



GPU Computing Applications

ADASS 2011

Dr. Gernot Ziegler
Developer Technology (HPC Compute)
NVIDIA UK



Agenda

- NVIDIA and history of GPU-based HPC computing
- Programming GPUs, an Overview
- A Couple of Examples
- GPU as a Signal Processor / Textures
- Further Material

VISUALIZATION

QUADRO™



PARALLEL COMPUTING

TESLA™



PERSONAL COMPUTING

GeForce™, TEGRA™



NVIDIA and HPC Evolution of GPUs

- **NVIDIA:** Based in Santa Clara, CA | ~\$4B revenue | ~5,500 employees
- **Founded in 1999** with primary business in semiconductor industry
 - **Products for graphics in workstations, notebooks, mobile devices, etc.**
 - **Began R&D of GPUs for HPC in 2004, released first Tesla and CUDA C in 2007**
- **Development of GPUs as a co-processing accelerator for x86 CPUs**

HPC Evolution of GPUs

- 2004: Began strategic investments in GPU as HPC co-processor
- 2006: G80 first GPU with built-in compute features, 128 cores; CUDA SDK Beta
- **2007:** Tesla 8-series based on G80, 128 cores - CUDA 1.0, 1.1
- **2008:** Tesla 10-series based on GT 200, 240 cores - CUDA 2.0, 2.3
- **2009:** Tesla 20-series, code named “Fermi” up to 512 cores - CUDA SDK 3.0, 3.2



3 Years With
3 Generations

Tesla Data Center & Workstation GPU Solutions



Tesla M-series GPUs
M2090 | M2070 | M2050

**Integrated CPU-GPU
Servers & Blades**



Tesla C-series GPUs
C2070 | C2050

**Workstations
2 to 4 Tesla GPUs**

		M2090	M2070	M2050
Cores		512	448	448
Memory		6 GB	6 GB	3 GB
Memory bandwidth (ECC off)		177.6 GB/s	150 GB/s	148.8 GB/s
Peak Perf Gflops	Single Precision	1331	1030	1030
	Double Precision	665	515	515

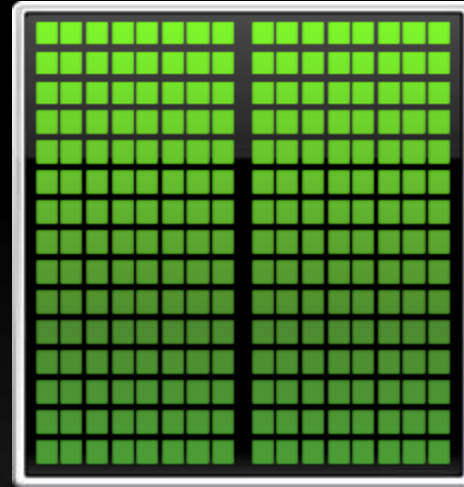
C2070	C2050
448	448
6 GB	3 GB
148.8 GB/s	148.8 GB/s
1030	1030
515	515

GPGPU Revolutionizes Computing

Latency Processor + Throughput processor



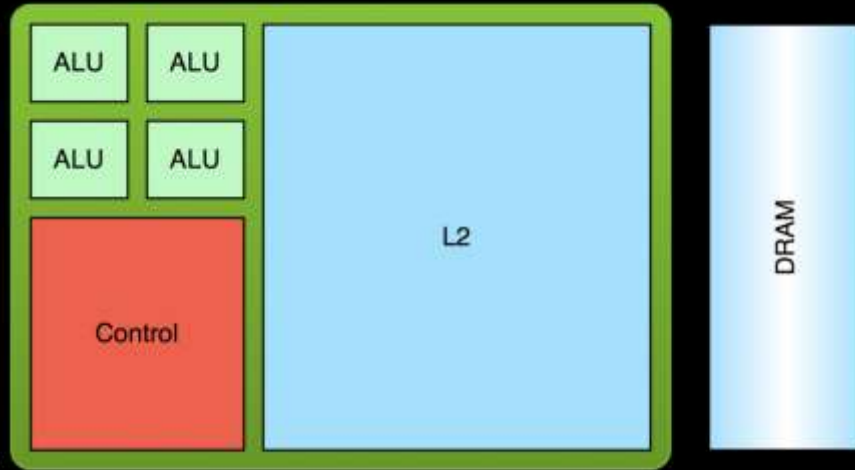
+



CPU

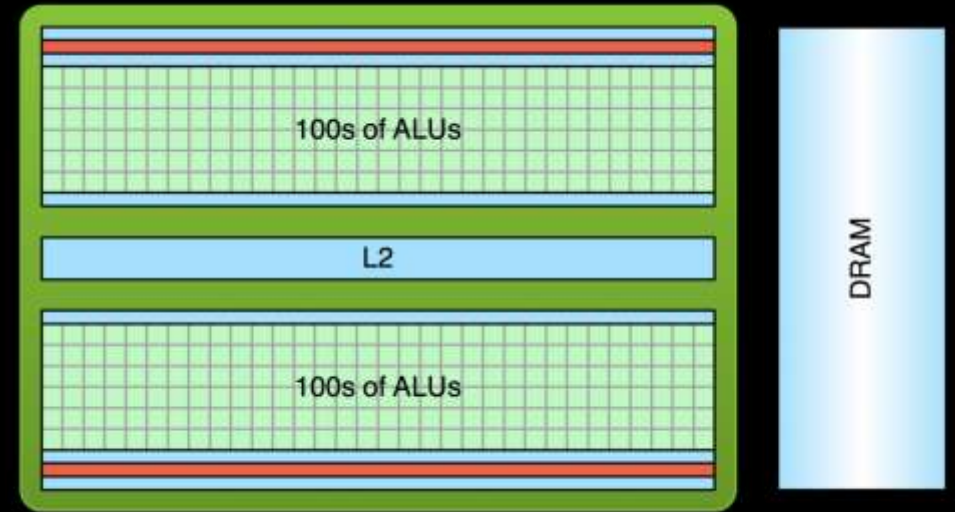
GPU

Low Latency or High Throughput?



CPU

- Optimized for low-latency access to cached data sets
- Control logic for out-of-order and speculative execution

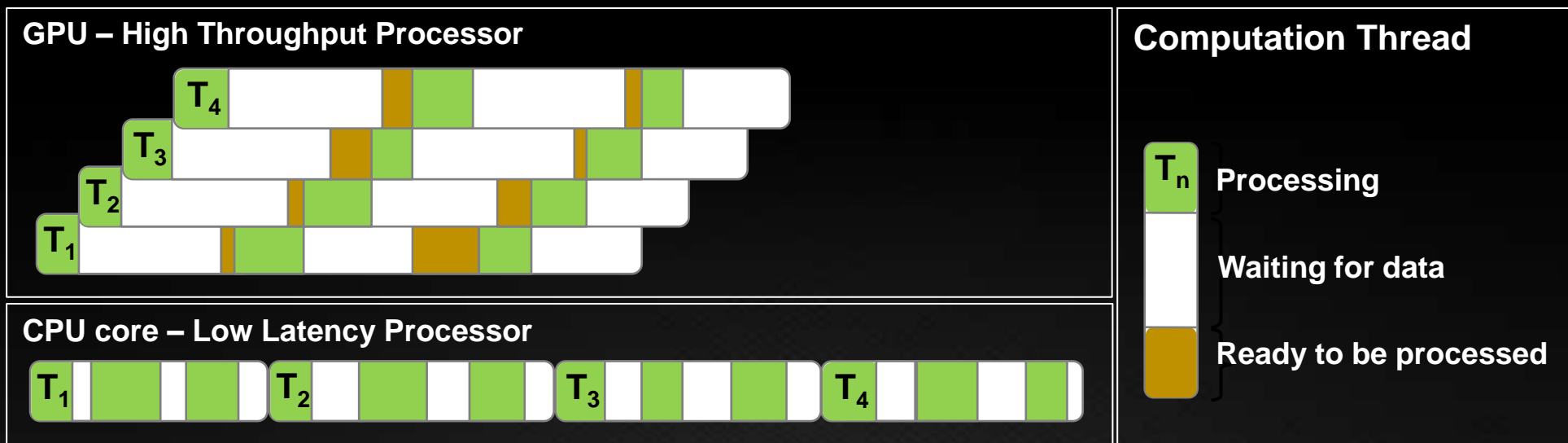


GPU

- Optimized for data-parallel, throughput computation
- Architecture tolerant of memory latency
- More transistors dedicated to computation

Low Latency or High Throughput?

- CPU architecture must minimize latency within each thread
- GPU architecture hides latency with computation (data-parallelism, to 30k threads!)



