NVIDIA GPU Computing

IDRIS - December 18th 2008

Jean-Christophe Baratault
jbaratault@nvidia.com
GPU as CPU coprocessor: Not a new idea


Source: www.gpgpu.org
GPU: Parallelized Architecture for years

Quadro FX 3000
2003

Vertex
3
Pixel
4x2

Quadro FX 4400
2004

Vertex
8
Pixel
24

Quadro FX 4500
2005
8 vertex pipes
24 pixel pipes
Why didn’t GPU Computing took off sooner?

- **GPU Architecture**
  - Gaming oriented, process pixel for display
  - Single threaded operations
  - No shared memory

- **Development Tools**
  - Graphics oriented (OpenGL, GLSL)
  - University research (Brook)
  - Assembly language

- **Deployment**
  - Gaming solutions with limited lifetime
  - Expensive OpenGL professional graphics boards
  - No HPC compatible products
NVIDIA invested in GPU Computing back in 2004

- Strategic move for the company
  - Expand GPU architecture beyond pixel processing
  - Future platforms will be hybrid, multi/many cores based

- Hired key industry experts
  - x86 architecture
  - x86 compiler
  - HPC hardware specialist

Provide a GPU based Compute Ecosystem by 2008
The Past 2 years

2006
- G80, first GPU with built-in Compute features
  - 128 core, scalable multi-threaded architecture
- CUDA SDK Beta

2007
- Tesla HPC product line
- CUDA SDK 1.0, 1.1
- University trainings programs
#1 GPU Architecture
G80 – GPU Architecture Tuned for Compute

- Processors execute computing threads
- Thread Execution Manager issues threads
- 128 **Thread Processors** grouped into 16 **Multiprocessors**
- Parallel Data Cache (Shared Memory) enables thread cooperation
G80GL
GL = OpenGL
Ultra high end Quadro FX

G84GL
High end Quadro FX
96 cores

G86GL
Mid range Quadro FX
64 cores

G80
All GPU features except OpenGL
Ultra high end GeForce

G84
Mid-range GeForce

G86
Mid-range GeForce

Same master architecture but less cores for each product market segment
June 2008: NVIDIA GT200 GPU
2nd Generation Parallel Computing Architecture

1.4 billion transistors
933 GFlops
240 processing cores

NVIDIA GPU marketing names

- GT200  Consumer GeForce
- GT200GL  Professional Quadro
- T10  HPC Tesla

GT200, GT200GL and T10 are based on the same master architecture but different features are enabled for each target market
GT200 / GT200GL / T10

Thread Processor (TP)
- Multi-banked Register File
- FP/Int
- SpcOps ALUs

Special Function Unit (SFU)
- Double Precision

TP Array Shared Memory

- 240 SP thread processors
- 30 DP thread processors
- Full scalar processor
- IEEE 754 64-bit floating point
Asynchronous transfer is used to hide data transfer from CPU to GPU or GPU to CPU while the GPU is processing data, thus improving application speedup.
Ever Increasing Floating Point Performance

NVIDIA GPUs Floating Point Performance

Peak Gigaflops

G8x

GT200
GPU Computing : Heterogeneous Computing

Computing with CPU + GPU

*Heterogeneous Computing*
\[ y[i] = a \times x[i] + y[i] - \text{Computed Sequentially} \]
$y[i] = a \times x[i] + y[i]$ – Computed In Parallel

$= a \times$

$X$

$Y$

$= +$

$y'$
#2 CUDA SDK
CUDA is C for Parallel Processors

- CUDA is industry-standard C
  - Write a program for one thread
  - Instantiate it on many parallel threads
  - Familiar programming model and language

- CUDA is a scalable parallel programming model
  - Program runs on any number of processors without recompiling

- CUDA parallelism applies to both CPUs and GPUs
  - Compile the same program source to run on different platforms with widely different parallelism
  - Map to CUDA threads to GPU threads or to CPU vectors
Heterogeneous Programming

CUDA = serial program with parallel kernels, all in C
- Serial C code executes in a host thread (i.e. CPU thread)
- Parallel kernel C code executes in many device threads across multiple processing elements (i.e. GPU threads)
One kernel is executed at a time on the GPU
Many threads execute each kernel
  - Each thread executes the same code…
  - … on different data based on its `threadID`
**Device** = GPU = set of multiprocessors
**Multiprocessor** = set of processors & shared memory

**Kernel** = GPU program
**Grid** = array of thread blocks that execute a kernel
**Thread block** = group of SIMD threads
**Warp** = group of threads
Transparent Scalability

- Hardware is free to schedule thread blocks on any processor
  - So they can run in any order, concurrently or sequentially
- This independence gives scalability
  - A kernel scales across parallel multiprocessors
CUDA Language: C with Minimal Extensions

- Philosophy: provide minimal set of extensions necessary to expose power

- Declaration specifiers to indicate where things live
  
  ```
  __global__ void KernelFunc(...); // kernel function, runs on device
  __device__ int GlobalVar; // variable in device memory
  __shared__ int SharedVar; // variable in per-block shared memory
  ```

- Extend function invocation syntax for parallel kernel launch
  
  ```
  KernelFunc<<<500, 128>>>(...); // launch 500 blocks w/ 128 threads each
  ```

- Special variables for thread identification in kernels
  
  ```
  dim3 threadIdx; dim3 blockIdx; dim3 blockDim; dim3 gridDim;
  ```

- Intrinsics that expose specific operations in kernel code
  
  ```
  __syncthreads(); // barrier synchronization within kernel
  ```
Simple “C” Description For Parallelism

```c
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);
```
CUDA Toolkit

The CUDA development environment includes:

- nvcc C compiler
- CUDA FFT and BLAS libraries for the GPU
- Profiler
- gdb debugger for the GPU
- CUDA runtime driver (also available in the standard NVIDIA GPU driver)
- CUDA programming manual
MATLAB plug-in for CUDA

This MATLAB plug-in for CUDA provides acceleration of standard MATLAB 2D FFTs and CUDA MEX example plug-in and build environment using Chris Brehmter's Fourier spectral simulation of 2D fluid flow MATLAB scripts from his course materials at the University of Washington.

When MATLAB makes 2D FFT calls of any size, the NVIDIA plug-in intercepts them and handles them with a MEX file that in-turn utilizes an optimized CUDA FFT implementation on the GPU.

