Outline

- Execution Model
- Architecture
- Demo
Execution Model
Software Architecture

Applications

DX10  OpenGL  OpenCL  CUDA C

Host OS

Device Driver

SIMT GPU

Hardware Thread Scheduling

Hundreds of Cores

Thousands of Threads

Scalar CPU
SIMT Multithreaded Execution

- **SIMT**: Single-Instruction Multi-Thread applies instruction to independent threads
- **SIMT** provides easy single-thread scalar programming with SIMD efficiency
- **Warp**: the set of 32 parallel threads that execute a SIMT instruction
- Hardware implements zero-overhead warp and thread scheduling
- Each thread processor manages state for up to 128 threads
- **SIMT** threads execute independently
- **SIMT** warp diverges and converges when threads branch independently
- Best efficiency and performance when threads of a warp execute together

Single-Instruction Multi-Thread instruction scheduler
Execution Model - CUDA

- **Local Memory**: per-thread
  - Private per thread
  - Auto variables, register spill

- **Shared Memory**: per-block
  - Shared by threads of CTA
  - Inter-thread communication

- **Global Memory**: per-application
  - Shared by all threads
  - Inter-Grid communication

**Thread**

- Local Memory
- Register File

**Block**

- Shared Memory

**Grid 0**

- Sequential Grids in Time

**Grid 1**

- Global Memory
Execution Model - Graphics

- Pixel Thread
  - Register File

- Pixel Warp
  - Plane Equations
  - Texture

- Pixel Grid 0
- Pixel Grid 1

- Local Registers: per Pixel
  - Private per pixel

- Planes/Textures: per Warp
  - Define surface-space inputs
  - Array of 1D/2D/3D data arrays

- Target Images: per Grid
  - Array of 2D surfaces
Architecture
Implementation

- TSMC 65nm
- 1.4 Billion Transistors
- 24.3 x 24.0 mm die
- 2236 Ball BGA

- 1 TeraFLOPS SP / 84 GigaFLOPS DP
  - 1.4 GHz Processor Clock
  - 140 GB/s
  - 1.1 GHz Memory Clock
- Up to 4GB on-board Memory
GTX200 Unified Visual Computing

Tesla Unified Graphics and Computing Architecture
Scales parallel performance 2X beyond G80
240 Thread Processor cores, 30K threads
Double precision 64-bit IEEE 754 floating point
SM Multithreaded Multiprocessor

- 8 SP Thread Processors
  - IEEE 754 32-bit floating point
  - 32-bit and 64-bit integer
  - 2K 32-bit registers per SP
  - Variable 4-128 registers / thread
- 2 SFU Special Function Units
- 1 DP Double Precision Unit
  - IEEE 754R 64-bit floating point
  - Fused multiply-add
- Scalar register-based ISA
- Multithreaded Instruction Unit
  - 1024 threads, hardware multithreaded
  - 32 SIMT warps of 32 threads
  - Independent thread execution
  - Mixed concurrent thread types
- 16KB Shared Memory
  - Concurrent threads share data
  - Low latency load/store
Double Precision Fused Multiply-Add

- Revised IEEE 754R standard specifies FMA
- FMA has lower latency than FMAD
- FMA improves accuracy of many computations
  - Dot products
  - Matrix multiplication
- Enables fast mixed-precision convergence algorithms
- Enables higher performance algorithms for extended-precision arithmetic
- Enables efficient implementation of exactly-rounded division and square root
Coalesced Memory I/O

- When 16 threads access a contiguous region of device memory
- 16 data elements loaded in one instruction
  - int, float: 64 bytes (fastest)
  - int2, float2: 128 bytes
  - int4, float4: 256 bytes (2 transactions)
- Regions aligned to multiple of size
- Coalescing scales gracefully for partially contiguous accesses
Performance: Single Precision BLAS

BLAS (SGEMM) on CUDA

CUBLAS: CUDA 2.0b2, Tesla C1060
ATLAS 3.81 on Dual 2.8GHz Opteron Dual-Core

Matrix Size

GFLOPS
Performance: Double Precision BLAS

BLAS (DGEMM) on CUDA

CUBLAS CUDA 2.0b2 on Tesla C1060
ATLAS 3.81 on Intel Xeon E5440 Quad-core, 2.83 GHz

Matrix Size

<table>
<thead>
<tr>
<th>Matrix Size</th>
<th>256x256</th>
<th>256x512</th>
<th>512x512</th>
<th>1024x512</th>
<th>1024x1024</th>
<th>2048x1024</th>
<th>2048x2048</th>
<th>4096x2048</th>
<th>4096x4096</th>
<th>8192x4096</th>
<th>8192x8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFLOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance: Scaling

Oil and Gas Computing: Reverse Time Migration
Hand Optimized SSE Versus CUDA C

Billions of Points / sec

<table>
<thead>
<tr>
<th>Number of Cores</th>
<th>X86 CPU</th>
<th>NVIDIA GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>384</td>
<td>384</td>
<td>384</td>
</tr>
</tbody>
</table>
Summary

- Rebalanced architecture to workload trends
- Scaled from 128 to 240 processors
- Hardware manages thousands of threads
  - Zero software overhead
  - Hides huge latencies
  - High achieved utilization
- Natively Scalar
  - No swizzling or vectorization overhead
  - Coalescing for high bandwidth memory I/O
- Software architecture allows 2X scaling on customer C code with no modification
Demo
More Information


http://www.nvidia.com/cuda