GPUs are Disruptive

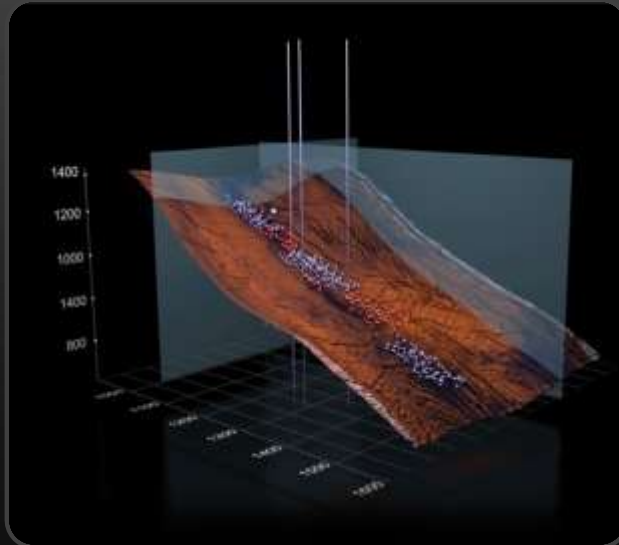
Speed

Days to minutes
Minutes to seconds



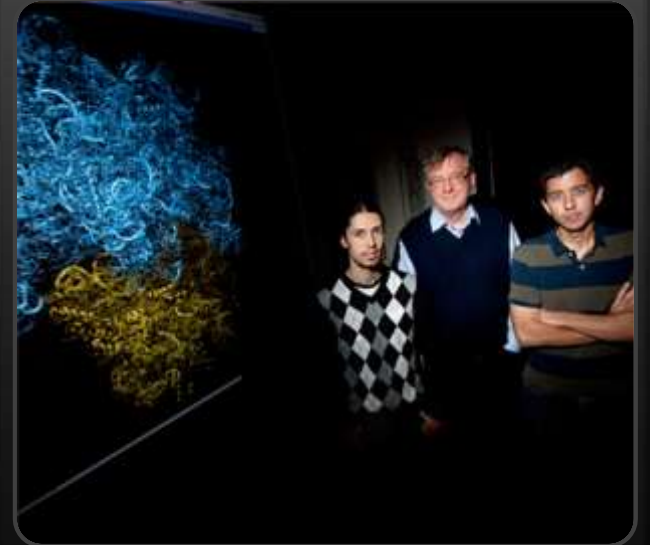
Throughput

Simulate more scenarios



Insight

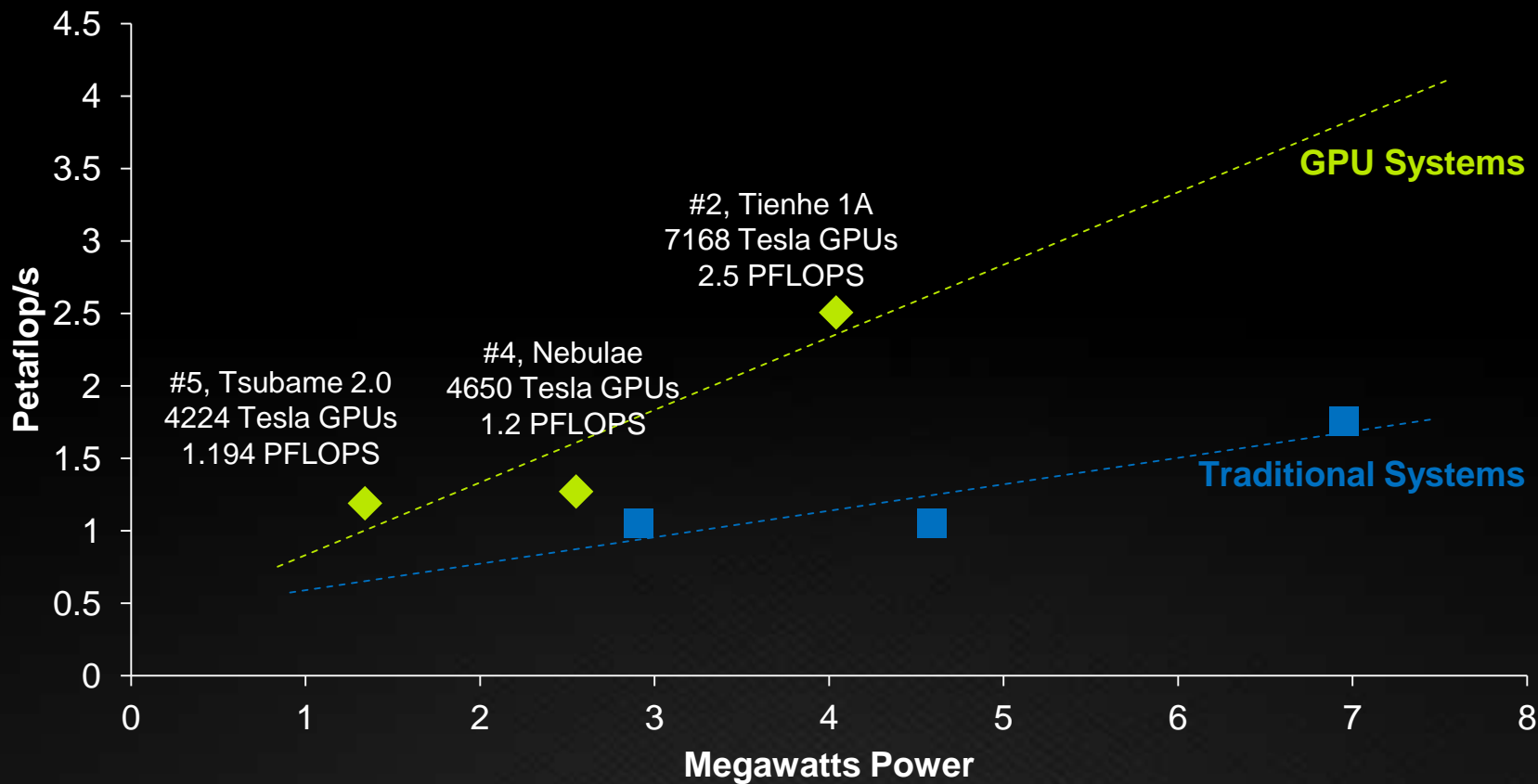
Real time analysis



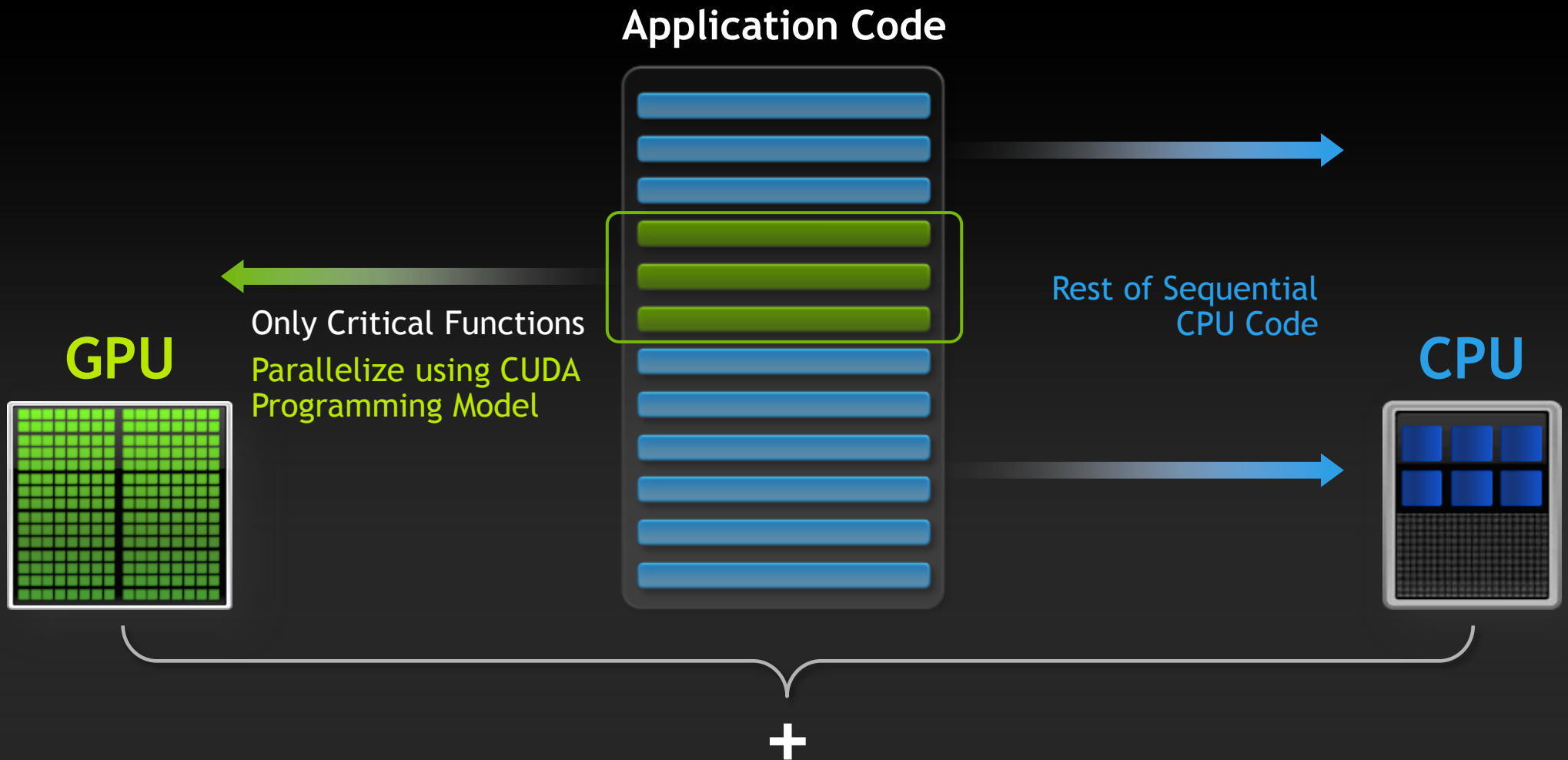
More Performance, Less Power

Performance per Megawatts Power

Fastest Top500 Systems



GPU as Coprocessor



CUDA: Easy to Use Parallel Programming Model

GPU Computing Applications

Libraries and Middleware

cuFFT cuBLAS cuRAND cuSPARSE	LAPACK CULA MAGMA	NPP cuDPP Thrust	VSIPL SVM OpenCurrent	PhysX Video OptiX Ray tracing	iray Rendering RealityServer	MATLAB Mathematica
---------------------------------------	-------------------------	------------------------	-----------------------------	--	------------------------------------	-----------------------