This is transparent to MATLAB users. Note: this current implementation uses a single-precision 2-FFT on the NVIDIA hardware, so the results are not in 64-bit precision that the native MATLAB implementation uses without the NVIDIA plug-in. Can be run with and without CUDA acceleration. Time to run the example shows a 14X speedup (from 216 seconds to 15 seconds using CUDA via the MEX file interface).

The MEX file example and build environment uses an F8_2DFlow example but the method is illustrative on how to write a custom CUDA interface via a MATLAB MEX file interface for all CUDA libraries.

[Download] MathWorks MATLAB® Plug-in for Linux
[Download] MathWorks MATLAB® Plug-in for Windows
[Download] Whitepaper: Accelerating MathWorks MATLAB® with CUDA

Previous Versions
[Download] CUDA 1.0 Plugin for Linux
[Download] CUDA 1.0 Plugin for Windows

Using MATLAB on Linux, the results for the computation of the advection of an elliptic vortex on a 256 x 256 mesh, stream function (left) and vorticity (right) in Figure 1 required 168 seconds. By contrast, the results using MATLAB with CUDA in Figure 2 required only 14.9 seconds.

For a better comparison of the quality of the results, we ran a 2D isotropic turbulence simulation compared the vorticity power spectra of the different runs. The first used the original MATLAB code (Figure 3) and the second used MATLAB accelerated with CUDA code (Figure 4). Even for this quantity, that is very sensitive to fine scales, the results are in close agreement.

Figure 3. Final Results Using MATLAB

Figure 4. Final Results Using MATLAB with CUDA
Pseudo-spectral simulation of 2D Isotropic turbulence.

512x512 mesh, 400 RK4 steps, Windows XP, MATLAB file http://www.amath.washington.edu/courses/571-winter-2006/matlab/FS_2Dturb.m

MATLAB 992 seconds

MATLAB with CUDA (single precision FFTs) 93 seconds
CUDA SDK Code Samples

- CUDA Basic Topics
- CUDA Advanced Topics
- Computational Finance
- Parallel Algorithms
- Linear Algebra
- Physically-Based Simulation
- Texture
- Video Decode
- Image/Video Processing
- Graphics Interop
- Performances Strategies
CUDA 2.0: 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 Zone: www.nvidia.com/cuda
CUDA Tutorial

Latest and greatest on www.nvidia.com/object/cuda_education.html

NVIDIA CUDA Tutorial, SuperComputing 2008 Austin Nov08
www.gpgpu.org/sc2008

Introduction (PDF)
Parallel Programming with CUDA (PDF)
CUDA Toolkit (PDF)
Optimizing CUDA (PDF)
Seismic Imaging on NVIDIA GPUs: Algorithms and Porting & Production Experiences (PDF)
Molecular Visualization and Analysis (PDF)
Molecular Dynamics (PDF)
Computational Fluid Dynamics (PDF)
NVIDIA CUDA French Partners
Training & Development

- CAPS
- ANEO
- Scalable Graphics
- GPU-Tech
- HPC Project
OpenCL
OpenCL

- A new compute API for parallel programming of heterogeneous systems
- Allows developers to harness the compute power of BOTH the GPU and the CPU
- A multi-vendor standards effort managed through the Khronos Group
NVIDIA and OpenCL

- **OpenCL is terrific**
  We support any initiative that unleashes the massive power of the GPU

- Neil Trevett, NVIDIA VP, chairs Khronos OpenCL working group - several active NVIDIA participants

- NVIDIA is working closer with Apple since the inception of OpenCL
  - OpenCL was developed on NVIDIA GPUs
  - First to show working OpenCL
  - Top to bottom supplier of GPUs for new Apple notebooks
OpenCL and C for Cuda

Entry point for developers who want low-level API

Entry point for developers who prefer high-level C

Shared back-end compiler and optimization technology

C for Cuda

OpenCL

PTX

GPU
Different Programming Styles

- **C for CUDA**
  - C with parallel keywords
  - C runtime that abstracts driver API
  - Memory managed by C runtime
  - Generates PTX

- **OpenCL**
  - Hardware API - similar to OpenGL
  - Programmer has complete access to hardware device
  - Memory managed by programmer
  - Generates PTX
NVIDIA’s OpenCL Roadmap

<table>
<thead>
<tr>
<th>2008</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
<td>Q3</td>
</tr>
<tr>
<td>Q4</td>
<td>Q4</td>
</tr>
</tbody>
</table>
OpenCL
The Open Standard for Heterogeneous Parallel Programming

Neil Trevett
President, Khronos Group and OpenCL Chair
SIGGRAPH Asia, December 2008
OpenCL and the Khronos Ecosystem

- **OpenCL**: Heterogeneous Parallel Computing
- **OpenGL**: Cross platform desktop 3D
- **OpenGL ES**: Embedded 3D
- **OpenMAX**: Streaming Media and Image Processing
- **OpenVG**: Vector 2D
- **OpenSL ES**: Enhanced Audio
- **EGL**: Surface and synch abstraction
- **OpenKODE**: Mobile OS Abstraction
- **OpenKGS**: Integrated Mixed-media Stack

**Parallel computing and visualization in scientific and consumer applications**

- **Collada**: 3D Asset Interchange Format

**Hundreds of man years invested by industry experts to create coordinated ecosystem**

**Umbrella specifications for mobile application portability**
Restrictions

• Pointers to functions are not allowed
• Pointers to pointers allowed within a kernel, but not as an argument
• Bit-fields are not supported
• Variable length arrays and structures are not supported
• Recursion is not supported
• Writes to a pointer of types less than 32-bit are not supported
• Double types are not supported, but reserved
• 3D Image writes are not supported

• Some restrictions are addressed through extensions
OpenCL
Source Code Examples

Mark Harris
NVIDIA Developer Technology
December 2008
MS DirectX 11
Microsoft is not part of the Khronos consortium

Microsoft is developing a competing technology called **DirectX 11 Compute**

MS DirectX SDK November 2008 - Compute Shader

*The Compute Shader is an additional stage independent of the Direct3D 11 pipeline that enables general purpose computing on the GPU. In addition to all shader features provided by the unified shader core, the Compute Shader also supports scattered reads and writes to resources through Unordered Access Views, a shared memory pool within a group of executing threads, synchronization primitives, atomic operators, and many other advanced data-parallel features.*

#3 Deployment Products
CUDA Everywhere but how?

