Languages, APIs and Development Tools for GPU Computing

Will Ramey
Product Manager, GPU Computing
GPU Computing Overview

**Broad Adoption**

- Over 150,000,000 installed CUDA-Architecture GPUs
- Over 90,000 GPU Computing Developers (9/09)
- Windows, Linux and MacOS Platforms supported
- GPU Computing spans HPC to Consumer
- 250+ Universities teaching GPU Computing on the CUDA Architecture

**GPU Computing Applications**

<table>
<thead>
<tr>
<th>CUDA C/C++</th>
<th>OpenCL</th>
<th>Direct Compute</th>
<th>Fortran</th>
<th>Python, Java, .NET, …</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Over 90,000 developers&lt;br&gt;- Running in Production since 2008&lt;br&gt;- SDK + Libs + Visual Profiler and Debugger</td>
<td>- 1st GPU demo&lt;br&gt;- Shipped 1st OpenCL Conformant Driver&lt;br&gt;- Public Availability</td>
<td>- Microsoft API for GPU Computing&lt;br&gt;- Supports all CUDA-Architecture GPUs (DX10 and DX11)</td>
<td>- PGI Accelerator&lt;br&gt;- PGI CUDA Fortran&lt;br&gt;- NOAA Fortran bindings&lt;br&gt;- FLAGON</td>
<td>- PyCUDA&lt;br&gt;- jCUDA&lt;br&gt;- CUDA.NET&lt;br&gt;- OpenCL.NET</td>
</tr>
</tbody>
</table>

**NVIDIA GPU**

with the CUDA Parallel Computing Architecture

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.
## GPU Computing Application Development

<table>
<thead>
<tr>
<th>Your GPU Computing Application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Acceleration Engines (AXEs)</strong></td>
</tr>
<tr>
<td>Middleware, Modules &amp; Plug-ins</td>
</tr>
<tr>
<td><strong>Foundation Libraries</strong></td>
</tr>
<tr>
<td>Low-level Functional Libraries</td>
</tr>
<tr>
<td><strong>Development Environment</strong></td>
</tr>
<tr>
<td>Languages, Device APIs, Compilers, Debuggers, Profilers, etc.</td>
</tr>
</tbody>
</table>

**CUDA Architecture**
Languages & APIs
CUDA C/C++ Update

- CUDA Toolkit 1.0
  - C Compiler
  - C Extensions
  - Single Precision
  - BLAS
  - FFT
  - SDK
  - 40 examples

- CUDA Toolkit 1.1
  - Win XP 64
  - Atomics support
  - Multi-GPU support

- CUDA Visual Profiler 2.2
  - cuda-gdb
  - HW Debugger

- CUDA Toolkit 2.0
  - Double Precision
  - Compiler Optimizations
  - Vista 32/64
  - Mac OSX
  - 3D Textures
  - HW Interpolation

- CUDA Toolkit 2.3
  - DP FFT
  - 16-32 Conversion intrinsics
  - Performance enhancements

- CUDA Toolkit 3.0 “Nexus” Beta
  - C++ Functionality
  - Fermi Arch support
  - Tools
  - Driver and RT

© 2009 NVIDIA Corporation
CUDA C/C++ Update

CUDA C++ Language Functionality
- C++ Class Inheritance
- C++ Template Inheritance
(Productivity)

CUDA Driver & CUDA C Runtime
- CUDA Driver / Runtime Buffer interoperability (Stateless CUDART)
- Separate emulation runtime
- Unified interoperability API for Direct3D and OpenGL
- OpenGL texture interoperability
- Fermi: Direct3D 11 interoperability

Fermi Architecture Support
- Native 64-bit GPU support
- Generic Address space
- Dual DMA support
- ECC reporting
- Concurrent Kernel Execution
Architecture supported in SW now

Tools
- "Nexus" Visual Studio IDE
- Debugger support for CUDA Driver API
- ELF support (cubin format deprecated)
- CUDA Memory Checker (cuda-gdb feature)
- Fermi: Fermi HW Debugger support in cuda-gdb
- Fermi: Fermi HW Profiler support in CUDA Visual Profiler

CUDA Toolkit 3.0 Beta
- C++ Functionality
- Fermi Arch support
- Tools
- Driver and RT

"Nexus" Beta

CUDA Toolkit 2.3
- 3D Textures
- HW Interpolation
- DP FFT
- 16-32 Conversion intrinsics
- Performance enhancements
- C Compiler
- C Extensions
- Single Precision
- BLAS
- FFT
- SDK
- 40 examples

CUDA Toolkit 2.0
- Win XP 64
- Atomics support
- Multi-GPU support
- Double Precision
- Compiler Optimizations
- Vista 32/64
- Mac OSX
- Performance enhancements

CUDA Toolkit 1.1
- Fermi Architecture Support
- Native 64-bit GPU support
- Generic Address space
- Dual DMA support
- ECC reporting
- Concurrent Kernel Execution
- Architecture supported in SW now
- C++ Class Inheritance
- C++ Template Inheritance
(Productivity)