C++

C

Fortran

Java
Python
Wrappers

Direct
Compute

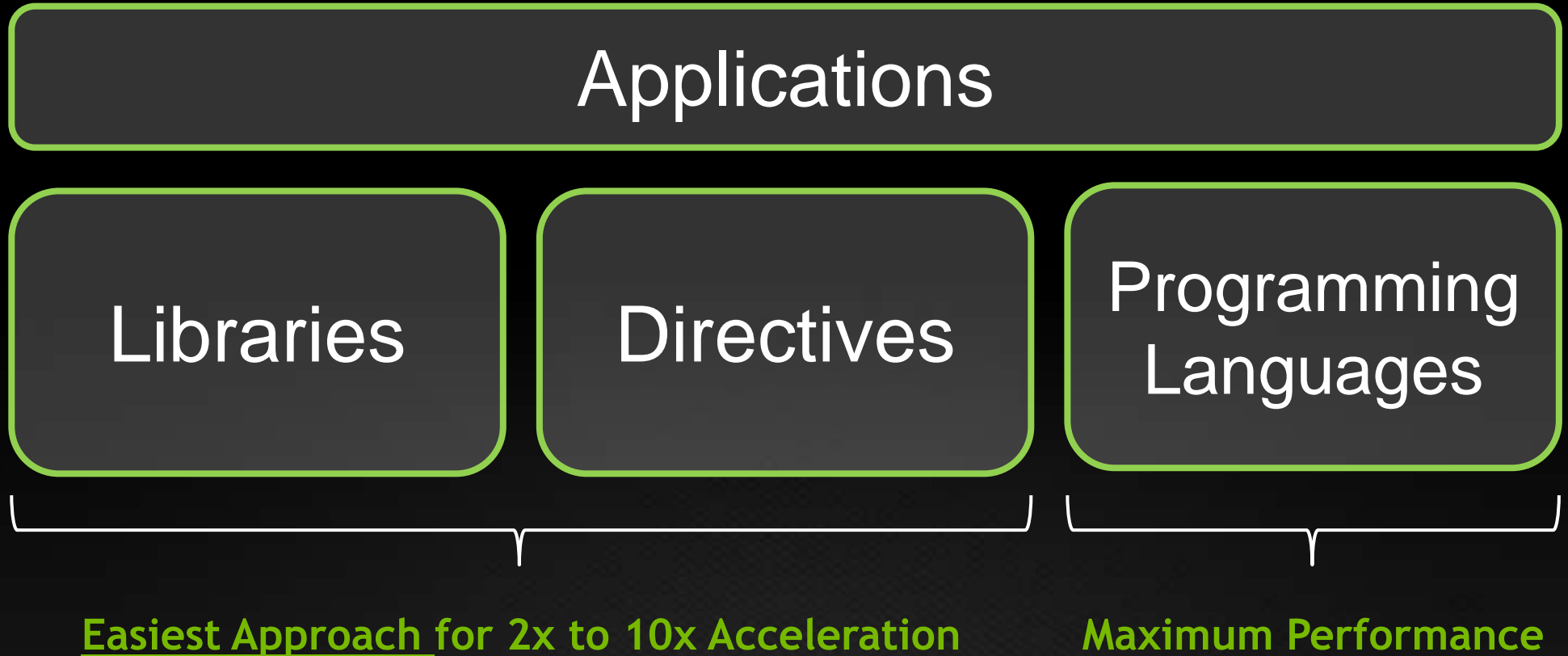
OpenCL™



NVIDIA GPU

CUDA Parallel Computing Architecture

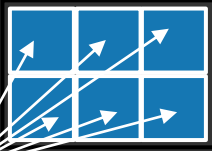
3 Ways to Accelerate Your Apps



Directives: Add One Line of Code

OpenMP

CPU



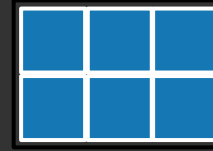
```
main() {
  double pi = 0.0; long i;

  #pragma omp parallel for reduction(+:pi)
  for (i=0; i<N; i++)
  {
    double t = (double)((i+0.05)/N);
    pi += 4.0/(1.0+t*t);
  }

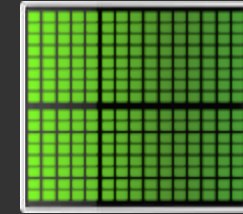
  printf("pi = %f\n", pi/N);
}
```

GPU Directives*

CPU



GPU



```
main() {
  double pi = 0.0; long i;

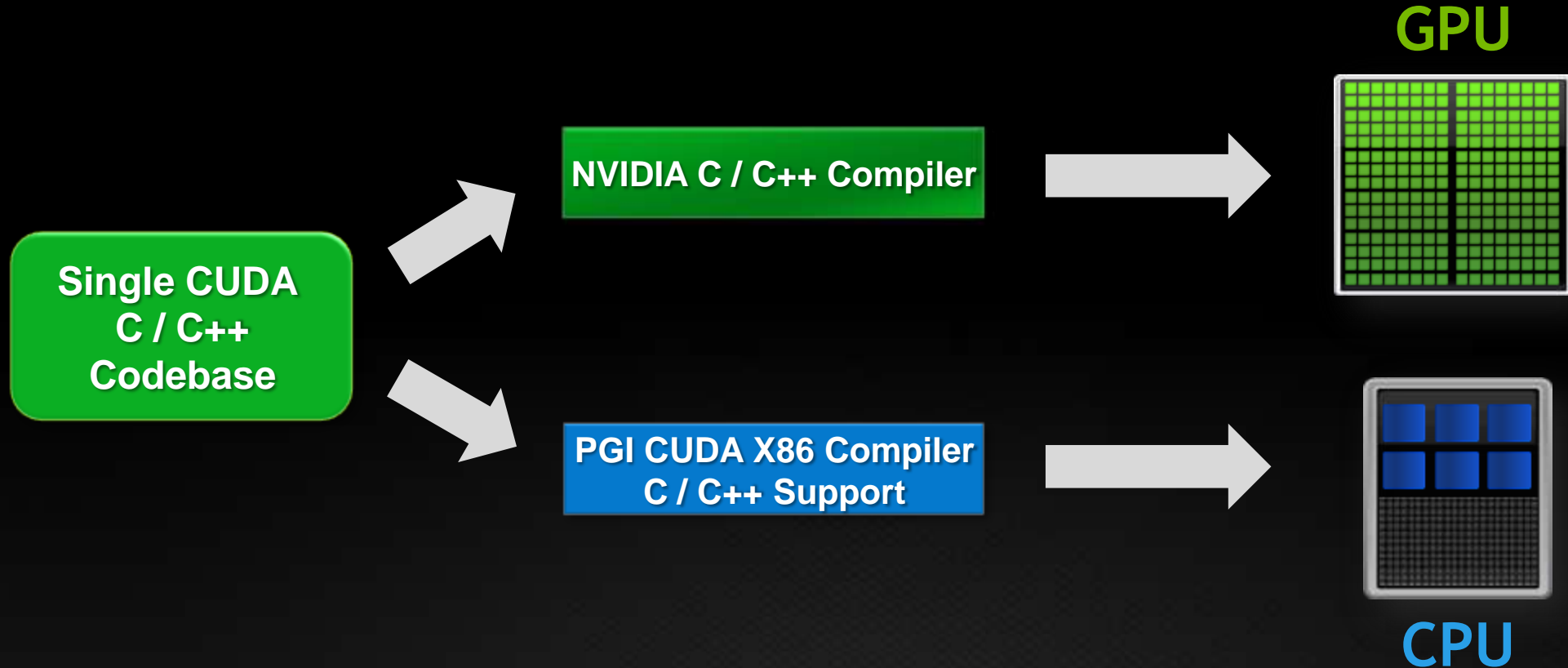
  #pragma omp acc_region_loop
  #pragma omp parallel for reduction(+:pi)
  for (i=0; i<N; i++)
  {
    double t = (double)((i+0.05)/N);
    pi += 4.0/(1.0+t*t);
  }

  #pragma omp end acc_region_loop
  printf("pi = %f\n", pi/N);
}
```

*Directives from Cray

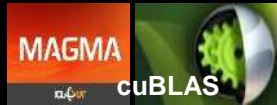
PGI CUDA x86

CUDA Now Available for CPUs and GPUs



GPU Libraries: Simply Use and Accelerate

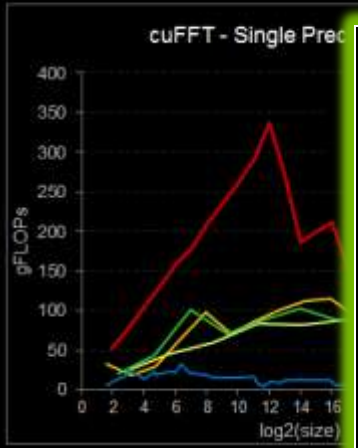
CUDA tools



Dense Linear Algebra

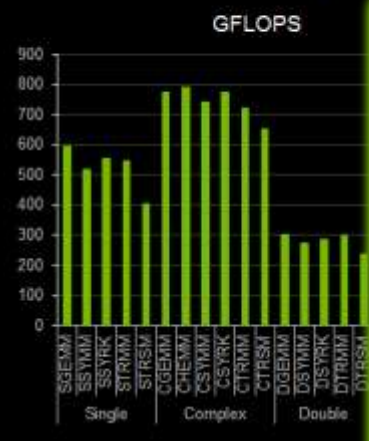
FFTs up to 10x Faster than MKL

1D used in audio processing and as a foundation for 2D and 3D FFTs



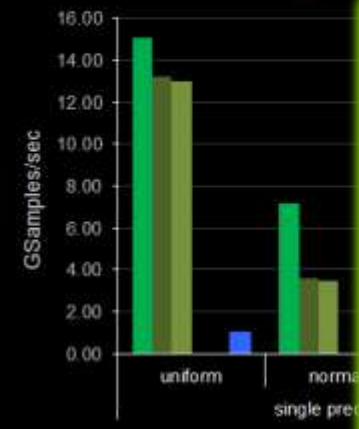
cuBLAS Level 3 Performance

Up to ~800GFLOPS and ~17x speedup over MKL



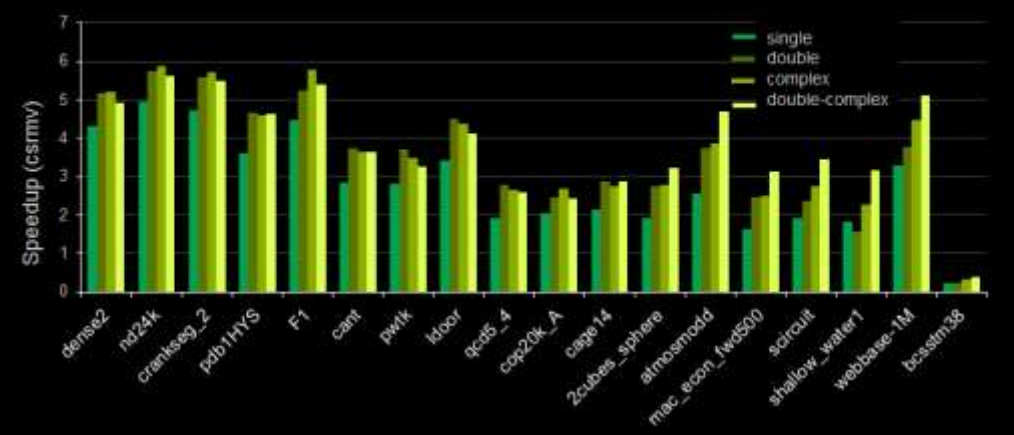
cuRAND Performance

cuRAND 64-bit Scrambled Sobol' 8x faster than MKL 32-bit plain Sobol'