- PC cluster with multiple GPUs
- HPC server rack
- Desktop with single GPU
- Laptop
- Linux
- NVIDIA Support?
- GeForce?
- WinXP
- Desktop with multiple GPUs
- No video output?
- Tesla?
- Quadro?
- 2D/3D real-time display?
Parallel Computing on All GPUs

Over 90 Million CUDA Compatibles GPUs since Nov 2006

GeForce
Entertainment

Quadro
Design & Creation

Tesla
High-Performance Computing

CUDA Compatible
# Selecting a CUDA Platform

<table>
<thead>
<tr>
<th>Feature</th>
<th>Tesla</th>
<th>Quadro</th>
<th>GeForce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stress tested and burned-in with added margin for numerical accuracy</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufactured by NVIDIA with professional grade memory</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NVIDIA care: 3-year warranty from NVIDIA, enterprise support</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4 Gigabyte on-board memory for large technical computing data sets</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Single card solution for professional visualization and CUDA computing</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Consumer middle-ware and applications: PhysX, Video, Imaging</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Consumer product life cycle</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Manufactured and guaranteed by NVIDIA graphics add-in card partners</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Product support through NVIDIA graphics add-in card partners</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
NVIDIA Manufactured Computing Products

- Manufacturing
  - Professional grade memory
    - Manufactured by NVIDIA
- Product Margin Testing
  - Numerical stress testing
    - 100% burn-in
- Tesla Suppliers
  - Tesla Preferred Partners
  - Custom solutions
- Lifetime and Warranty
  - 3-year warranty
  - Extended product life
GeForce

CUDA for consumer applications
CUDA for consumer applications

- Over 80M GeForce CUDA compatible systems
  - Widely available
  - CUDA works on desktop and laptops
  - CUDA available for XP, Vista and MAC OS
    - Single GPU and multi-GPU with SLI technology

- The right platform to develop consumer multimedia applications

- Easy and fast access to CUDA programming model
  - First step for university students to discover CUDA
badaboom!

Ultra-Fast GeForce Video Transcoding

- Up to 19x faster than multi-core CPU
- Over 4 times faster than real-time
CUDA is Taking Over Distributed Computing
Advancing Scientific Discovery

- Announcing 4 new distributed computing platforms on CUDA
  - SETI@home
  - BOINC Platform
  - GPUGRID
  - Einstein@home

- SETI@home client featured with CUDA
  - Up to 10x faster than CPU - No ATI option today
  - Available 12/17 from setiathome.berkeley.edu

- Press release on 12/17
CUDA: Pervasive in Scientific Research

**BOINC**
- Berkeley Open Infrastructure for Network Computing
- Foundation for many distributed compute projects
  - *Now designed for CUDA*

**GPUGRID**
- Biomolecular simulations for scientific research
  - *CUDA speeds up by average of 20x*

**SETI@home**
- Search for extra-terrestrial intelligence by tracking narrow-bandwidth radio signals from space
  - *CUDA speeds up by as much as 10x*

**Einstein@home**
- Enhancing the search for gravitational radiation and discovery of pulsars
  - *Optimizing for CUDA*

**Folding@home**
- Studying protein folding to better understand causes of diseases like Alzheimers and cancer
  - *CUDA speeds up by as much as 10x*
SETI@home

*Powered by GeForce with CUDA*

- 10x faster than an average CPU
- Nearly 2x speed of fastest home PC CPU
- Same performance at 1/10\(^{th}\) the price *

![Bar Chart]

**Telescope Work Units per Day**

<table>
<thead>
<tr>
<th>Graphics Card</th>
<th>CPU 1</th>
<th>CPU 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTX 280</td>
<td>221.0</td>
<td>127.6</td>
</tr>
<tr>
<td>9800GTX Core i7 965 3.2 GHz</td>
<td>129.0</td>
<td></td>
</tr>
<tr>
<td>9800GTX Core2 Duo E8200 2.66 GHz</td>
<td>85.5</td>
<td></td>
</tr>
<tr>
<td>9600GT Core2 Duo E8200 2.66 GHz</td>
<td>33.9</td>
<td></td>
</tr>
<tr>
<td>Core2 Duo E8200 2.66 GHz</td>
<td>22.9</td>
<td></td>
</tr>
<tr>
<td>Phenom 9950 2.66GHz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Based on upgrade options for consumers who own a common Intel Core2 Duo E8200 based PC. Consumer may choose NVIDIA GeForce 9800GTX for $149 or must upgrade entire system to Intel Core i7 965, 3GB DDR3, x58 motherboard for total of $1,449. Prices based on NewEgg as of 12/15/09.
Run SETI@home on your NVIDIA GPU

Most computers are equipped with a Graphics Processing Unit (GPU) which handles their graphical output, including the 3-D animated graphics used in computer games. The computing power of GPUs has increased rapidly, and they are now often much faster than the computer's main processor, or CPU.

NVIDIA (a leading GPU manufacturer) has developed a system called CUDA that uses GPUs for scientific computing. With NVIDIA's assistance, we've developed a version of SETI@home that runs on NVIDIA GPUs using CUDA. This version runs from 5x to 10x faster than the CPU-only version. We urge SETI@home participants to use it if possible.

Just follow these instructions:

1) Check whether your computer has a CUDA-capable GPU
   The CUDA version of SETI@home works on most newer NVIDIA GPUs. To find out if your GPU is compatible:

   - Identify the model name of your GPU. On Windows, click on My Computer / Properties / Hardware / Device Manager, and open Display Adapters. This will show the model name.
   - Check NVIDIA's list of CUDA-enabled products. If your GPU is listed here and has at least 256MB of RAM, it's compatible.

2) Download the latest NVIDIA driver
   CUDA requires a recent driver to work. If you already have a recent NVIDIA driver, you can try going to step 3. If not, or if you have problems, then you should get the CUDA 2.0 driver. Download it from NVIDIA and install it (a reboot will be required). Note: you only need the driver, not the Toolkit or the SDK. Get the CUDA 2.0 version, or the latest release version. Using beta drivers is not guaranteed to work. If BOINC (in the advanced view messages tab) reports that it has found a CUDA GPU, your current driver set may be recent enough to work properly.

3) Install the latest BOINC software
   You'll need version 6.4.4 or later of the BOINC software. Download it from here and install it.

   You're done! Now start up BOINC, and before long you'll be finishing jobs in no time, and racking up big credit numbers.

