Languages, APIs and Development Tools for GPU Computing

Phillip Miller
Director, Software Product Management
Professional Solutions Group
CUDA Architecture
Available on All Modern NVIDIA GPUs

GeForce®
Entertainment

Tesla™
High Performance Computing

Quadro®
Design & Creation

Over 150 Million CUDA-enabled GPUs Installed
GPUs are Accelerating Time to Discovery

- **Computational Chemistry (UIUC)**: 4.6 Days vs. 27 Minutes
- **Neurological Modeling (Evolved Machines)**: 2.7 Days vs. 30 Minutes
- **Cell Phone RF Simulation (Nokia, Motorola)**: 8 Hours vs. 13 Minutes
- **3D CT Ultrasound (Techniscan)**: 3 Hours vs. 16 Minutes
Huge Speed-Ups Across Many Fields

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Field</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Electron Repulsion Integral</td>
<td>Quantum Chemistry</td>
<td>130X</td>
</tr>
<tr>
<td>Lattice Boltzmann</td>
<td>CFD</td>
<td>123X</td>
</tr>
<tr>
<td>Euler Solver</td>
<td>CFD</td>
<td>16X</td>
</tr>
<tr>
<td>GROMACS</td>
<td>Molecular Dynamics</td>
<td>137X</td>
</tr>
<tr>
<td>Lattice QCD</td>
<td>Physics</td>
<td>30X</td>
</tr>
<tr>
<td>Multifrontal Solver</td>
<td>FEA</td>
<td>20X</td>
</tr>
<tr>
<td>nbody</td>
<td>Astrophysics</td>
<td>100X</td>
</tr>
<tr>
<td>Simultaneous Iterative Reconstruction Technique</td>
<td>Computed Tomography</td>
<td>32X</td>
</tr>
</tbody>
</table>
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

CUDA C/C++
- Over 90,000 developers
- Running in Production since 2008
- SDK + Libs + Visual Profiler and Debugger

OpenCL
- 1st GPU demo
- Shipped 1st OpenCL Conformant Driver
- Public Availability (Since April)

Direct Compute
- Microsoft API for GPU Computing
- Supports all CUDA-Architecture GPUs (DX10 and DX11)

Fortran
- PGI Accelerator
- PGI CUDA Fortran
- NOAA Fortran bindings
- FLAGON

Python, Java, .NET, ...
- PyCUDA
- jCUDA
- CUDA.NET
- OpenCL.NET

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

Your GPU Computing Application

Application Acceleration Engines (AXEs)
Middleware, Modules & Plug-ins

Foundation Libraries
Low-level Functional Libraries

Development Environment
Languages, Device APIs, Compilers, Debuggers, Profilers, etc.

CUDA Architecture
CUDA C/C++ Update

- **2007**
  - July 07: CUDA Toolkit 1.0
    - C Compiler
    - Win XP 64
    - Single Precision
    - BLAS
    - FFT
    - SDK
    - 40 examples
  - Nov 07: CUDA Toolkit 1.1
    - C Compiler
    - C Extensions
    - Single Precision
    - Win XP 64
    - Multi-GPU support
    - SDK
  - Atomics support

- **2008**
  - April 08: CUDA Visual Profiler 2.2
    - cudacuda-gdb HW Debugger
  - Aug 08: CUDA Toolkit 2.0
    - Double Precision
    - Compiler Optimizations
    - Vista 32/64
    - Mac OSX
    - 3D Textures
    - Performance enhancements

- **2009**
  - July 09: CUDA Toolkit 2.3
    - DP FFT
    - 16-32 Conversion intrinsics
  - Nov 09: CUDA Toolkit 3.0 Beta
    - C++ Functionality
    - Fermi Arch support
    - Tools
    - Driver and RT

© 2009 NVIDIA Corporation
Fortran Language Solutions

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

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

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

- 2 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

R195
- OpenCL ICD
- OpenGL Interoperability
- Double Precision
- NVIDIA Compiler Optimization Flags
- Query for Compute Capability
- Byte Addressable Stores
- 32-bit Atomics
- Images
- NVIDIA OpenCL Visual Profiler
- NVIDIA OpenCL Code Samples