cuSPARSE is up to 6x Faster than MKL

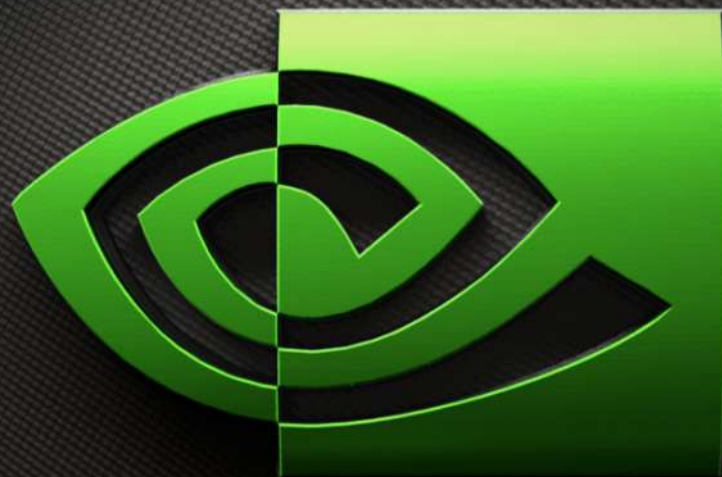
Sparse Matrix x Dense Vector



Parallel Algorithms

QUADA

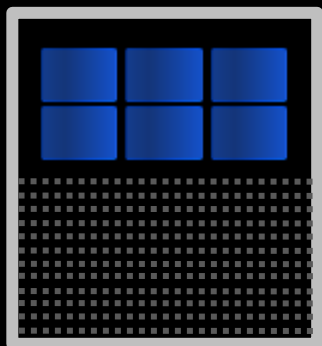
Lattice QCD



nVIDIA

GPU Applications

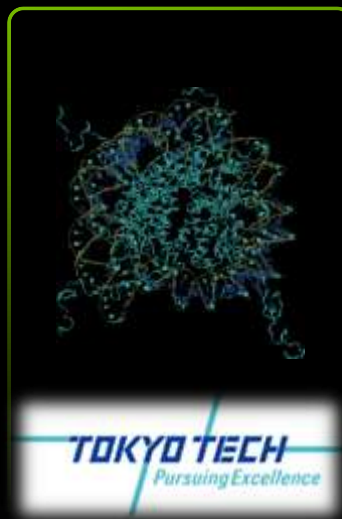
AMBER 10x Faster (@ less power)



16 CPUs

0.36 ns/day

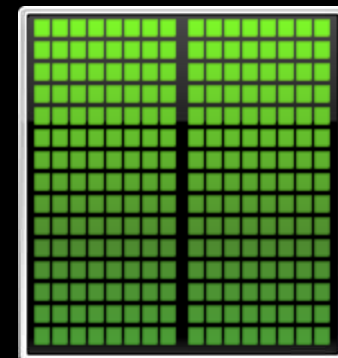
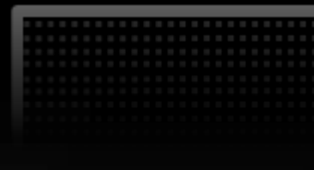
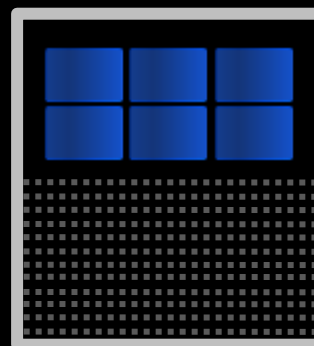
7,737 kJ



Processors

10x Faster

4x Energy Savings



16 CPUs + 24 GPUs

3.44 ns/day

1,142 kJ

World's Fastest Molecular Dynamics Simulation

Sustained Performance of 1.87 Petaflops/s

Institute of Process Engineering (IPE)

Chinese Academy of Sciences (CAS)

Simulation for Crystalline Silicon

Used for Photovoltaic cells & Semiconductors



**Used all 7168 Tesla GPUs on
Tianhe-1A GPU Supercomputer**





TESLA enables medical scans to be performed faster, more accurately and at much lower doses of radiation. Up to 28,000 Americans each year develop cancer due to radiation from CT scans. A UCSD sophisticated algorithm for image reconstruction uses TESLA to reduce the CT radiation required by up to 70 times.



20+ Oil & Gas Companies with CUDA Projects

Successful Customers



Oil & Gas ISVs



GPU vs CPU Improvements



Performance / Watt

18x - 27x

12x - 17x

Performance / Space

20x - 31x

15x - 20x

Performance / Cost

15x - 20x

10x - 12x

JPMorgan Chase

Innovation in Risk Management Technology Award

Project: *GPU Deployment for Risk Computation*

Risk calculations on long-running exotics instruments, for example, are now performed up to 30 times faster.

JPMorgan's equity derivatives business alone had found itself spending more than \$30m annually on servers for market, credit, scenario and regulatory risk computation.

In response, the US bank launched an ambitious plan to reduce the computational costs of risk calculation by three-quarters over a three-year timeframe, while also enabling more accurate and frequent risk calculations.

“This transformation has not only reduced the total cost of ownership of JPMorgan’s risk-management platform,” says Alain Gaudeau, CTO of global equity derivatives at JPMorgan. “It has also created a quantum leap forward in the speed of risk calculations and the speed we can service client requests opportunities.”

WestLB

VI. CONCLUSION

We have replaced significant parts of our computing grid for structured equity products by hybrid GPU/CPU pricing engines. On average a hybrid pricing engine with two Tesla C1060 and two quad core CPUs has replaced 140 CPU cores of the conventional computing grid. For the local volatility model, the speedup is so massive that at least 4 CPU cores are necessary to feed the GPU with input data and process the results. Altogether this leads to a significant reduction of costs for computational resources on the one hand and at the same time allows for a more precise analysis of intrinsic risks of complex structured equity products.

- Bernemann, A.; Schreyer, R.; Spanderen, K.; , "Pricing structured equity products on GPUs," *High Performance Computational Finance (WHPCF)*, 2010 *IEEE Workshop on* , vol., no., pp.1-7, 14-14 Nov. 2010
doi: 10.1109/WHPCF.2010.5671821
URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5671821&isnumber=5671810>

Who's Who of Customers (see also CUDAZone)

Higher Ed



Chinese Academy of Sciences

Georgia Tech



HARVARD School of Engineering and Applied Sciences



Government



Air Force Research Laboratory



Naval Research Laboratory

Life Sciences

Boston Scientific



Max Planck Institute



Mass General Hospital

Oil & Gas



PETROBRAS



TOTAL



Others

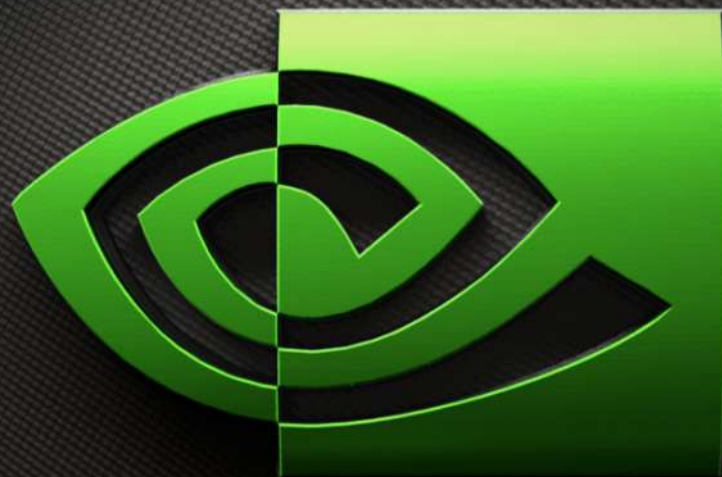


Agilent

Bloomberg

P&G

...and Many More Customers




nVIDIA

Jacket

Jacket™ for MATLAB®: Algorithm Development

AccelerEyes

Technology



MATLAB®

Jacket

JIT Compiler

Run-time system

CUDA Libraries

Applications

Mathematics

Statistics

Signal Processing

Image Processing

Video Processing

Rapid Prototyping

Customers

Government

Bio/Life Science

Academia & Research

Financial Services



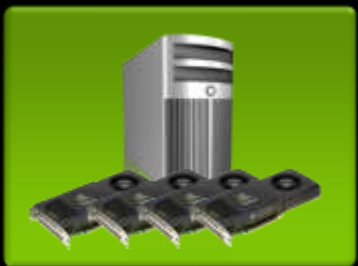
Resources

<http://www.accelereyes.com>

Whitepapers Case Studies Documentation User Forums Application Examples

Platforms

Tesla Personal Supercomputer



Tesla GPU Clusters

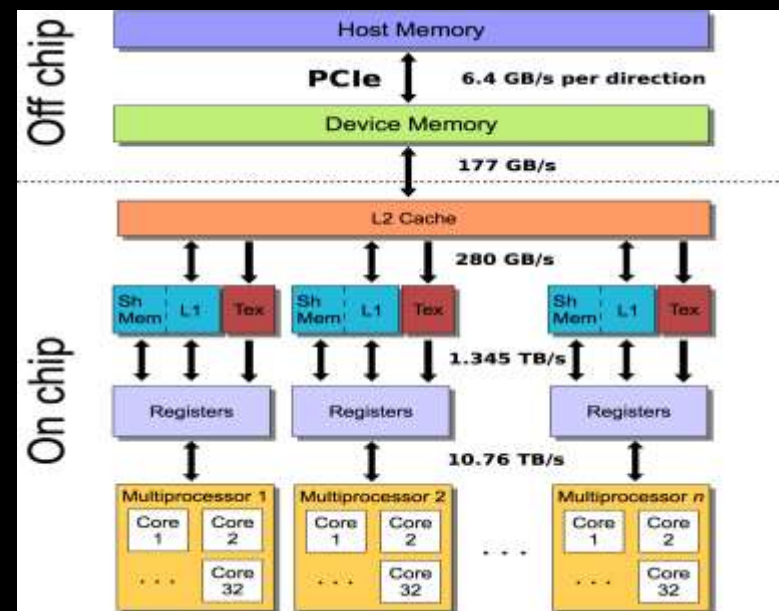


Cross-Correlation on GPUs

Clark, La Plante and Greenhill
see also talk by Greenhill



- **Cross-Correlation of multiple signal sources**
- **The X-engine is "just" linear algebra**
 - $O(N^2)$ compute, $O(N)$ memory traffic (load)
- **GPUs well very suited to this task**
 - **Apply multi-level memory tiling to memory hierarchy**
 - **Software-managed cache gives explicit control**
 - **Maximize arithmetic intensity (flop:byte ratio)**
- **Achieves 79% peak performance on Fermi**
 - **> 1 Tflops on M2090**
- **Repository: <https://github.com/mikeaclark/xGPU>**
- **Preprint: <http://arxiv.org/abs/1107.4264>**



