High Performance Computing
David B. Kirk, Chief Scientist, NVIDIA
A History of Innovation

- Invented the Graphics Processing Unit (GPU)
- Pioneered programmable shading
- Over 2000 patents*

![Image showing graph with years and corresponding transistor counts]

<table>
<thead>
<tr>
<th>Year</th>
<th>Model</th>
<th>Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>NV1</td>
<td>1 Million</td>
</tr>
<tr>
<td>1999</td>
<td>GeForce 256</td>
<td>22 Million</td>
</tr>
<tr>
<td>2002</td>
<td>GeForce4</td>
<td>63 Million</td>
</tr>
<tr>
<td>2003</td>
<td>GeForce FX</td>
<td>130 Million</td>
</tr>
<tr>
<td>2004</td>
<td>GeForce 6</td>
<td>222 Million</td>
</tr>
<tr>
<td>2005</td>
<td>GeForce 7</td>
<td>302 Million</td>
</tr>
<tr>
<td>2006-2007</td>
<td>GeForce 8</td>
<td>754 Million</td>
</tr>
<tr>
<td>2008</td>
<td>GeForce GTX 200</td>
<td>1.4 Billion</td>
</tr>
</tbody>
</table>
Real-time Ray Tracing Demo

- Real system
- NVSG-driven animation and interaction
- Programmable shading
- Modeled in Maya, imported through COLLADA
- Fully ray traced

2 million polygons
Bump-mapping
Movable light source
5 bounce reflection/refraction
Adaptive antialiasing
Poznaj świadczenia z dokumentacji procesu beatyfikacyjnego i kanonizacyjnego Jana Pawła II

CUDA

już od 7 marca

Książka wraz z filmem VCD "CUDA" dostępna w księgarniach, parafiach, salonach EMPIK, w księgarniach internetowych i na stronie www.stanislawbm.pl tel. 012 429 52 17
CUDA Uses Kernels and Threads for Fast Parallel Execution

- Parallel portions of an application are executed on the GPU as kernels
  - One kernel is executed at a time
  - Many threads execute each kernel

- Differences between CUDA and CPU threads
  - CUDA threads are extremely lightweight
    - Very little creation overhead
    - Instant switching
  - CUDA uses 1000s of threads to achieve efficiency
void saxpy_serial(int n, float a, float *x, float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}

// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);

__global__ void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n)  y[i] = a*x[i] + y[i];
}

// Invoke parallel SAXPY kernel with 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
The Key to Computing on the GPU

- Standard high level language support
  - C, soon C++ and Fortran
  - Standard and domain specific libraries
- Hardware Thread Management
  - No switching overhead
  - Hide instruction and memory latency
- Shared memory
  - User-managed data cache
  - Thread communication / cooperation within blocks
- Runtime and tool support
  - Loader, Memory Allocation
  - C stdlib
CUDA

Heterogeneous Computing

Oil & Gas  Finance  Medical  Biophysics  Numerics  Audio  Video  Imaging

© 2008 NVIDIA Corporation.
Universities Teaching Parallel Programming With CUDA

- Duke
- Erlangen
- ETH Zurich
- Georgia Tech
- Grove City College
- Harvard
- IIIT
- IIT
- Illinois Urbana-Champaign
- INRIA
- Iowa
- ITESM
- Johns Hopkins
- Kent State
- Kyoto
- Lund
- Maryland
- McGill
- MIT
- North Carolina - Chapel Hill
- North Carolina State
- Northeastern
- Oregon State
- Pennsylvania
- Polimi
- Purdue
- Santa Clara
- Stanford
- Stuttgart
- Suny
- Tokyo
- TU-Vienna
- USC
- Utah
- Virginia
- Washington
- Waterloo
- Western Australia
- Williams College
- Wisconsin
Wide Developer Acceptance

- Interactive visualization of volumetric white matter connectivity (146X)
- Ionic placement for molecular dynamics simulation on GPU (36X)
- Transcoding HD video stream to H.264 (19X)
- Simulation in Matlab using .mex file CUDA function (17X)
- Astrophysics N-body simulation (100X)
- Financial simulation of LIBOR model with swaptions (149X)
- GLAME@lab: An M-script API for linear Algebra operations on GPU (47X)
- Ultrasound medical imaging for cancer diagnostics (20X)
- Highly optimized object oriented molecular dynamics (24X)
- Cmatch exact string matching to find similar proteins and gene sequences (30X)
Folding@home on GeForce / CUDA

186x Faster Than CPU

CPU
PS3
Radeon HD 4870
GeForce GTX 280

4
100
220
746
Faster is not “just Faster”

- 2-3X faster is “just faster”
  - Do a little more, wait a little less
  - Doesn’t change how you work
- 5-10x faster is “significant”
  - Worth upgrading
  - Worth re-writing (parts of) the application
- 100x+ faster is “fundamentally different”
  - Worth considering a new platform
  - Worth re-architecting the application
  - Makes new applications possible
  - Drives “time to discovery” and creates fundamental changes in Science
Tesla T10: 1.4 Billion Transistors