CUDA Toolkit 1.0
- July 07
- Nov 07
- April 08
- Aug 08
- July 09
- Nov 09

© 2009 NVIDIA Corporation
Fortran Language Solutions

- **PGI Accelerators**
  - High-level *implicit* programming model, similar to OpenMP
  - Auto-parallelizing compiler

- **PGI CUDA Fortran Compiler**
  - High-level *explicit* programming model, similar to CUDA C Runtime

- **NOAA F2C-ACC**
  - Converts Fortran codes to CUDA C
  - Some hand-optimization expected

- **FLAGON**
  - Fortran 95 Library for GPU Numerics
  - Includes support for cuBLAS, cuFFT, CUDPP, etc.
OpenCL

- Cross-vendor open standard
  - Managed by the Khronos Group

- Low-level API for device management and launching kernels
  - Close-to-the-metal programming interface
  - JIT compilation of kernel programs

- C-based language for compute kernels
  - Kernels must be optimized for each processor architecture

NVIDIA released the first OpenCL v1.0 conformant driver for Windows and Linux to thousands of developers in June 2009.

http://www.khronos.org/opencl
NVIDIA OpenCL Support

- OpenCL ICD
- OpenGL Interoperability
- Double Precision
- NVIDIA Compiler Flags
- Query for Compute Capability
- Byte Addressable Stores
- 32-bit Atomics
- Images
- OpenCL 1.0 Driver
  Min. Spec. NV R190 released June 2009

NVIDIA OpenCL Documentation
NVIDIA OpenCL Visual Profiler
NVIDIA OpenCL Code Samples
DirectCompute

- Microsoft standard for all GPU vendors
  - Released with DirectX® 11 / Windows 7
  - Runs on all 150M+ CUDA-enabled DirectX 10 class GPUs and later

- Low-level API for device management and launching kernels
  - Good integration with other DirectX APIs

- Defines HLSL-based language for compute shaders
  - Kernels must be optimized for each processor architecture
## Language & APIs for GPU Computing

<table>
<thead>
<tr>
<th>Solution</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA C / C++</td>
<td>Language Integration</td>
</tr>
<tr>
<td></td>
<td>Device-Level API</td>
</tr>
<tr>
<td>PGI Accelerator</td>
<td>Auto Parallelizing Compiler</td>
</tr>
<tr>
<td>PGI CUDA Fortran</td>
<td>Language Integration</td>
</tr>
<tr>
<td>CAPS HMPP</td>
<td>Auto Parallelizing Compiler</td>
</tr>
<tr>
<td>OpenCL</td>
<td>Device-Level API</td>
</tr>
<tr>
<td>DirectCompute</td>
<td>Device-Level API</td>
</tr>
<tr>
<td>PyCUDA</td>
<td>API Bindings</td>
</tr>
<tr>
<td>CUDA.NET, OpenCL.NET, jCUDA</td>
<td>API Bindings</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>CUDA C/C++</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
</tr>
<tr>
<td>CAPS HMPP</td>
<td>✓</td>
</tr>
<tr>
<td>PGI Accelerators</td>
<td>✓</td>
</tr>
<tr>
<td>PGI CUDA Fortran</td>
<td>✓</td>
</tr>
<tr>
<td>NV cuda-gdb</td>
<td>✓</td>
</tr>
<tr>
<td>Allinea DDT</td>
<td>✓</td>
</tr>
<tr>
<td>TotalView Debugger</td>
<td>✓</td>
</tr>
<tr>
<td>NV Visual Profiler</td>
<td>✓</td>
</tr>
<tr>
<td>TAU CUDA</td>
<td>✓</td>
</tr>
<tr>
<td>Mcore Platform</td>
<td>✓</td>
</tr>
<tr>
<td>Analyzer</td>
<td></td>
</tr>
</tbody>
</table>

cuda-gdb will be extended to support OpenCL debugging in a future release, with solutions from Allinea, TotalView and others expected to follow.
NVIDIA SDKs

Hundreds of code samples for CUDA C/C++, DirectCompute, and OpenCL

- Finance
- Oil & Gas
- Video/Image Processing
- 3D Volume Rendering
- Particle Simulations
- Fluid Simulations
- Math Functions
## Massively Parallel Development

**Tools for Linux**

<table>
<thead>
<tr>
<th></th>
<th>Debugging</th>
<th>Profiling</th>
<th>Analysis</th>
<th>Cluster Support</th>
<th>Lib’s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CUDA C/C++</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>OpenCL</strong></td>
<td>Coming Soon</td>
<td>✓</td>
<td>Coming Soon</td>
<td>✓</td>
<td>Coming Soon</td>
</tr>
<tr>
<td><strong>Fortran</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

cuda-gdb will be extended to support OpenCL debugging in a future release, with solutions from Allinea, TotalView and others expected to follow.
CUDA debugging integrated into GDB on Linux

- Supported on 32bit and 64bit systems
- Seamlessly debug both the host/CPU and device/GPU code
- Set breakpoints on any source line or symbol name
- Access and print all CUDA memory allocs, local, global, constant and shared vars