Also:

- You can use your GPU to help biomedical research as well as SETI: GPUgrid.net, also has a CUDA application (and more projects are on the way).
Folding@home
Powered by GeForce with CUDA

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>PS3</th>
<th>Radeon HD 3870</th>
<th>Radeon HD 4850</th>
<th>GeForce GTX 280</th>
</tr>
</thead>
<tbody>
<tr>
<td>nanoseconds of simulation per day</td>
<td>4</td>
<td>100</td>
<td>170</td>
<td>377</td>
<td>740</td>
</tr>
</tbody>
</table>
What is it?

A volunteer distributed computing project

It is a novel distributed supercomputing infrastructure made of many PlayStation3 and NVIDIA graphics cards joined together to deliver high-performing all-atom biomolecular simulations. This project gives a new powerful computational tool to scientists and you are an important part of it.

Be part of it

If you enjoy science, you can participate by donating computing time to scientific research. Simply follow the instructions below to start, gain your credits for the results you return, join a team, meet and exchange experiences with other participants in the forums.

Want to know more?

Visit our Science page and find out how PlayStation3 and NVIDIA graphics cards can help biomedical research. And visit our Gallery to know more about us and what we do through our pictures and videos.

Join us!

Click on your system and follow the instructions

PlayStation3  NVIDIA Graphics Card

Spotlight

News

Supported RIDMC version is now 4.4.5
December 11, 2005

Please upgrade to version 4.4.5 to have a correct estimation of elapsed time.

GPU Grid and GPU Grids new website in few styles selectable by users.
November 12, 2005

We have created a new resources section which contains project brochures and t-shirt graphics and duplicated retails.

PS3GRID and NVIDIA Grid new website in few styles selectable by users.

Visit this thread in the forums for full details about this machine.

User of the day

Czech Crunchers Unit

Returning participants

PS3GRID: 50
GPU Grids: 0
PS3GRID: 15
NVIDIA Grids: 10

Subscribe to the RSS feed

CPU

CELL

GPU

~ 10-20 GFLOPs/core
~ 230 GFLOPs
~ 1000 GFLOPs
Quadro FX

CUDA for
professional visualization applications
The Ultimate in GPU Scalability

Visual Computing Density (Perf / m3)

- Mobile WS
- SLI WS
- VCS
- VCS Node
- VCS Cluster
- Proprietary Graphics Systems
Mercury Computers – Oil & Gas
Quadro FX used for data and graphics processing
Tesla

CUDA for
HPC solutions
## Double Precision Floating Point

<table>
<thead>
<tr>
<th>Feature</th>
<th>NVIDIA Tesla T10</th>
<th>x86 (SSE4)</th>
<th>Cell SPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>IEEE 754</td>
<td>IEEE 754</td>
<td>IEEE 754</td>
</tr>
<tr>
<td>Rounding modes for FADD and FMUL</td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
<td>All 4 IEEE, round to nearest, zero, inf, -inf</td>
</tr>
<tr>
<td>Denormal handling</td>
<td>Full speed</td>
<td>Supported, costs 1000's of cycles</td>
<td>Supported only for results, not input operands (input denormals flushed-to-zero)</td>
</tr>
<tr>
<td>NaN support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Overflow and Infinity support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Flags</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FMA</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Square root</td>
<td>Software with low-latency FMA-based convergence</td>
<td>Hardware</td>
<td>Software only</td>
</tr>
<tr>
<td>Division</td>
<td>Software with low-latency FMA-based convergence</td>
<td>Hardware</td>
<td>Software only</td>
</tr>
<tr>
<td>Reciprocal estimate accuracy</td>
<td>24 bit</td>
<td>12 bit</td>
<td>12 bit + step</td>
</tr>
<tr>
<td>Reciprocal sqrt estimate accuracy</td>
<td>23 bit</td>
<td>12 bit</td>
<td>12 bit + step</td>
</tr>
<tr>
<td>log2(x) and 2^x estimates accuracy</td>
<td>23 bit</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Single Precision BLAS: CPU vs GPU

BLAS (SGEMM) on CUDA

CUBLAS: CUDA 2.0b2, Tesla C1060 (10-series GPU)
ATLAS 3.81 on Dual 2.8GHz Opteron Dual-Core

Matrix Size

GFLOPS

CUDA
ATLAS 1 Thread
ATLAS 4 Threads
Double Precision BLAS: CPU vs GPU

**BLAS (DGEMM) on CUDA**

- CUBLAS CUDA 2.0b2 on Tesla C1060 (10-series)
- ATLAS 3.81 on Intel Xeon E5440 Quad-core, 2.83 GHz

**Graph Details:**
- X-axis: Matrix Size
- Y-axis: GFLOPS

---

CUBLAS performance on CUDA 2.0b2 with Tesla C1060 (10-series) is highlighted.
# Tesla C1060 Computing Card

<table>
<thead>
<tr>
<th>Specification</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>1 x Tesla T10</td>
</tr>
<tr>
<td>Number of cores</td>
<td>240</td>
</tr>
<tr>
<td>Core Clock</td>
<td>1.29 GHz</td>
</tr>
<tr>
<td>On-board memory</td>
<td>4.0 GB</td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td>102 GB/sec peak</td>
</tr>
<tr>
<td>Memory I/O</td>
<td>512-bit, 800MHz GDDR3</td>
</tr>
<tr>
<td>Form factor</td>
<td>Full ATX: 4.736” x 10.5”</td>
</tr>
<tr>
<td></td>
<td>Dual slot wide</td>
</tr>
<tr>
<td>System I/O</td>
<td>PCIe x16 Gen2</td>
</tr>
<tr>
<td>Power</td>
<td>200 W maximum</td>
</tr>
<tr>
<td></td>
<td>160 W typical</td>
</tr>
<tr>
<td></td>
<td>(5.83 GFlops/Watt)</td>
</tr>
<tr>
<td></td>
<td>25 W idle</td>
</tr>
</tbody>
</table>
Impact of 4GB Memory on Performance

- 4GB of memory is critical for best CUDA performance
- Enables processing on larger data sets
  - Solve larger problems
- Double precision applications require more memory

![Graph showing the impact of 4GB memory on Reverse Time Migration (Seismic Processing)]

- Speedup for Tesla C870, 1.5 GB
- Speedup for Tesla C1060, 1.5 GB
- Speedup for Tesla C1060, 4 GB

**Impact of 4 GB memory**
Globally, researchers are building GPU-based workstations.

- **3 GPUs**
  - Yonsei University, Korea

- **3 GPUs**
  - University of Illinois

- **2 GPUs**
  - University of Cambridge, UK

- **16 GPUs**
  - The Visual Neuroscience Group, University of Antwerp, Belgium
Cluster vs Workstation: Trade-offs

