OPENCL AT NVIDIA – BEST PRACTICES, LEARNINGS AND PLANS

Nikhil Joshi and Karthik Raghavan Ravi, May 8, 2017

AGENDA

Insights

- Memory management
- MultiGPU/multi-queue use cases

Best Practices

- Efficient memory management
- Efficient data transfers

Plans/Updates

- New development
- Performance improvements

INSIGHTS

ALLOCATING MEMORY

What the OpenCL spec says (and does not)

CL_MEM_ALLOC_HOST_PTR

> “This flag specifies that the application wants the OpenCL implementation to allocate memory from host accessible memory”

Spec does not specify

- Type of host memory (pinned vs pageable)
- Device accessibility (by default assumed)
- Memory placement (host vs device)

ALLOCATING MEMORY

Data transfer performance characteristics

Perf characteristics change based on

- Type of host memory (Pinned vs pageable)
- Size of the buffer
- Choice of API (Read/WriteBuffer vs Map/Unmap)

MEMORY MANAGEMENT

NVIDIA Interpretation and Heuristics

GPU friendly

- Memory placement close to GPU

Lazy

- Deferred until memory are actually needed on device

Host memory pinning

- Treats pinned memory as scarce resource
- Not always pinned

FINER CONTROL OVER MEMORY MGMT

New extension (tentatively “cl_nv_create_buffer”)

Current heuristic optimal for common GPU-bound use cases, but not all use cases

For example:

- Fully async copies between host and device
- Sparse access from kernel

New extension under preview that provides greater control over memory to better optimize for each use case. Production expected 3Q17.

MULTIGPU/MULTIQUEUE USE CASES

Current driver tuned for

- single gpu, single command-queue use cases
- max perf, optimal latency

Not optimized for

- MultiGPU/multi-command queue use-cases
- Pageable memcpy (naïve copies, very large buffers)

COMING UP

Currently rearchitecting parts of the driver to improve performance on these scenarios, expected 3Q17

Fast paths will continue to remain the same

BEST PRACTICES

BEST PRACTICES

cl_nv_create_buffer

Gives all functionality of clCreateBuffer. In addition, provides knobs for memory placement [trading off access latency vs data migration cost]

- close to GPU
  - fast access from GPU
  - ideal for heavy access from kernel
- close to CPU
  - saves GPU memory and data migration cost
  - ideal for sparse access from kernel

BEST PRACTICES

cl_nv_create_buffer

host allocation [trading off speed vs availability]

- Pinned memory
  - fast, async copies between GPU
  - this is a scarce system resource
- Pageable memory
  - easily available
  - Not as fast as pinned copies

BEST PRACTICES

cl_mem_flags & cl_map_flags

Choosing cl_mem_flags and cl_map_flags appropriately can save unnecessary data movement and improve performance

CL_MEM_READ_ONLY

CL_MEM_WRITE_ONLY

CL_MAP_WRITE_INVALIDATE_REGION

BEST PRACTICES

Read/WriteBuffer vs Map/UnmapBuffer

Prefer Map/Unmap over Read/Write

- Map/Unmap internally uses pinned memory
- Pinned memcpy bandwidth near SOL

Read/WriteBuffer

- Perf depends on nature of host memory
- Pinned memory perf comparable to Map/Unmap
- Pageable memory bandwidth 30%-50% of pinned memcpy bandwidth
  - *Upcoming improvements will bridge some of the gap to pinned copy performance

PLANS & UPDATES

UPDATES

New development

- OpenCL 1.2+ preview support
- Upcoming cl_nv_create_buffer extension

Perf improvements

- Improvements for multiGPU, multiple command queues use-cases
- Better pageable memcpy

OPENCL 1.2+

OpenCL 2.0 features preview, available 378+

- Device-Side-Enqueues
- Shared Virtual Memory - Coarse Grained Buffer
- Generic Address Spaces
- 3D writes

NOTE

- Not OpenCL 2.0 conformant
- 1.2+ features are experimental, not intended to be used in production

QUESTIONS?

PREVIOUS TALKS

Focused on Applications

“Using OpenCL for Performance-Portable, Hardware-Agnostic, Cross-Platform Video Processing”

by Dennis Adams (Sony Creative Software Inc.)

“Boosting Image Processing Performance in Adobe Photoshop with GPGPU Technology”
by Joseph Hsieh (Adobe)

PREVIOUS TALKS

Focused on Kernel performance

“Better Than All the Rest: Finding Max-Performance GPU Kernels Using Auto-Tuning”

by Cedric Nugteren (SURFsara HPC centre)

“Auto-Tuning OpenCL Matrix-Multiplication: K40 versus K80”

by Cedric Nugteren (SURFsara)

PREVIOUS TALKS

Driver/Runtime performance

“Performance Considerations for OpenCL on NVIDIA GPUs”

by Karthik Raghavan Ravi (NVIDIA)

MORE OPPORTUNITIES TO DISCUSS

H7109: Creating Efficient OpenCL Software [Connect With The Experts sessions]

- 5/8, 1PM-2PM, Lower Level Pod B
- 5/9, 4PM-5PM, Lower Level Pod C