R190
- OpenCL 1.0 Driver
  Min. Spec. NV R190 released June 2009

Conformance
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></td>
</tr>
<tr>
<td>PGI CUDA Fortran</td>
<td>Auto Parallelizing Compiler</td>
</tr>
<tr>
<td></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>
</tbody>
</table>
NVIDIA Developer Resources

DEVELOPMENT TOOLS
- CUDA Toolkit: Complete GPU computing development kit
- Visual Profiler: GPU hardware profiler for CUDA C and OpenCL
- Nexus: Development environment with Visual Studio integration [beta]
- NVPerfKit: OpenGL/D3D performance tools
- FX Composer: Shader Authoring IDE

SDKs AND CODE SAMPLES
- GPU Computing SDK: CUDA C, OpenCL, DirectCompute
- Graphics SDK: DirectX & OpenGL code samples
- PhysX SDK: Complete game physics solution
- OpenAutomate: Test automation SDK

VIDEO LIBRARIES
- Video Decode Acceleration: NVCUVID, DXVA, Win7 MFT
- Video Encode Acceleration: NYCUVID, Win7 MFT
- Post-Processing: Noise reduction / De-interlace / Polyphase scaling / Color process

ENGINES & LIBRARIES
- NPP Image Libraries: Performance primitives for imaging
- Numeric Libraries: cuFFT, cuLA, cuBLAS
- App Acceleration Engines: Optimized software modules for GPU acceleration
- Shader Library: Shader and post processing
- Optimization Guides: GPU computing and Graphics development best practices

DEVELOPMENT TOOLS
- CUDA Toolkit: Complete GPU computing development kit
- Visual Profiler: GPU hardware profiler for CUDA C and OpenCL
- Nexus: Development environment with Visual Studio integration [beta]
- NVPerfKit: OpenGL/D3D performance tools
- FX Composer: Shader Authoring IDE

SDKs AND CODE SAMPLES
- GPU Computing SDK: CUDA C, OpenCL, DirectCompute
- Graphics SDK: DirectX & OpenGL code samples
- PhysX SDK: Complete game physics solution
- OpenAutomate: Test automation SDK

VIDEO LIBRARIES
- Video Decode Acceleration: NVCUVID, DXVA, Win7 MFT
- Video Encode Acceleration: NYCUVID, Win7 MFT
- Post-Processing: Noise reduction / De-interlace / Polyphase scaling / Color process

ENGINES & LIBRARIES
- NPP Image Libraries: Performance primitives for imaging
- Numeric Libraries: cuFFT, cuLA, cuBLAS
- App Acceleration Engines: Optimized software modules for GPU acceleration
- Shader Library: Shader and post processing
- Optimization Guides: GPU computing and Graphics development best practices

© 2009 NVIDIA Corporation
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
NVIDIA Application Acceleration Engines

**OptiX** – ray tracing engine
- Programmable GPU ray tracing pipeline that greatly accelerates general ray tracing tasks
- Supports programmable surfaces and custom ray data

**PhysX** – physics and dynamics engine
- GPU computed physics – fast enough for real time.
- In use across games, DCC tools, and simulation

**SceniX** – scene management engine
- High performance OpenGL scene graph built around CgFX for maximum interactive quality
- Provides ready access to new GPU capabilities & engines
“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>Compute</td>
</tr>
<tr>
<td><strong>“Nexus”</strong> Pro**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>“Nexus”</strong></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

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

<table>
<thead>
<tr>
<th>Feature</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>
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>CUDA C/C++</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OpenCL</td>
<td>Coming Soon ✓</td>
<td>Coming Soon ✓</td>
<td>✓</td>
<td>Coming Soon ✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fortran</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-gdb

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
  OpenCL Visual Profiler for Windows and Linux

Included in the CUDA Toolkit
Massively Parallel Development (summary)
Tools for Linux - Language / API Support

<table>
<thead>
<tr>
<th></th>
<th>CUDA C/C++</th>
<th>OpenCL</th>
<th>Fortran</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPS HMPP</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PGI Accelerators</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGI CUDA Fortran</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>NV cuda-gdb</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allinea DDT</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>TotalView Debugger</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NV Visual Profiler</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>TAU CUDA</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Mcore Platform Analyzer</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.
200+ Universities Teaching GPU Computing on CUDA
Thank You