Cluster

- Need server room; shared resource
- Energy hungry: to power up & cool
- Expensive to build/maintain; need IT dept

Workstation

- Dedicated resource for each person
- No system management/IT required
- Built for the office: cool, quiet, & compact

Performance Gap
The Tesla Visual Supercomputer

Return of the Scientific Workstation

- 4 TeraFlops Workstation
  - 4 CUDA GPUs
  - 960 cores
  - 16 GB fast GPU memory

Specs:
- Quad-core CPU (1P or 2P)
- 16 GB System memory
- 4 Tesla/Quadro GPUs

Optimized for scientific computing

The power of a cluster in a workstation
100x more affordable

Supercomputing Cluster

100x Faster

Tesla Personal Supercomputer

Standard Workstation

$100K - $1M

< $10K

Performance

Accessibility
Personal Super Computers

Transtec
Germany

Comptronic
Germany

Fluidyna
Germany

CAD2
UK

Carri
France

Sprinx
Czech

Axxiv
Switzerland

NEXT
Italy

Exon
Italy

Armari
UK

Concordia Graphics
Italy

E4
Italy

Azken Muga
Spain

Viglen
UK
La puissance d'un supercalculateur à portée de main

Votre activité requiert des calculs mathématiques complexes et exigeants ?

Vous avez besoin de plus de puissance et de rapidité ?

Vous recherchez une solution dédiée pour développer vos propres codes de calcul ?

4340€ HT seulement !

Station HP xw6600 et carte NVIDIA® Tesla™ C1060

Offre spéciale de fin d'année

Un supercalculateur dédié à vos calculs sur votre bureau

Bénéficiez d'une solution de supercalc parallèle, et intégrez dans une seule station de travail les performances traditionnellement réservées aux clusters d'entreprise :

- Puissance brute : 1 Teraflop* 240 coeurs
- 4 GB de mémoire

Vous bénéficiez ainsi d'une ressource de bureau dédiée bien plus rapide et écoénergétique qu'un cluster partagé dans un centre de calcul.

4340€ HT !

Ecran 24" Performance

HP LP2475W en option pour 420€ HT.

Spécifications techniques.

Station de travail HP xw6600 :
- 10560 W 80 Energy Efficient Linux
- Installer Kit Software. 2 x Intel Xeon 5450 3.60 GHz QC. 8GB de mémoire.
- Carte graphique NVIDIA Quadro NVS 290. 2 x 146GB SAS 3Gb/s 15K tours.
- DVD / RW SuperMulti SATA. PCI Express 16x.

Carte NVIDIA Tesla C1060 avec processeur Tesla T10P
- massivement parallèle et multi-cœurs, couplée au standard de programmation CUDA C afin de simplifier la programmation multicœur.

Applications supportées

Ref : 74032240

* Possibilité d'ajouter une deuxième carte pour atteindre 1,8 Teraflops.
## Tesla S1070 1U System

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>4 x Tesla T10</td>
</tr>
<tr>
<td>Number of cores</td>
<td>960</td>
</tr>
<tr>
<td>Core Clock</td>
<td>1.5 GHz</td>
</tr>
<tr>
<td>Performance</td>
<td>4 Teraflops</td>
</tr>
<tr>
<td>Total system memory</td>
<td>16.0 GB (4.0 GB per T10P)</td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td>408 GB/sec peak (102 GB/sec per T10P)</td>
</tr>
<tr>
<td>Memory I/O</td>
<td>2048-bit 800MHz GDDR3 (512-bit per T10P)</td>
</tr>
<tr>
<td>Form factor</td>
<td>1U (EIA 19” rack)</td>
</tr>
<tr>
<td>System I/O</td>
<td>2 PCIe x16 Gen2</td>
</tr>
<tr>
<td>Typical power</td>
<td>700 W</td>
</tr>
</tbody>
</table>

Connection to host system(s) using two PCIe interface cards.
NVIDIA Tesla S1070 SKUs

**Tesla S1070-400**
- 1.296 GHz GPU clock
- Peak performance
  - 32-bit: 3.73 TF (933 GF per GPU)
  - 64-bit: 310 GF (77.7 GF per GPU)

**Tesla S1070-500**
- 1.44 GHz GPU clock
- Limited availability
- Peak performance
  - 32-bit: 4.14 TF (1.03 TF per GPU)
  - 64-bit: 345 GF (86.4 GF per GPU)
Tesla S1070: 2U Sample Configuration

Server

1U

Tesla S1070

1U

PCle Host Interface Cards

PCle Gen2 Cables

Two PCIe Gen2 Cables (50 cm or 2 m length)

Two PCle Gen2 Host Interface Cards
Tesla S1070: 3U Sample Configuration

Server

Tesla S1070

Server

1U

1U

1U

PCIe Host Interface Card

PCIe Gen2 Cables

Two PCIe Gen2 Cables (50 cm or 2 m length)

Two PCIe Gen2 Host Interface Cards
Tesla S1070 OEM Partners

- HP
- Dell
- SUN
- SGI
- Bull
- Lenovo
- IBM
- FSC
- Supermicro
Scalable Professional Development Platforms

Laptop / Desktop
- Single User
- Discover GPU Computing
  - Easy entry path but limited performance and memory size
  - Limited dataset size
- 1 to 3K €

Supercomputing PC
- Multiple Tesla C1060
- Single User
- Development & Prototyping
  - Multi-GPU performance scaling
  - 1TFlops and 4GB per GPU
  - Larger dataset
- 3 to 8K €

Hybrid Cluster
- Tesla S1070
- Multiple Users
- Dev., Prototyping & Production
  - 4 TFlops per 1U Tesla
  - 16GB per 1U Tesla
  - Up to TB datasets
- 8K € per 2U (CPU+GPU)

CUDA Compatible
The new Bull NovaScale supercomputer consists of a cluster of 1,068 Intel Nehalem nodes, delivering some 103 TFlops, and 192 NVIDIA Tesla GPU nodes, providing additional power of up to 192 TFlops.