GPU Tree-code

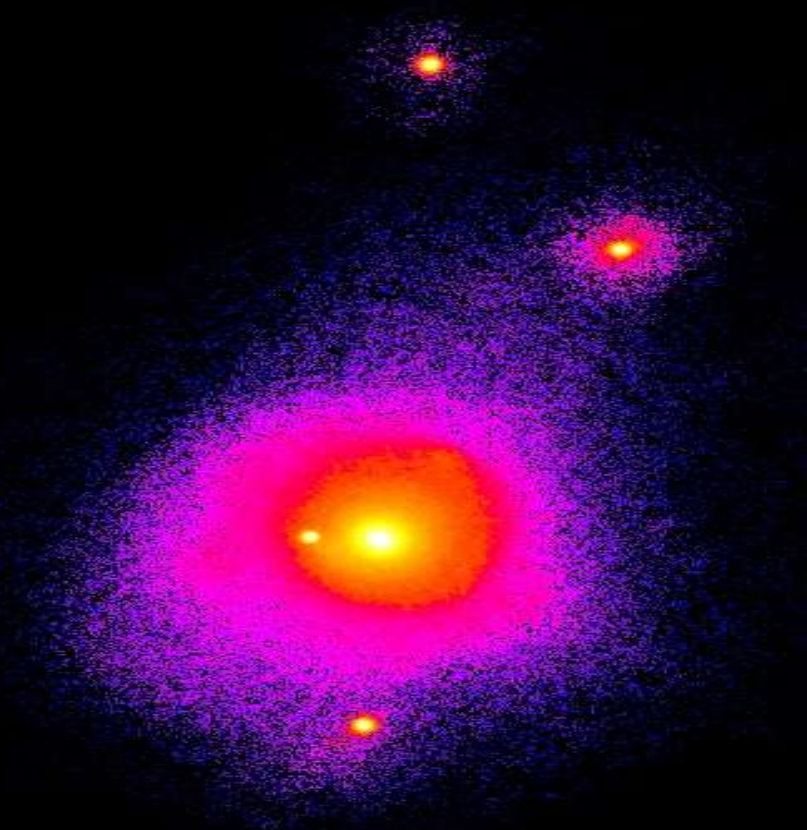
(Leiden University, Netherlands)

- **Bonsai**

- **Multi-GPU, everything runs on the GPU, (CPU only starts the kernels)**
- **Novel octree GPU data algorithm**
- **Current GPU version ~45x faster on a GTX480 than comparable quad-core CPU code**

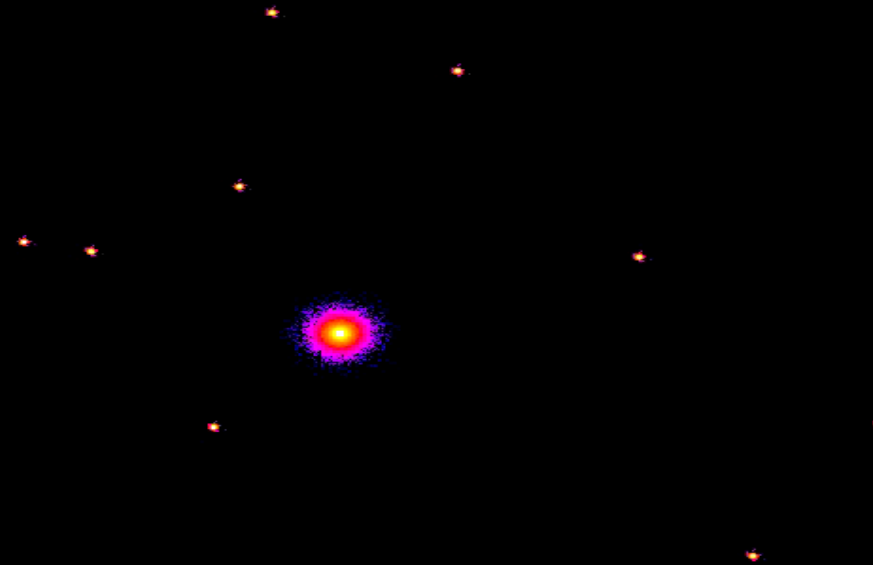
- **Usage**

- **Galaxy merger simulations**
- **Suitable for Large N and many small N simulations**



By: Jeroen Bédorf, Evghenii Gaburov and Simon Portegies Zwart
[1] <http://arxiv.org/abs/1106.1900> and <http://castle.strw.leidenuniv.nl/>

Current frame: 0
CPU load: 0%
Input size: 800x800
Output size: 800x800



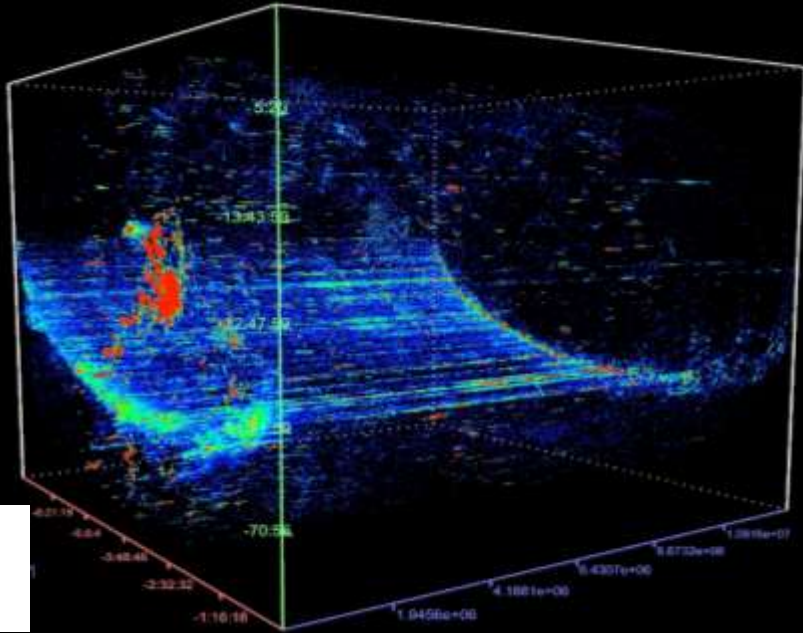
- **Movie: Snapshot of the baryonic matter, colored by density**
 - **22 Million particles (20M dark matter, 2M baryonic)**
 - **Takes ~2 days on 16 GTX480 GPUs**

**Last but not least:
GPU-accelerated astro-research @
SWINBURNE (as presented)**



Real-time Terascale Volume Rendering

A.Hassan, C.Fluke, D.Barnes (Monash)



Real-time volume rendering of HIPASS

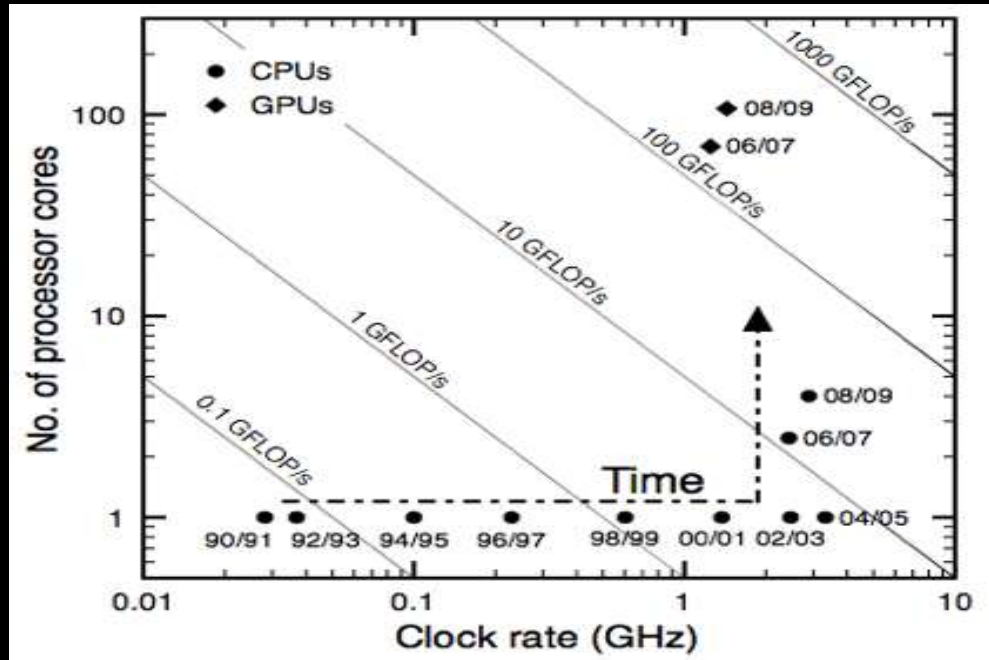
- **Aim:** Enable fully interactive, real-time 3D visualisation of terascale datasets.
- **Method:** Volume rendering via customised message passing and job control over distributed cluster of GPUs.

- Achieved better than 30 fps on 200 GB data file.
- Within reach of TB data from ASKAP radio telescope.



