL, including issues in porting from C++ to Python
- Performing matrix multiplication and high-performance sparse matrix multiplication