48 Tesla S1070 1U servers
= 192 GPUs
= 768GB

http://www.cea.fr/english_portal/news_list/bull_novascale_supercomputer_genci_and_the_cea
CEA Hybrid Cluster

Over 295 TFlops
#1 in Europe

1,000 TB HDD
LUSTRE file system

25TB RAM

3 x 42U rack for GPU
192 TFlops Peak

17 x 42U Bull NovaScale
103 TFlops Peak

42U rack with 16 pcs Tesla S1070

42U rack with 60 pcs Intel Nehalem CPU. 3GB RAM/core

HDD, network, service nodes

Infiniband DDR

Innovative GPU Platform
Allow specific applications to use multiple CPU-GPU cores and the entire system RAM

SMP Production Platform

- 3GB RAM per CPU core
- For all industrial and research applications

Open Source system software
Some Tesla S1070 Customers

- Tokyo Technical Institute  170 pcs S1070, #27 in TOP 500 Nov08
- Max Plank Institute
- Univ. Francfort, Cardiff, Reims…
- CEA CCRT
- CINES
- EADS
- TOTAL
- BNPParibas
Applications
Accelerating Time to Discovery

- Computational Chemistry: 4.6 Days (27 Minutes), 2.7 Days (30 Minutes), 8 Hours (13 Minutes), 3 Hours
- Neurological Modeling: 4.6 Days (27 Minutes), 2.7 Days (30 Minutes), 8 Hours (13 Minutes), 3 Hours
- Cell Phone RF Simulation: 4.6 Days (27 Minutes), 2.7 Days (30 Minutes), 8 Hours (13 Minutes), 3 Hours
- 3D Breast Ultrasound: 4.6 Days (27 Minutes), 2.7 Days (30 Minutes), 8 Hours (13 Minutes), 3 Hours
Peak Performance Improvement

Dip Azimuth - Structural Seismic Attribute

- Intel Xeon (3.0 GHz) x1
- Intel Xeon Quad Core (3.0 GHz) x2.3
- CUDA Quadro FX 5600
- CUDA Tesla C1060 x43

Mean Performance Improvement

Dip Azimuth - Structural Seismic Attribute

- Intel Xeon (3.0 GHz) x1
- Intel Xeon Quad Core (3.0 GHz) x2.3
- CUDA Quadro FX 5600
- CUDA Tesla C1060 x38
Autodock for Cancer Research

National Cancer Institute reports 12x speedup

Wait for results reduced from 2 hours to 10 minutes

“We can only hope that in the long run, Silicon Informatics' efforts will accelerate the discovery of new drugs to treat a wide range of diseases, from cancer to Alzheimer's, HIV to malaria.”

Dr. Garrett Morris, Scripps, Author of AutoDock
Tesla Revolutionizes Breast Cancer Detection

3D Ultrasound Image Reconstruction

Ultrasound Processing Time

<table>
<thead>
<tr>
<th></th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 CPUs</td>
<td>181</td>
</tr>
<tr>
<td>4 GPUs</td>
<td>16</td>
</tr>
</tbody>
</table>

From 3 hours to 15 minutes

Techniscan Medical Systems
Bioinformatics

Molecular Biology

- Searches for similarities in protein and DNA databases
- Smith-Waterman Algorithm is the most accurate but most time consuming, CUDA enables faster results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Runtime</td>
<td>Speedup</td>
</tr>
<tr>
<td>Seq (1)</td>
<td>1.47s</td>
<td>1.00</td>
</tr>
<tr>
<td>Seq (2)</td>
<td>2.73s</td>
<td>0.54</td>
</tr>
<tr>
<td>Par (1)</td>
<td>21.22s</td>
<td>0.07</td>
</tr>
<tr>
<td>Par (2)(^1)</td>
<td>1.86s</td>
<td>0.70</td>
</tr>
<tr>
<td>CUDA</td>
<td>0.66s</td>
<td>2.23</td>
</tr>
</tbody>
</table>

MPI-HMMR

Open source MPI implementation of the HMMER protein sequence analysis suite

3 GPUs, 117x Speedup

CUDA hmmsearch Performance

Use as many GPUs as your system supports

mpiHMMER

mpiHMMER is a multi-layer performance-enhanced version of Sean Eddy's HMMER.

Performance enhancing optimizations implemented include
- MPI Support
- Parallel I/O Support (New)
- Multi-GPU Support (New)

PIO-HMMER and MPI-HMMER

Currently Supports
- GPP Clusters
- NVIDIA CUDA

Coming Soon
- Integrated Accelerator Platform

Team
John Paul Walters
Rohan Darole
Joseph Landman
Vipin Chaudhary

Use as many GPUs as your system supports

Visit www.mpihmmr.org for more information and for software downloads.

Scalable Informatics
Oil and Gas: Migration Codes

“Based on the benchmarks of the current prototype [128 node GPU cluster], this code should outperform our current 4000-CPU cluster”

Leading global independent energy company

![ADI Wave-equation Performance](image1)

![Kirchhoff Learning Curve](image2)
NVIDIA Tesla computing solutions enable the necessary translational energy efficient parallel computing power. With NVIDIA’s supercomputer and based on the revolutionary NVIDIA CUDA parallel computing architecture, Tesla scales to solve the world’s most important computing challenges—more quickly and accurately.

To learn more about Tesla computing solutions, visit [http://www.nvidia.com/tesla](http://www.nvidia.com/tesla).

To learn more about the CUDA parallel computing architecture, visit [http://www.nvidia.com/cuda](http://www.nvidia.com/cuda).
For more information
Jean-Christophe Baratault
jbaratault@nvidia.com
www.nvidia.com/Tesla
Backup Slides
Definitions

- **Device** = GPU = set of multiprocessors
- **Multiprocessor** = set of processors & shared memory
- **Kernel** = GPU program
- **Grid** = array of thread blocks that execute a kernel
- **Thread block** = group of SIMD threads that execute a kernel and can communicate via shared memory

<table>
<thead>
<tr>
<th>Memory</th>
<th>Location</th>
<th>Cached</th>
<th>Access</th>
<th>Who</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Off-chip</td>
<td>No</td>
<td>Read/write</td>
<td>One thread</td>
</tr>
<tr>
<td>Shared</td>
<td>On-chip</td>
<td>N/A</td>
<td>Read/write</td>
<td>All threads in a block</td>
</tr>
<tr>
<td>Global</td>
<td>Off-chip</td>
<td>No</td>
<td>Read/write</td>
<td>All threads + host</td>
</tr>
<tr>
<td>Constant</td>
<td>Off-chip</td>
<td>Yes</td>
<td>Read</td>
<td>All threads + host</td>
</tr>
<tr>
<td>Texture</td>
<td>Off-chip</td>
<td>Yes</td>
<td>Read</td>
<td>All threads + host</td>
</tr>
</tbody>
</table>
Heterogeneous Memory Model

Host memory

cudaMemcpy()