Analysing algorithms for GPUs and beyond

B.Barsdell, D.Barnes (Monash), C.Fluke



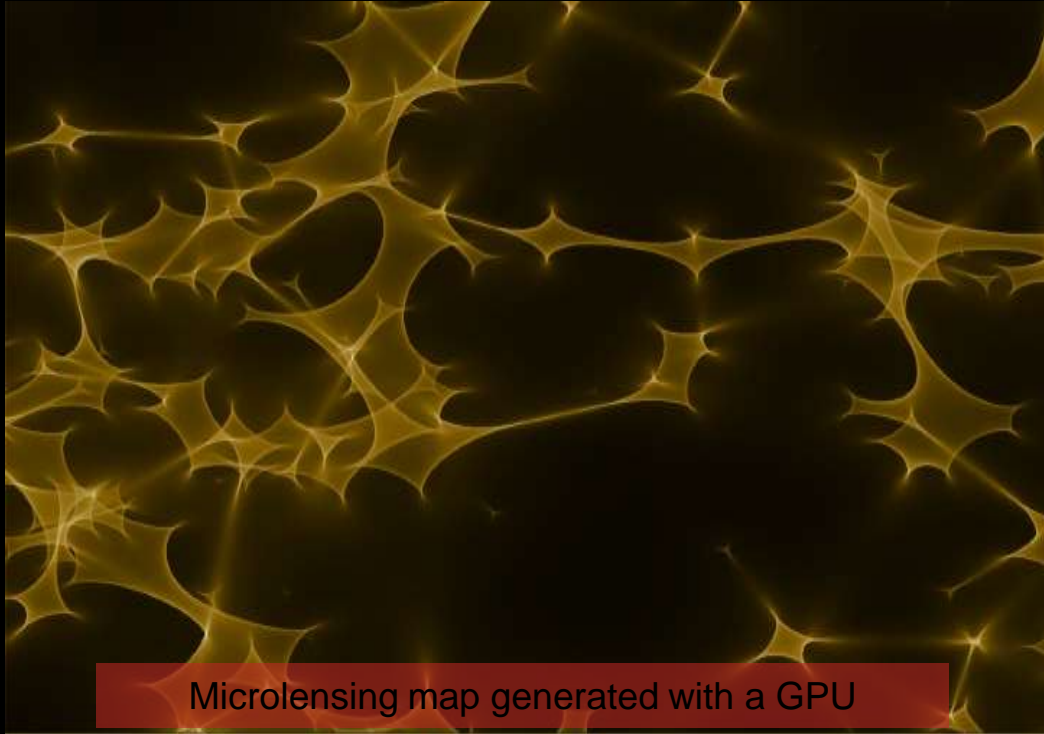
- **Aim: Develop a generalised approach to using GPUs for scientific computing.**
- **Method: Algorithm analysis techniques allow rapid assessment of GPU-suitability for a broad range of problems.**

GPUs are taking us to exciting new territories, beyond the current CPU multi-core corner

- A generalised approach to GPUs makes it easier to exploit their power and avoids the risk of wasted development time.

GERLUMPH: Gravitational Microlensing survey

C.Fluke, G.Vernardos, D.Croton, N.Bate (USyd)



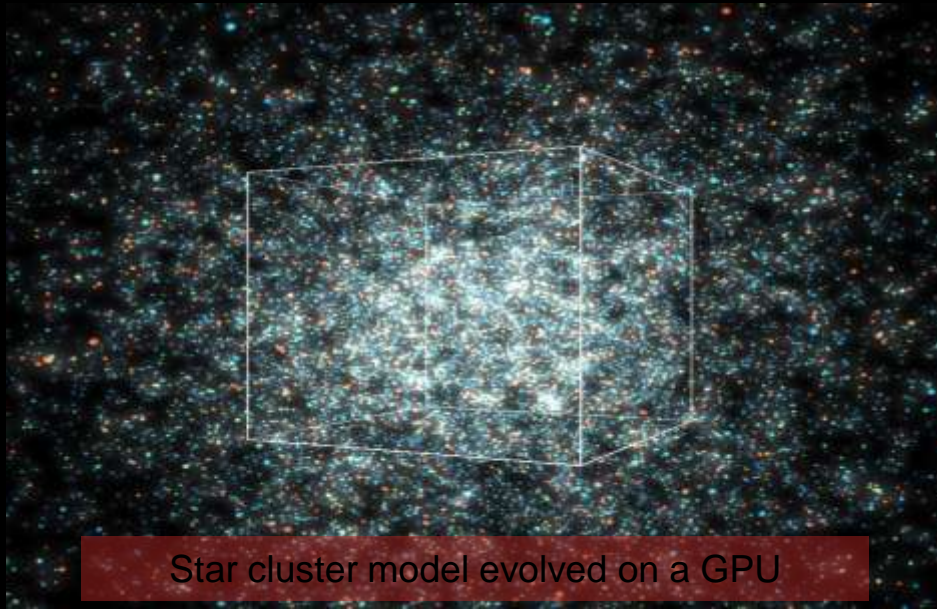
- **Aim: Perform high-resolution numerical parameter survey in preparation for future all-sky searches for microlensed quasars.**
- **Method: ray-shooting is embarrassingly parallel, so ideal for GPUs.**

- Improved microlensing modeling will lead to better constraints on quasar properties.



Realistic N-body Models of Star Clusters

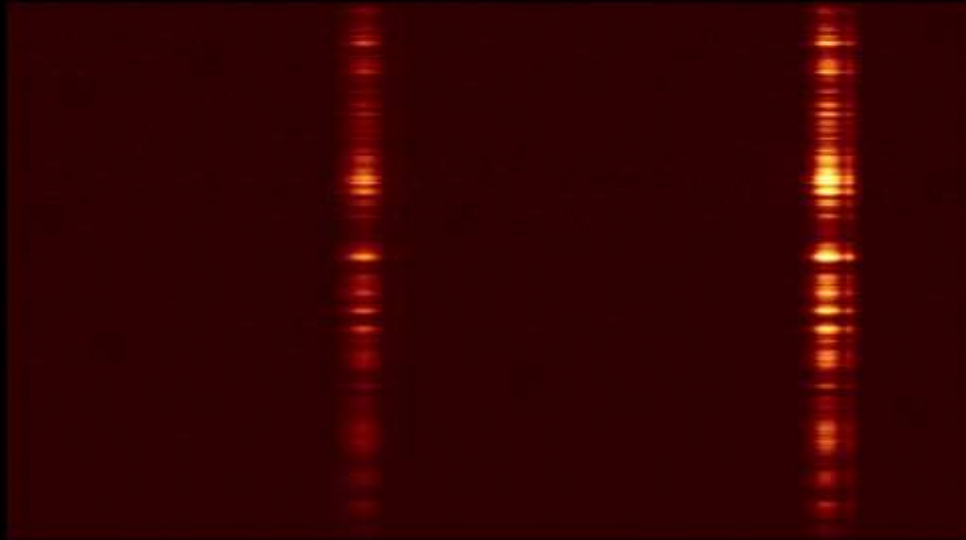
Jarrood Hurley, Anna Sippel, Juan Madrid, Guido Moyano-Loyola



- **Aim:** Evolve *direct* N-body models of star clusters to study their evolution from the dawn of the universe until today.
 - **Method:** Gravitational force calculations between stars are optimal to be parallelized and hence excellent for GPUs.
- Larger and improved star cluster models will lead to better understanding of how they evolve within the framework of galaxy formation.

DSPSR: Digital Signal Processing for Radio Pulsars

W. van Straten, A. Jameson, J. Kocz, M. Bailes, & P. Demorest (NRAO)



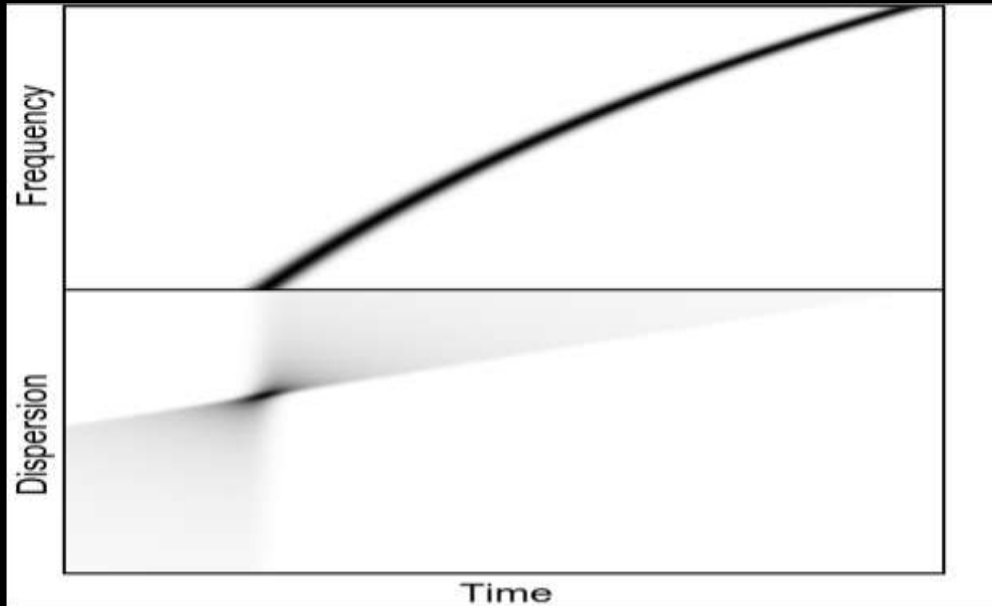
Pulsar signal processed using GPU cluster

- **Aim: Develop affordable & efficient pulsar instrumentation for next-generation telescopes (e.g. Square Kilometre Array)**
- **Method: GPUs love the large FFT sizes required to correct the dispersive effects of the interstellar medium.**

- GPUs have eliminated the need for costly specialized hardware at the observatory

Spotting radio transients with GPUs

B.Barsdell, M.Bailes, D.Barnes (Monash), C.Fluke



Signals smeared in the interstellar medium can be reconstructed in real-time with GPUs

- **Aim: Detect radio pulsars and transient events in real time.**
- **Method: GPUs eat up signal processing and large parameter space searches, allowing us to break through the real-time barrier.**

- Real-time detection opens a new window on the Universe, allowing us to catch short-lived transient events in the act.



