GpuWrapper: A Portable API for Heterogeneous Programming at CGG

Victor Arslan, Jean-Yves Blanc, Gina Sitaraman, Marc Tchiboukdjian, Guillaume Thomas-Collignon

Passion for Geoscience

March 2nd, 2016
GpuWrapper: Objectives & Design

- **GpuWrapper**: CGG compatibility layer to support both CUDA & OpenCL
  - Increase applications portability
  - Future proof accelerated applications development

- **Objectives**
  - Reach maximum performance on Nvidia & AMD GPUs
  - Consider the complete platform not only the hardware: Compiler, Runtime, Debugger, Profiler, Numerical libraries

- **Accelerated application = unified host code + device-specific kernels**
  - One version of the host code for all devices (majority of application code is the host part)
  - One version of a kernel for each device (enable device-specific tuning)

<table>
<thead>
<tr>
<th>GPU Architecture</th>
<th>Nvidia Kepler</th>
<th>Nvidia Maxwell</th>
<th>AMD Hawaii</th>
<th>AMD Fiji</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory BW</td>
<td>336 GB/s</td>
<td>336 GB/s</td>
<td>320 GB/s</td>
<td>512 GB/s</td>
</tr>
<tr>
<td>Compute (SP)</td>
<td>5 Tflops</td>
<td>6 Tflops</td>
<td>5 Tflops</td>
<td>8 Tflops</td>
</tr>
</tbody>
</table>
GpuWrapper API

- Initialization & cleanup of devices
- Allocation and memory transfers
- Kernel launch
- Synchronizations with streams and events
- Device-to-device communications
- FFT API

GpuWrapper is targeted at CGG applications, hence it is simpler

- 50 API calls
- 3k LOC each for OpenCL & CUDA implementations
OpenCL vs CUDA

- OpenCL and CUDA are quite similar but there are some differences

<table>
<thead>
<tr>
<th></th>
<th>CUDA</th>
<th>OpenCL</th>
<th>GpuWrapper</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU memory buffers</td>
<td>Pinned to a GPU</td>
<td>Automatically moved to device before use</td>
<td>Similar to CUDA</td>
</tr>
<tr>
<td>Kernel compilation</td>
<td>Offline</td>
<td>JIT</td>
<td>Similar to CUDA</td>
</tr>
<tr>
<td>Kernel launch</td>
<td>&lt;&lt;&lt;…&gt;&gt;&gt; syntax with compiler support</td>
<td>clSetKernelArg</td>
<td>Similar to OpenCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clEnqueueNDRange</td>
<td></td>
</tr>
<tr>
<td>Pointer arithmetic on host</td>
<td>Supported</td>
<td>Use subbuffers</td>
<td>Similar to OpenCL</td>
</tr>
<tr>
<td>Device initialization</td>
<td>Implicit</td>
<td>Explicit</td>
<td>Similar to CUDA</td>
</tr>
<tr>
<td>FFT library</td>
<td>cuFFT</td>
<td>clFFT</td>
<td>Common API to use both</td>
</tr>
</tbody>
</table>
Nvidia GPUs vs AMD GPUs

- Nvidia and AMD GPU architectures are similar but …
- Differences sometimes require different kernel implementations
  - To fit within specifications
  - To get good performance (especially differences impacting occupancy)

<table>
<thead>
<tr>
<th>Short list of characteristics</th>
<th>Nvidia (CC 3.0)</th>
<th>Nvidia (CC 3.5)</th>
<th>AMD Hawaii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max threads per block</td>
<td>1024</td>
<td>1024</td>
<td>256</td>
</tr>
<tr>
<td>Max shared memory size per block</td>
<td>48KB</td>
<td>48KB</td>
<td>32KB</td>
</tr>
<tr>
<td>Max registers per thread</td>
<td>63</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>Warp size</td>
<td>32</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Cache line size</td>
<td>128B</td>
<td>128B</td>
<td>64B</td>
</tr>
</tbody>
</table>
Low latency GPU to GPU communications

- **GpuWrapper API semantic**
  - Common memory buffer accessible by kernels
  - For a pair of GPUs

- **GpuWrapper implementations**
  1. Memory buffer allocated in pinned host memory
  2. Memory buffer allocated in one of the GPUs (GPUDirect/DirectGMA)

- **Policy to select implementation**
  - Use #2 when GPUs are on the same socket
  - Otherwise use #1
  - Specifically for AMD:
    - Use #1 for small buffers (better performance)
    - or when both remote read & remote write are required (remote read not supported)

This functionality is no longer duplicated for each application
Porting applications from CUDA to the GpuWrapper API

- 2 steps process to port applications
  1. Port CUDA API host code to use the GpuWrapper API (keep kernels in CUDA)
    Check results on NVIDIA GPU
  2. Implement OpenCL version of the CUDA kernels (host code stays the same)
    Check results on AMD GPU

- Factors impacting results
  - Different compilers
  - Compilation flags
  - Different FFT libraries

<table>
<thead>
<tr>
<th>OpenCL</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag</td>
<td>Default</td>
</tr>
<tr>
<td>-cl-mad-enable</td>
<td>False</td>
</tr>
<tr>
<td>-cl-fp32-correctly-rounded-divide-sqrt</td>
<td>False</td>
</tr>
<tr>
<td>-cl-denorms-are-zero</td>
<td>False</td>
</tr>
</tbody>
</table>
Evaluation of the GpuWrapper Overhead

Does the GpuWrapper add overhead over the CUDA implementation on the same hardware?

- Experimental Setup
  - Gather representative set of production jobs for each application
  - Run each job on each generation of Nvidia GPU accelerated nodes
  - Compare performance of the pure CUDA implementation and the GpuWrapper implementation

![Negligible Overhead](image.png)
Performance Comparison on Wave-Equation Modeling

Performance scales with memory BW

Increasing wave-equation complexity
Related Work: Portable Heterogeneous Programming

- **Directive-based**
  - OpenMP4, OpenACC
  - Higher level

- **Source to source translation**
  - CU2CL, Ocelot, Kim et al 2015
  - Great for initial port but
  - Some features are not supported
  - Not viable over the lifetime of the application

- **Compatibility layer**
  - Swan, OCCA, Souza et al 2015, AMD HIP
  - OCCA and HIP also provide a common kernel language
  - Very similar to our approach but missing specific features
Conclusion & Acknowledgments

- **GpuWrapper API**
  - Multiple applications successfully deployed in production
    - Compatible with both CUDA and OpenCL
    - Fully transparent for geophysicists
    - No overhead on Nvidia GPUs
  - Over 1 SP Petaflops of AMD GPUs successfully deployed in production
    - With two system vendors (Supermicro, Dell)
    - Great performance on wave-equation modeling
  - Wrapper for future manycores?

- **Acknowledgements**
  - Great support from AMD (many thanks to Benjamin Coquelle)
  - Great system support from Dell