Device 0 memory

Device 1 memory
Memory Hierarchy

Thread

Per-thread Local Memory

Block

Per-block Shared Memory

Sequential Kernels

Kernel 0

Per-device Global Memory

Kernel 1
Kernel = Many Concurrent Threads

- One kernel is executed at a time on the device
- Many threads execute each kernel
  - Each thread executes the same code…
  - … on different data based on its threadID

CUDA threads might be
- Physical threads
  - As on NVIDIA GPUs
  - GPU thread creation and context switching are essentially free
- Or virtual threads
  - E.g. 1 CPU core might execute multiple CUDA threads

```c
float x = input[threadID];
float y = func(x);
output[threadID] = y;
...```

Threads are grouped into **thread blocks**

Kernel = **grid** of thread blocks

By definition, threads in the same block may **synchronize with barriers**

```c
__syncthreads();
int left = scratch[threadID - 1];
```
Hardware Implementation: A Set of SIMT Multiprocessors

- Each multiprocessor is a set of 32-bit processors with a Single-Instruction Multi-Thread architecture
  - 30 multiprocessors on GT200
  - 8 processors per multiprocessors

- At each clock cycle, a multiprocessor executes the same instruction on a group of threads called a warp
  - The number of threads in a warp is the warp size (= 32 threads on GT200)
  - A half-warp is the first or second half of a warp
Hardware Implementation: Memory Architecture

- The global, constant, and texture spaces are regions of device memory.
- Each multiprocessor has:
  - A set of 32-bit `registers` per processor (16,384 on GT200)
  - **On-chip shared memory** (16KB on GT200)
    - Where the shared memory space resides
  - A read-only **constant cache**
    - To speed up access to the constant memory space
  - A read-only **texture cache**
    - To speed up access to the texture memory space
Hardware Implementation:
Execution Model

- Each multiprocessor processes batches of blocks one batch after the other
  - **Active blocks** = the blocks processed by one multiprocessor in one batch
  - **Active threads** = all the threads from the active blocks
- The multiprocessor’s registers and shared memory are split among the active threads
- Therefore, for a given kernel, the number of active blocks depends on:
  - The number of registers the kernel compiles to
  - How much shared memory the kernel requires
- If there cannot be at least one active block, the kernel fails to launch
Hardware Implementation: Execution Model

- Each active block is split into warps in a well-defined way
- Warps are time-sliced

In other words:
- Threads within a warp are executed *physically* in parallel
- Warps and blocks are executed *logically* in parallel
Scalability Solution

- Programmer uses multi-level data parallel decomposition
  - Decomposes problem into a sequence of steps (Grids)
  - Decomposes Grid into independent parallel Blocks (thread blocks)
  - Decomposes Block into cooperating parallel elements (threads)

- GPU hardware distributes thread blocks to available multiprocessors
  - GPU balances work load across any number of multiprocessors cores
  - Core executes program that computes Block

- Each thread block computes independently of others
  - Enables parallel computing of Blocks of a Grid
  - No communication among Blocks of same Grid
  - Scales one program across any number of parallel cores

- Programmer writes one program for all GPU sizes
- Program does not know how many cores it uses
- Program executes on GPU with any number of cores
Compiling CUDA

1. C CUDA Application
2. NVCC
3. PTX Code
4. PTX to Target Compiler
5. G80, ..., GT200

Virtual

Physical

Target code
Role of Open64

Open64 compiler gives us

- A complete C/C++ compiler framework. Forward looking. We do not need to add infrastructure framework as our hardware arch advances over time.

- A good collection of high level architecture independent optimizations. All GPU code is in the inner loop.

- Compiler infrastructure that interacts well with other related standardized tools.
CUDA Advantages over Legacy GPGPU

- Random access byte-addressable memory
  - Thread can access any memory location
- Unlimited access to memory
  - Thread can read/write as many locations as needed
- Shared memory (per block) and thread synchronization
  - Threads can cooperatively load data into shared memory
  - Any thread can then access any shared memory location
- Low learning curve
  - Just a few extensions to C
  - No knowledge of graphics is required
- No graphics API overhead
GPU Comparison

Nov06  **G80**  128 SP  384-bit mem i/f  PCIe Gen1
Jun08  **GT200**  240 SP  512-bit mem i/f  PCIe Gen2
<table>
<thead>
<tr>
<th>Tesla</th>
<th>C870</th>
<th>C1060</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>G80</td>
<td>T10</td>
</tr>
<tr>
<td>Device memory</td>
<td>1.5GB</td>
<td>4GB</td>
</tr>
<tr>
<td>Multiprocessor</td>
<td>16</td>
<td>30</td>
</tr>
<tr>
<td>Cores</td>
<td>128</td>
<td>240</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Per multiprocessor</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared memory</td>
<td>16KB</td>
<td>16KB</td>
</tr>
<tr>
<td>Cache for constant memory</td>
<td>8KB</td>
<td>8KB</td>
</tr>
<tr>
<td>Cache for texture memory</td>
<td>8KB</td>
<td>8KB</td>
</tr>
<tr>
<td>Active block</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Active warps</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>Active threads</td>
<td>768</td>
<td>1 024</td>
</tr>
<tr>
<td>Registers</td>
<td>8 192</td>
<td>16 384</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Threads per block</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>x, y, z dimension</td>
<td>512, 512, 64</td>
</tr>
<tr>
<td>Grid thread block</td>
<td>65 535</td>
</tr>
<tr>
<td>Warp size</td>
<td>32</td>
</tr>
<tr>
<td>Constant memory</td>
<td>64KB</td>
</tr>
</tbody>
</table>

**GT200 New features**

- Atomic functions operating on 32-bit words in global memory
- Atomic functions operating in shared memory
- Atomic functions operating on 64-bit words in global memory
- Warp vote functions
- Double-precision floating-point numbers
## Selecting a CUDA Platform

<table>
<thead>
<tr>
<th>Feature</th>
<th>Tesla</th>
<th>Quadro</th>
<th>GeForce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stress tested and burned-in with added margin for numerical accuracy</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufactured by NVIDIA with professional grade memory</td>