Thread Processor Cluster (TPC)

Thread Processor Array (TPA)

Thread Processor
Tesla 10-Series

Double the Performance

1 Teraflop

500 Gigaflops

Tesla 8  Tesla 10

Double the Memory

1.5 Gigabytes  4 Gigabytes

Tesla 8  Tesla 10

Double Precision

Finance  Science  Design

© 2008 NVIDIA Corporation.
**Tesla T10 Double Precision Floating Point**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>IEEE 754</td>
</tr>
<tr>
<td>Rounding modes for FADD</td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
</tr>
<tr>
<td>Denormal handling</td>
<td>Full speed</td>
</tr>
<tr>
<td>NaN support</td>
<td>Yes</td>
</tr>
<tr>
<td>Overflow and Infinity support</td>
<td>Yes</td>
</tr>
<tr>
<td>Flags</td>
<td>No</td>
</tr>
<tr>
<td>FMA</td>
<td>Yes</td>
</tr>
<tr>
<td>Square root</td>
<td>Software with low-latency FMA-based convergence</td>
</tr>
<tr>
<td>Division</td>
<td>Software with low-latency FMA-based convergence</td>
</tr>
<tr>
<td>Reciprocal estimate accuracy</td>
<td>24 bit</td>
</tr>
<tr>
<td>Reciprocal sqrt estimate accuracy</td>
<td>23 bit</td>
</tr>
<tr>
<td>log2(x) and 2^x estimates accuracy</td>
<td>23 bit</td>
</tr>
</tbody>
</table>

© 2008 NVIDIA Corporation.
Double the Performance Using T10

**DNA Sequence Alignment**
- G80
- T10P

**Dynamics of Black Holes**
- CPU
- G80
- T10

**Video Application**
- G80
- T10P

**Cholesky Factorization**
- G80
- T10P

**LB Flow Lighting**
- G80
- T10P

**Ray Tracing**
- G80
- T10P

**Reverse Time Migration**
- G80, 1.5 GB
- T10P, 1.5 GB
- T10P, 4 GB

© 2008 NVIDIA Corporation.
How to Get to 100x?

Traditional Data Center Cluster

Quad-core CPU

cores per server

1000’s of cores
1000’s of servers

More Servers To Get More Performance
Heterogeneous Computing Cluster

10,000’s processors per cluster

- Hess
- NCSA / UIUC
- JFCOM
- SAIC
- University of North Carolina
- Max Plank Institute
- Rice University
- University of Maryland
- GusGus
- Eotvas University
- University of Wuppertal
- IPE/Chinese Academy of Sciences
- Cell phone manufacturers
Building a 100TF datacenter

<table>
<thead>
<tr>
<th>CPU 1U Server</th>
<th>4 GPUs: 960 cores</th>
<th>Tesla 1U System</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 CPU cores</td>
<td>4 GPUs: 960 cores</td>
<td></td>
</tr>
<tr>
<td>0.07 Teraflop</td>
<td>4 Teraflops</td>
<td></td>
</tr>
<tr>
<td>$ 2000</td>
<td>$ 8000</td>
<td></td>
</tr>
<tr>
<td>400 W</td>
<td>800 W</td>
<td></td>
</tr>
<tr>
<td>1429 CPU servers</td>
<td>25 CPU servers</td>
<td></td>
</tr>
<tr>
<td>$ 3.1 M</td>
<td>25 Tesla systems</td>
<td></td>
</tr>
<tr>
<td>571 KW</td>
<td>$ 0.31 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>27 KW</td>
<td></td>
</tr>
</tbody>
</table>

10x lower cost
21x lower power

© 2008 NVIDIA Corporation.
Tesla S1070 1U System

4 Teraflops\(^1\)
800 watts\(^2\)

\(^1\) single precision
\(^2\) typical power
Tesla C1060 Computing Processor

957 Gigaflops\(^1\)

160 watts\(^2\)

\(^1\) single precision
\(^2\) typical power
Application Software
Industry Standard C Language

Libraries
- cuFFT
- cuBLAS
- cuDPP

System
- 1U PCI-E Switch

CUDA Compiler
- C Fortran Multi-core

CUDA Tools
- Debugger
- Profiler

4 cores

240 cores

© 2008 NVIDIA Corporation.
CUDA Source Code

Industry Standard C Language

Industry Standard Libraries

CUDA Compiler
C Fortran

Standard
Debugger Profiler

Multi-core

© 2008 NVIDIA Corporation.
CUDA 2.1: Many-core + Multi-core support

C CUDA Application

NVCC
Many-core PTX code
PTX to Target Compiler
Many-core

NVCC --multicore
Multi-core CPU C code
gcc and MSVC
Multi-core

© 2008 NVIDIA Corporation.
CUDA Everywhere!