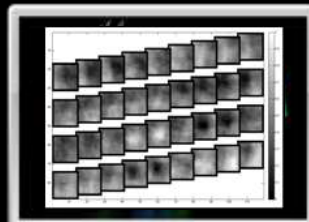
17X

Neuro-imaging
Georgia Tech



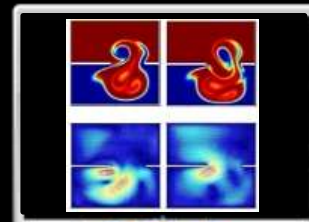
4.5X

Geophysics
Boise State



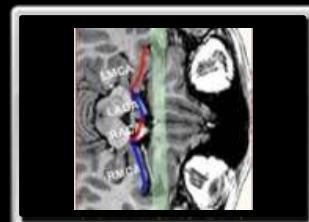
20X

Video Processing
Google



10X

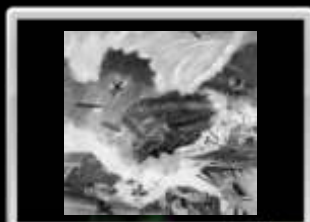
Fluid Dynamics
LSU



12X

Medical Devices
Spencer Tech

<http://www.accelereyes.com/successstories>



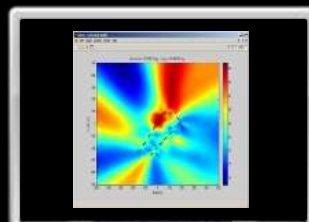
5X

Weather Modeling
NCAR



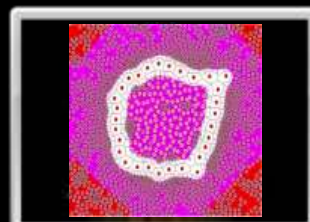
35X

Power Engineering
IIT India



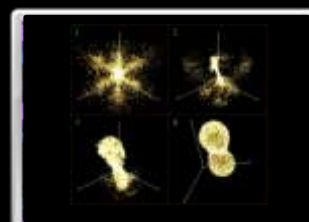
17X

Track Bad Guys
BAE Systems



70X

Drug Delivery
Georgia Tech



35X

Bioinformatics
Leibniz Research



GPU as Signal Processor & Texture Concepts

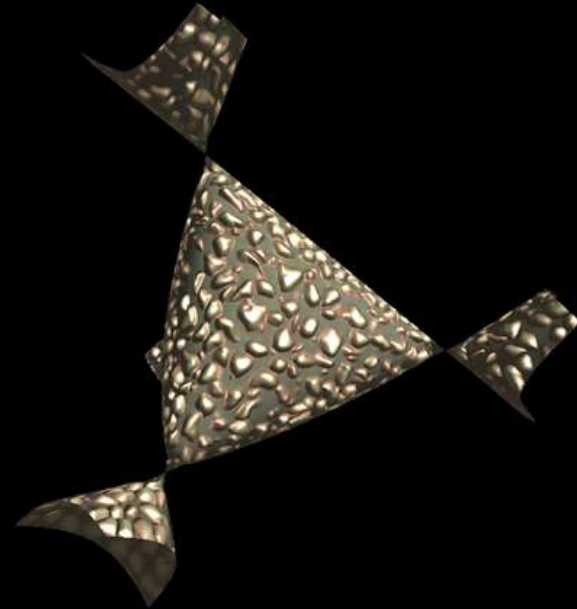
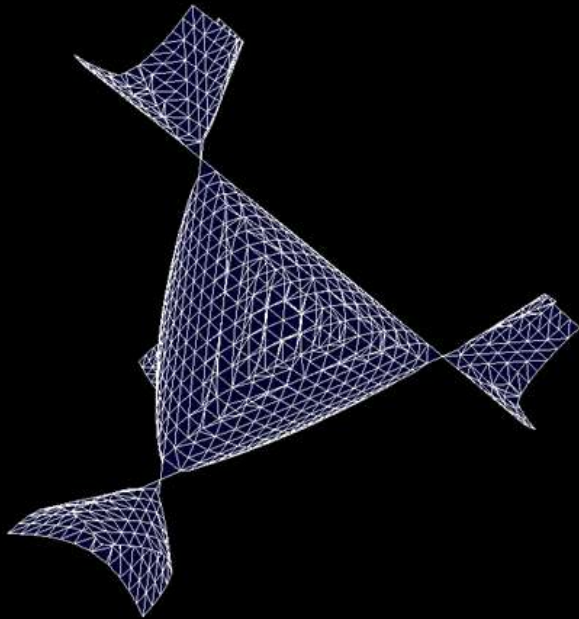


GPU as signal processor

- Massive on-board memory bandwidth (150 GB/s)
- PCIeexpress can be bottleneck, but can overlap computation and transfers
- CUDA C: Program (Kernel) for single data element, parallelization "in background" (e.g. branching divergences handled by hardware)
- Scales with future GPUs (Design intention: 2 x of existing code per architecture generation)
- TEXTURES:
GPU-specific technology that accelerates Image and volume processing!
- See
Dr.-Ing. Karl Schwarz, Siemens Medical Computer Tomography, International Supercomputing 2011 Proceedings,
"Texture Unit as a Performance booster"

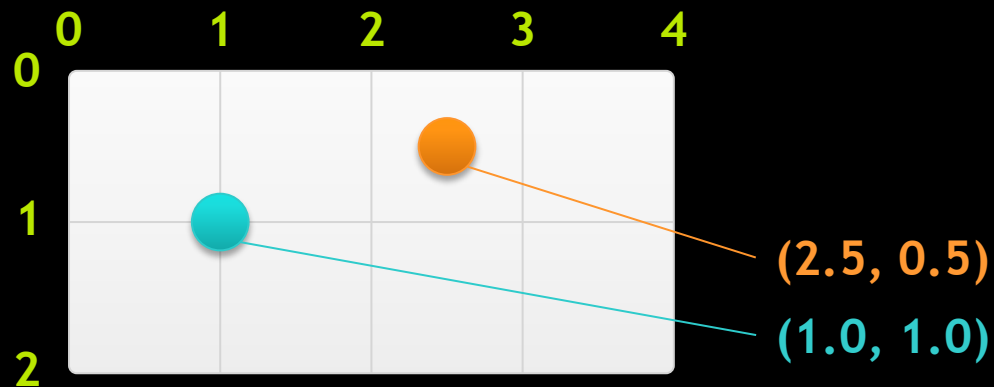
Texturing

- **Original purpose:**
 - Provide surface coloring for 3D meshes (a "wrapping")
 - 3D mesh has "texture coordinates", hardware looks up 2D color array



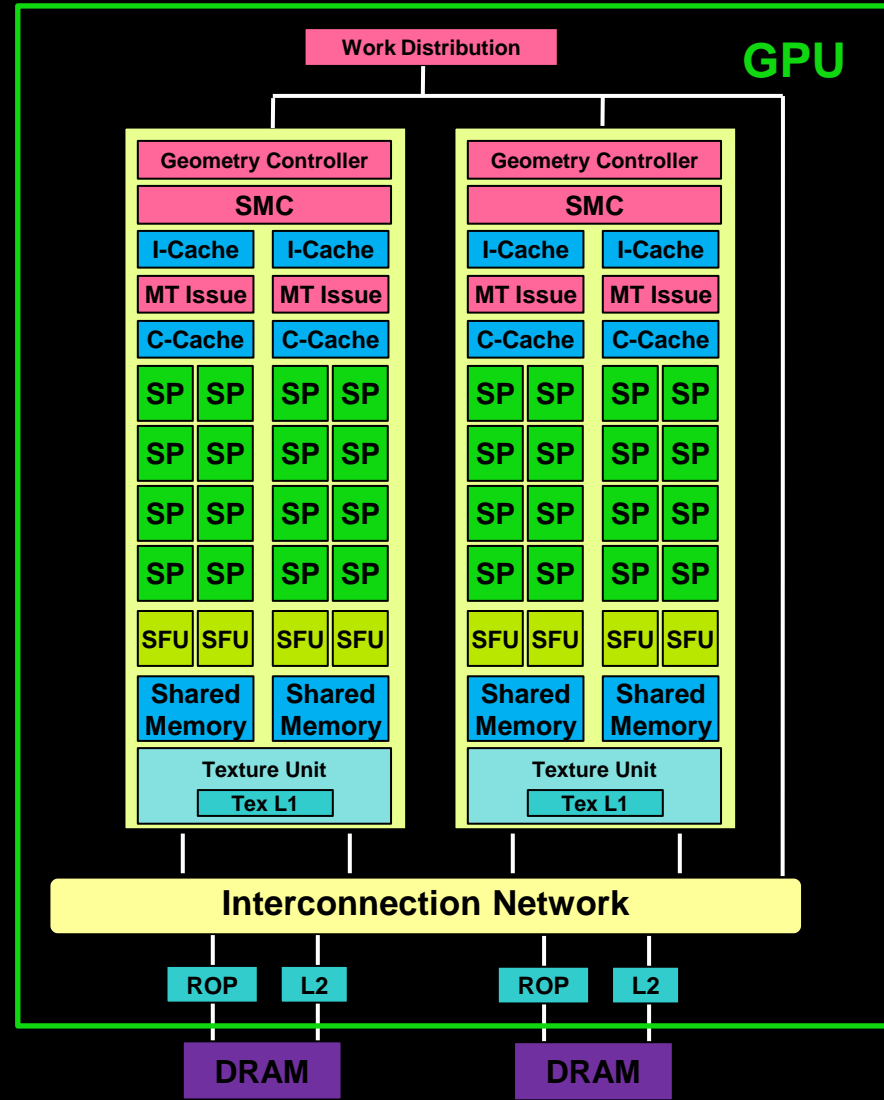
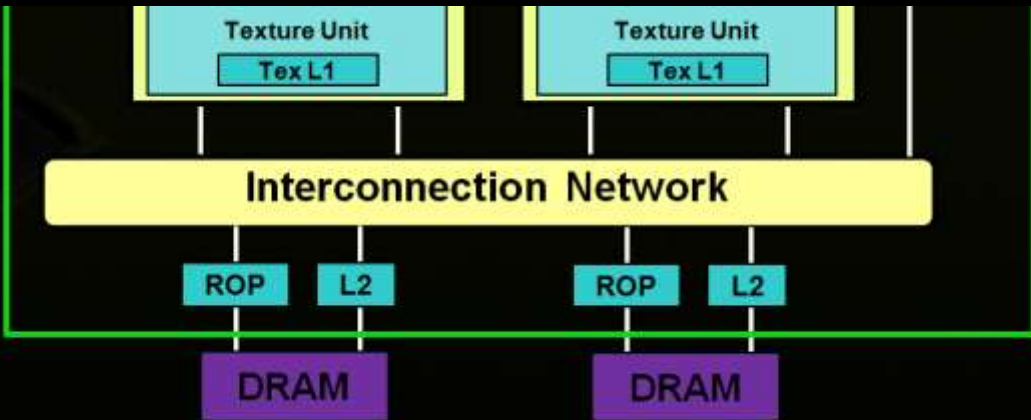
Textures