<td>✓</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>NVIDIA care: 3-year warranty from NVIDIA, enterprise support</td>
<td>✓</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>4 Gigabyte on-board memory for large technical computing data sets</td>
<td>✓</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Single card solution for professional visualization and CUDA computing</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Consumer middle-ware and applications: PhysX, Video, Imaging</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Consumer product life cycle</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Manufactured and guaranteed by NVIDIA graphics add-in card partners</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Product support through NVIDIA graphics add-in card partners</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td><strong>Tesla 8-series</strong></td>
<td></td>
<td><strong>Tesla 10-series</strong></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------</td>
<td>--------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td><strong>C870 card</strong></td>
<td></td>
<td><strong>C1060 card</strong></td>
</tr>
<tr>
<td>Number of Cores</td>
<td>128</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>32-bit FP Performance</td>
<td>0.5 Teraflop</td>
<td></td>
<td>1 Teraflop</td>
</tr>
<tr>
<td>On-board Memory</td>
<td>1.5 GB</td>
<td></td>
<td>4.0 GB</td>
</tr>
<tr>
<td>Memory interface</td>
<td>384-bit GDDR3</td>
<td></td>
<td>512-bit GDDR3</td>
</tr>
<tr>
<td>Memory I/O bandwidth</td>
<td>77 GB/sec</td>
<td></td>
<td>102 GB/sec</td>
</tr>
<tr>
<td>System interface</td>
<td>PCIe x16 Gen1</td>
<td></td>
<td>PCIe x16 Gen2</td>
</tr>
</tbody>
</table>
Intel PCIe bus

**PCle x16 Gen2**
- x16 physical & electrical 5.5GB/s
- x16 physical / x8 electrical 2.7GB/s
- x16 physical / x4 electrical 1.4GB/s

**PCle x16 Gen1**
- x16 physical & electrical 2.5GB/s
- x16 physical / x8 electrical 1.4GB/s
- X16 physical / x4 electrical 700MB/s
<table>
<thead>
<tr>
<th>Feature</th>
<th>NVIDIA Tesla</th>
<th>NVIDIA Quadro</th>
<th>NVIDIA GeForce</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Designed and Mfg by</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Product Engineered By</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>Add In Card maker (AIC)</td>
</tr>
<tr>
<td>Components Selected and Sourced by</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>AIC</td>
</tr>
<tr>
<td>ECO Control</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>AIC</td>
</tr>
<tr>
<td>Quality Testing</td>
<td>Compute and Memory</td>
<td>Professional Graphics</td>
<td>Consumer Graphics</td>
</tr>
<tr>
<td>Form Factors</td>
<td>Card and 1U</td>
<td>Card, Deskside and 1U</td>
<td>Card</td>
</tr>
<tr>
<td>Roadmap</td>
<td>High Performance Computing</td>
<td>Professional Graphics (Open GL &amp; DirectX Applications)</td>
<td>Consumer (Gaming) (DirectX Games)</td>
</tr>
<tr>
<td>Operating Specifications</td>
<td>Corporate Compute Environment</td>
<td>Professional Workstation, Thin Client (passive)</td>
<td>Consumer (Gaming)</td>
</tr>
<tr>
<td>Supported Provided By</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>AIC</td>
</tr>
<tr>
<td>Max Data Readback</td>
<td>3 GB/s (CUDA)</td>
<td>3 GB/s (OGL, DX)</td>
<td>1 GB/s</td>
</tr>
<tr>
<td>Max Frame Buffer/GPU (On Board Memory)</td>
<td>4 GB</td>
<td>4 GB</td>
<td>1 GB</td>
</tr>
<tr>
<td>Lifecycle</td>
<td>36 months</td>
<td>24-36 months</td>
<td>9-12 month</td>
</tr>
<tr>
<td></td>
<td>Managed by NVIDIA</td>
<td>Managed by NVIDIA</td>
<td>Varies by AIC manufacturer</td>
</tr>
</tbody>
</table>
## Tesla – Quadro Positioning

<table>
<thead>
<tr>
<th>High Level Positioning</th>
<th>Optimized for \textit{Computing}</th>
<th>Optimized for \textit{Professional Visualization}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Testing</td>
<td>Compute validation of memories (additional testing for data access)</td>
<td>Testing for graphics image rendering (frame buffer)</td>
</tr>
<tr>
<td>Graphics Capabilities</td>
<td>Standard OpenGL (compatible with mGPU/GeForce)</td>
<td>Quadro OpenGL &amp; Direct X (certified for Pro WS Apps)</td>
</tr>
<tr>
<td>Products</td>
<td>HPC boards &amp; 1U systems</td>
<td>Full graphics product line (mGPU, 2D, 3D, Vertical, Systems)</td>
</tr>
<tr>
<td>Roadmap</td>
<td>Computing &gt; Double Precision (FP64) &gt; ECC &gt; Computing developer program &gt; Tesla cluster promotion</td>
<td>Professional Visualization &gt; More shader, geometry, fill rate &gt; Increases in image quality &gt; Pro App Scaling &gt; Quadro specific features &gt; Virtualization &amp; Remoting</td>
</tr>
</tbody>
</table>
Le plus important des supercalculateurs à base de GPU est en France

Un supercalculateur à base de processeurs graphiques a été commandé à Bull par l'administration française. C'est la plus puissante architecture à base de processeurs graphiques connue à ce jour et elle est en France.

Tanguy ANDRILLON

Le CEA et le CNRS préparent un supercalculateur Bull

Le CEA et le CNRS préparent un supercalculateur Bull. Le centre, qui cumule 550 teraflops, a débuté les travaux de conception du supercalculateur. Il est prévu d'avoir un supercalculateur Bull d'une puissance de 1000 teraflops en 2009.

Le CEA et le CNRS préparent un supercalculateur Bull

Le CEA et le CNRS préparent un supercalculateur Bull. Le centre, qui cumule 550 teraflops, a débuté les travaux de conception du supercalculateur. Il est prévu d'avoir un supercalculateur Bull d'une puissance de 1000 teraflops en 2009.


Le CEA et le CNRS réunissent leurs deux supercalculateurs de l'Écosse au sein du Centre National Jacques Louis Lions de Calcul Haute Performance de l'Écosse et en augmentant la puissance. L'ensemble d'au moins 1.500 Tflops (200 au CNRS et 200 au CEA) dès l'an prochain aux chercheurs français pour étudier en particulier le domaine de la simulation nucléaire. C'est le grand calculateur, le centre de calcul, qui est candidat pour accueillir un nœud du réseau européen Prace (Partnership for Advanced Computing in Europe) destiné aux machines de capacité parallèle. A la clé une survivance à hauteur de 10% pour l'infrastructure. Le projet d'un montant de 10M€ est financé par le Geni (Grand équipement National de Calcul Intensif), société civile créée pour coordonner les politiques françaises d'équipement en supercalculateurs.