A History of Innovation

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

1995
NV1
1 Million Transistors

1999
GeForce 256
22 Million Transistors

2002
GeForce4
63 Million Transistors

2003
GeForce FX
130 Million Transistors

2004
GeForce 6
222 Million Transistors

2005
GeForce 7
302 Million Transistors

2006-2007
GeForce 8
754 Million Transistors

2008
GeForce GTX 200
1.4 Billion Transistors
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 „Cud” 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<<<nbblocks, 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 Compiler Downloads

- July
- August
- September
- October
- November
- December
- January
- February
- March
- April
### 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

146X
Interactive visualization of volumetric white matter connectivity

36X
Ionic placement for molecular dynamics simulation on GPU

19X
Transcoding HD video stream to H.264

17X
Simulation in Matlab using .mex file CUDA function

100X
Astrophysics N-body simulation

149X
Financial simulation of LIBOR model with swaptions

47X
GLAME@lab: An M-script API for linear Algebra operations on GPU

20X
Ultrasound medical imaging for cancer diagnostics

24X
Highly optimized object oriented molecular dynamics

30X
Cmatch exact string matching to find similar proteins and gene sequences

© 2008 NVIDIA Corporation.
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

Die Photo of Tesla T10
Double the Performance

Tesla 10-Series

Double the Memory

Double Precision

Tesla 8

Tesla 10

1 Teraflop

500 Gigaflops

1.5 Gigabytes

4 Gigabytes

Tesla 8

Tesla 10

Finance
Science
Design
## Tesla T10 Double Precision Floating Point

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>IEEE 754</td>
</tr>
<tr>
<td><strong>Rounding modes for FADD</strong></td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
</tr>
<tr>
<td><strong>Denormal handling</strong></td>
<td>Full speed</td>
</tr>
<tr>
<td><strong>NaN support</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Overflow and Infinity support</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>FMA</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Square root</strong></td>
<td>Software with low-latency FMA-based convergence</td>
</tr>
<tr>
<td><strong>Division</strong></td>
<td>Software with low-latency FMA-based convergence</td>
</tr>
<tr>
<td><strong>Reciprocal estimate accuracy</strong></td>
<td>24 bit</td>
</tr>
<tr>
<td><strong>Reciprocal sqrt estimate accuracy</strong></td>
<td>23 bit</td>
</tr>
<tr>
<td><strong>log2(x) and 2^x estimates accuracy</strong></td>
<td>23 bit</td>
</tr>
</tbody>
</table>
Double the Performance Using T10

DNA Sequence Alignment

Dynamics of Black holes

Video Application

Cholesky Factorization

LB Flow Lighting

Ray Tracing

Reverse Time Migration

© 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

1928 processors 1928 processors
Building a 100TF datacenter

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

**10x lower cost**

**21x lower power**
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
CUDA Source Code
Industry Standard C Language

Industry Standard Libraries

CUDA Compiler
C Fortran

Standard
Debugger Profiler

Multi-core
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
CUDA Everywhere!