Included in the CUDA Toolkit
CUDA Visual Profiler

- Analyze GPU HW performance signals, kernel occupancy, instruction throughput, and more
- Highly configurable tables and graphical views
- Save/load profiler sessions or export to CSV for later analysis
- Compare results visually across multiple sessions to see improvements
- Windows, Linux and Mac OS X supported

Included in the CUDA Toolkit
“Nexus” 1.0 Beta

Parallel Debugger
GPU source code debugging
Variable & memory inspection

System Analyzer
Platform-level Analysis
For the CPU and GPU
Visualize Compute Kernels, Driver API Calls, and Memory Transfers

Graphics Inspector
Visualize and debug graphics content
# Massively Parallel Development Tools for Windows

<table>
<thead>
<tr>
<th></th>
<th>Visual Studio Integration</th>
<th>Parallel Debugging</th>
<th>Parallel Profiling</th>
<th>System Analysis/Trace (CPU/GPU)</th>
<th>Premium Support</th>
<th>Early Access</th>
<th>Multi-Vendor GPU Support</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compute</td>
<td>Gfx</td>
<td>Compute</td>
<td>Gfx</td>
<td>Compute</td>
<td>Gfx</td>
<td></td>
</tr>
<tr>
<td>&quot;Nexus** Pro</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>&quot;Nexus**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NV Visual Profiler</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mcore Platform Analyzer</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PerfKit/PerfHUD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gDEBugger</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

* "Nexus" is NVIDIA’s code name
Massively Parallel Development  
Tools for Linux

<table>
<thead>
<tr>
<th></th>
<th>Compilation</th>
<th>Debugging</th>
<th>Profiling</th>
<th>Analysis</th>
<th>Premium Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPS HMPP</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PGI Accelerators</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PGI CUDA Fortran</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>NV cuda-gdb</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allinea DDT</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>TotalView Debugger</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>NV Visual Profiler</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAU CUDA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mcore Platform Analyzer</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

cuda-gdb will be extended to support OpenCL debugging in a future release, with solutions from Allinea, TotalView and others expected to follow.
Thank You
Python + CUDA = PyCUDA

- All of CUDA in a modern scripting language
- Full Documentation
- Free, open source (MIT)
- Also: PyOpenCL

- CUDA C Code = Strings
- Generate Code Easily
  - Automated Tuning
- Batteries included: GPU Arrays, RNG, ...
- Integration: numpy arrays, Plotting, Optimization, ...

Slide courtesy of Andreas Klöckner, Brown University
HMPP Workbench Overview

- C and Fortran GPU programming directives
  - Define and execute GPU-accelerated versions of functions
  - Implement efficient communication patterns
  - Build parallel hybrid applications with OpenMP and MPI

- An open hybrid compiling workbench
  - Automatically generate CUDA computations
  - Use standard compilers and hardware vendor tools
  - Drive the whole compilation workflow

- A runtime library
  - Dispatch computations on available GPUs
  - Scale to multi-GPUs systems

http://www.caps-entreprise.com/hmpp.html
Using jCUDA you can create cross-platform CUDA solutions, that can run on any operating system supported by CUDA without changing your code. Either select between Windows XP or Vista by Microsoft or even Linux/MacOS/Solaris systems. Current support is for both 32 and 64 bits of every platform.

**Features**

- Double precision
- Object model for CUDA programming
- CUDA 2.1 Driver API
- CUDA 2.1 Runtime API
- CUFFT routines
- OpenGL interoperability
- * Support for CUBLAS routines will be added in the future

**Operating System Support**

- Microsoft Windows
- Linux
- * Support for Mac OSX will be added in the future
- * Support for Solaris 10 (x86) will be added in the future

CUDA.NET

2.3.6, 14/9/2009
Updates to native wrappers, added SizeT structure to handle 32/64 systems compatibility for functions taking size_t as parameter.
Support for CUDA 2.3 through .NET bindings to CUDA functions.
Currently supported on Windows, Linux and MacOS platforms.

Features

- Double precision
- Object model for CUDA programming
- CUDA 2.2 Driver API
- CUDA 2.2 Runtime API
- CUFFT routines
- CUBLAS routines
- Direct3D interoperability
- OpenGL interoperability

Operating System Support

- Microsoft Windows
- Linux 32/64 bit (using Mono)
- Mac OSX (using Mono)

OPENCL.NET

2.3.6, 14/9/2009
Updates to native wrappers, added SizeT structure to handle 32/64 systems compatibility for functions taking size_t as parameter.
Support for CUDA 2.3 through .NET bindings to CUDA functions.
Currently supported on Windows, Linux and MacOS platforms.

Features

- Double precision
- Object model for CUDA programming
- CUDA 2.2 Driver API
- CUDA 2.2 Runtime API
- CUFFT routines
- CUBLAS routines
- Direct3D interoperability
- OpenGL interoperability

Operating System Support

- Microsoft Windows
- Linux 32/64 bit (using Mono)
- Mac OSX (using Mono)