Cross-Platform GPGPU with OpenCL

George van Venrooij
Organic Vectory B.V.
george.van.venrooij@organicvectory.com

OpenCL

- As defined by the Khronos Group (www.khronos.org):
  - OpenCL™ is the first open, royalty-free standard for cross-platform, parallel programming of modern processors found in personal computers, servers and handheld/embedded devices. OpenCL (Open Computing Language) greatly improves speed and responsiveness for a wide spectrum of applications in numerous market categories from gaming and entertainment to scientific and medical software.
  - Khronos Group controls many other standards like:
    - OpenGL (ES)
    - OpenVG
    - COLLADA
    - WebGL
    - … and many more

OpenCL vs. CUDA

OpenCL & CUDA

Device Memory Model

(source: nVidia OpenCL Tutorial Slides)
Preliminary Conclusions

- There are cases where OpenCL performs better than CUDA.
- There are cases where CUDA performs better than OpenCL.
- OpenCL seems to have slightly higher overhead for kernel launches compared to CUDA on NVIDIA’s platform.
- For some cases the differences can be large, but...

Measuring = knowing!
Host Synchronization: CUDA Streams

- Streams are a sequence of commands that execute in-order.
- Streams can contain kernel launches and/or memory transfers.
- Host code can wait for stream completion using the `cudaStreamSynchronize()` call.
- Events can be inserted into the stream.
- Host code can query event completion or perform a blocking wait for an event.
- Useful for synchronization with host code and timing.

Host Synchronization: OpenCL Command Queues

- Default behavior of command queue's is similar to CUDA Streams.
- One big difference: out-of-order execution mode.
- `clEnqueue...()` commands can be given a set of events to wait for.
- Each command itself can generate an event.
- Based on the dependencies between commands in the queue, OpenCL can determine which commands are allowed to execute simultaneously.
- It is possible to create multiple queues for a device.
- It is possible have commands in one queue wait for events from a different queue.

Task & Data Parallelism

- The commands and the events they must wait for, create a “task graph”.
- OpenCL will execute the commands in the queue as it sees fit, respecting the dependencies specified.
- The end-result is a task-parallel framework supporting data-parallel tasks.
- Your application could be written entirely in OpenCL kernels, requiring only a small framework that fills the command queue.

Intermediate Conclusions

- The programming methodology for data-parallel application is virtually identical, i.e. if you can program in one language/environment, you can program in the other.
- CUDA currently offers certain productivity advantages at the kernel level.
- NVidia’s hardware seems to be more capable on the GPGPU side when compared to ATi’s hardware.
- OpenCL has the platform advantage in that it presents a unified platform API for ALL computing hardware in your machine.
- OpenCL programs can be run on hardware from different vendors.

OpenCL Implementations

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Type</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>CPU</td>
<td>x86_64 (Intel)</td>
</tr>
<tr>
<td>nVidia</td>
<td>GPU</td>
<td>GeForce 8/9 series and higher</td>
</tr>
<tr>
<td>ATI</td>
<td>GPU</td>
<td>R700/800 series</td>
</tr>
<tr>
<td>AMD</td>
<td>CPU</td>
<td>any x86/x86_64 with SSE3 extensions</td>
</tr>
<tr>
<td>Samsung</td>
<td>CPU</td>
<td>ARM A9</td>
</tr>
<tr>
<td>IBM</td>
<td>ACCELERATOR</td>
<td>CELL BE</td>
</tr>
<tr>
<td>ZiiLabs</td>
<td>CPU</td>
<td>ARM</td>
</tr>
</tbody>
</table>

ANNOUNCED/UPCOMING:

- Imagination Technology: GPU PowerVR SGX Series 5
- VIA: GPU VN 1000 Chipset
- S3: GPU Chrome 5400E Graphics Processor
- Apple: CPU ARM A4

Portability to other platforms

- Results of a kernel are guaranteed across platforms.
- Optimal Performance is not.
- All platforms are required to support data-parallelism, but are not required to support task-parallelism.
- OpenCL can be considered a replacement for OpenMP (data-parallel).
- OpenCL can be considered a replacement for Threads (task-parallel).
Libraries & Tools for OpenCL

- ATI StreamProfiler (ATI hardware only)
- NVidia Visual Profiler (NVidia hardware only)
- Stream KernalAnalyzer (ATI hardware only)
- NVidia NSight (NVidia hardware only)
- gDebugger CL (Windows, Mac, Linux, currently in beta)
- libstdcl (wrapper around context/queue management functions)
- GATLAS (Matrix multiplication)
- ViennaCL (BLAS level 1 and 2)
- Language bindings for C++, Fortran, Java, Matlab, .NET, Python and Scala are available

Libraries & Tools for CUDA

- cuBLAS (closed-source)
- cuFFT (closed-source)
- CUDPP (data-parallel primitives)
- Thrust (high-level CUDA & OpenMP-based algorithms)
- CULATools (LAPACK)
- NSight debugger
- NVidia Visual Profiler
- Language bindings for Python, Java, .NET, MATLAB, Fortran, Perl, Ruby, Lua

Things to consider

- Platforms
  - OpenCL is currently the only choice if you do not want to tie your application to NVidia’s hardware
- API stability/agility
  - OpenCL changes more slowly, retains backward compatibility
  - CUDA changes more rapidly, unlocks new hardware features quicker
- Third-party library availability
  - OpenCL is about 2 years younger, so less numerous and less mature libraries are available
  - CUDA has spawned a host of initiatives and various libraries are available, especially in the scientific computing domain
- Supporting tools
  - OpenCL has a fairly young, but decent set of tools
  - NVidia recently launched the NSight debugger which seems more mature

Questions

Further Reading

- GPGPU
  - www.gpgpu.org
- OpenCL General
  - www.khronos.org/opengl
- OpenCL Implementations
- OpenCL/CUDA Comparisons
- Mobile/Embedded OpenCL announcements
References

- http://blog.accelereyes.com/blog/2010/05/10/nvidia-fermi-cuda-and-openc/
- http://www.gremedy.com/gDEBuggerCL.php
- http://browneferotechnology.com/stdcl.html
- http://golem5.org/gattas/