- **Read-only object**
 - Dedicated cache
- **Dedicated filtering hardware**
(Linear, bilinear, trilinear)
- **Addressable as 1D, 2D or 3D**
- **Out-of-bounds address handling**
(Wrap, clamp)

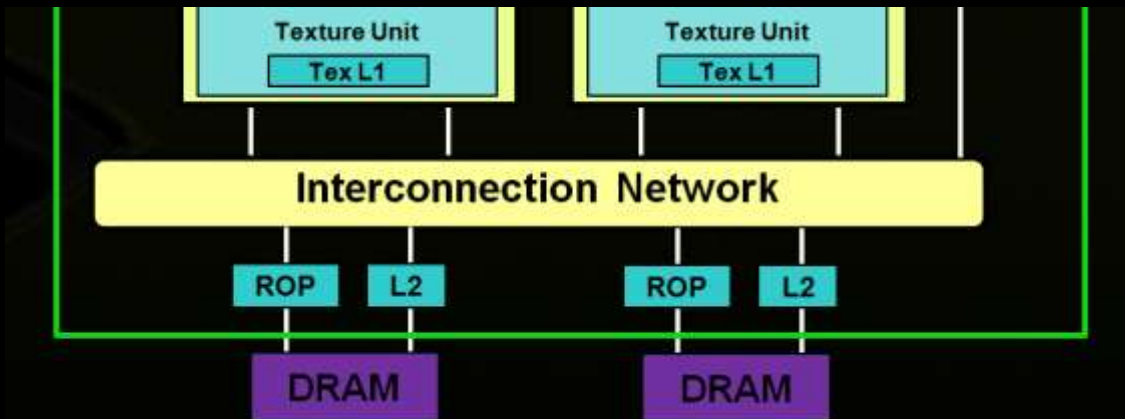


Texture Unit

- **Texture Read:**
Global memory read via texture hardware path
- **Data reads are cached**
 - Texture Cache (separate from L1)
 - Specialized for 2D/3D spatial locality



Texture Unit



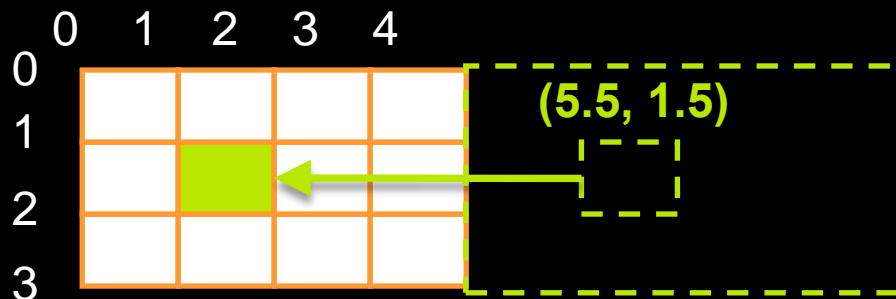
- **Data conversion (integer to float, 16 bit float to 32 bit float)**
- **Data Interpolation (aka Filtering)**
 - Linear / bilinear / trilinear data interpolation in hardware
- **Boundary modes (for “out-of-bounds” addresses)**
 - Addressable in 1D, 2D, or 3D.
 - Coordinate normalization mode (access becomes resolution-independent)
 - Clamp to edge / Clamp to Border color / Repeat / Mirror
- **Works best with CUDA Array as Data Storage**

Texture Coordinates

- Texture fetch in device code takes floating point **texture coordinates**
- **Lookup mode** and coordinates determine data element fetch from global memory: "Nearest neighbour" mode uses less data than "linear interpolation" mode
- Coordinate bounds can reflect input data dimensions, or be **normalized** (0.0 .. 1.0)
- Boundary handling in different ways:

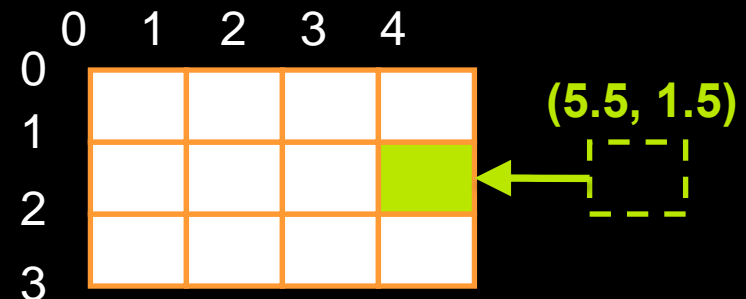
Wrap

- Out-of-bounds coordinate is wrapped (modulo arithmetic)



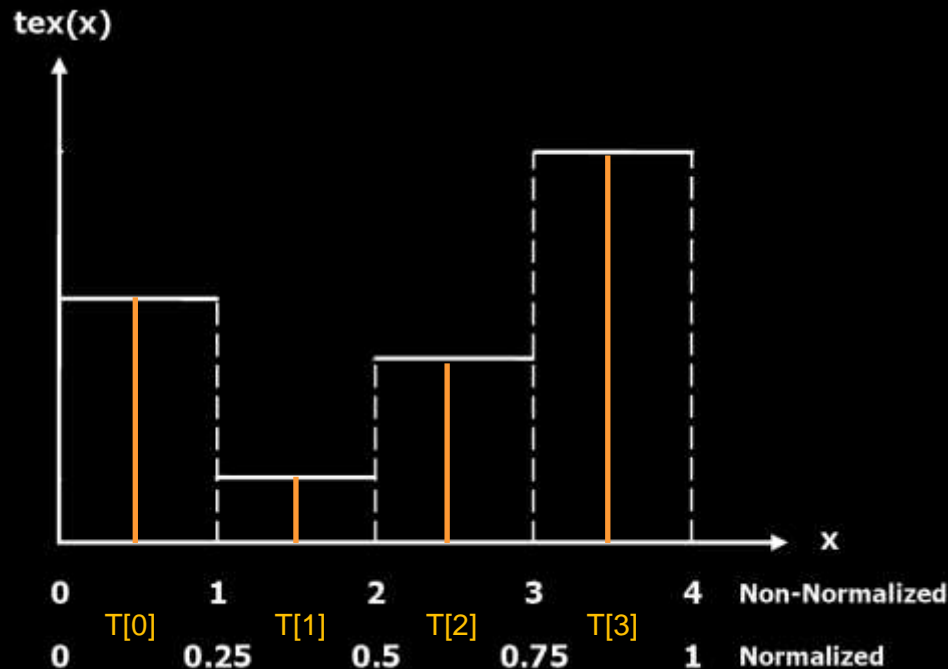
Clamp

- Out-of-bounds coordinate is clamped to closest boundary



Texture Data Processing

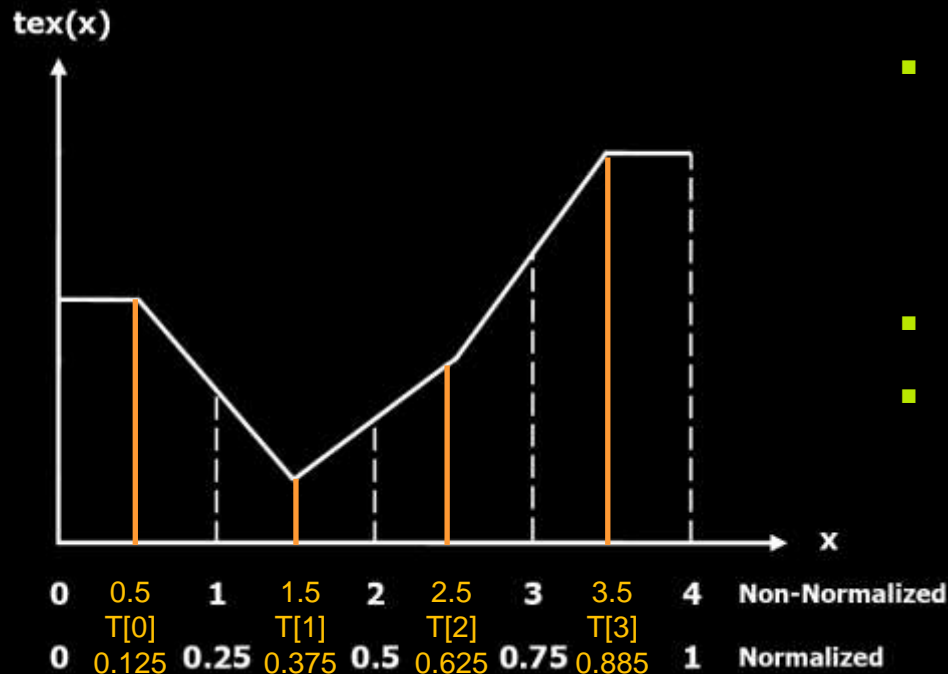
- **Texture unit can convert integer input to floating point output**
 - E.g. 8bit input: `uchar4(255, 128, 0, 0)` becomes `float4(1.0, 0.5, 0.0, 0.0)`
- **Coordinate to Data mapping for "Nearest neighbour" mode:**
 - Example: Input data T, four values:



- All input data elements cover equal output ranges
- Details in Programming Guide, Appendix E

Texture Interpolation

- **Texture unit can interpolate between adjacent data elements**
 - Fractional part of texture coordinate becomes interpolation weight (Note: Interpolation weight is 8 bit quantized!)
 - Only in float conversion mode, bind to CUDA array or pitchlinear memory



- Warning:
Input's data values can NOT be read at integer offsets!
- But: Additional GFlops!
- Details in Programming Guide, Appendix E

Summary

- **Texturing provides additional performance**

- Extra cache capacity
- Linear interpolation of adjacent data in hardware
- Array boundary handling
- Integer-to-float conversion, data unpacking

- **Algorithmic design considerations**

- Texture binding modes (linear memory, pitchlinear memory, CUDA Array)
- Texture coordinate offsets for correct linear interpolation
- 8bit weight quantization during linear interpolation
- Can't flush texture cache during kernel execution
- 3D: xy-interpolation (layered textures) vs. Trilinear xyz-interpolation (3D textures)

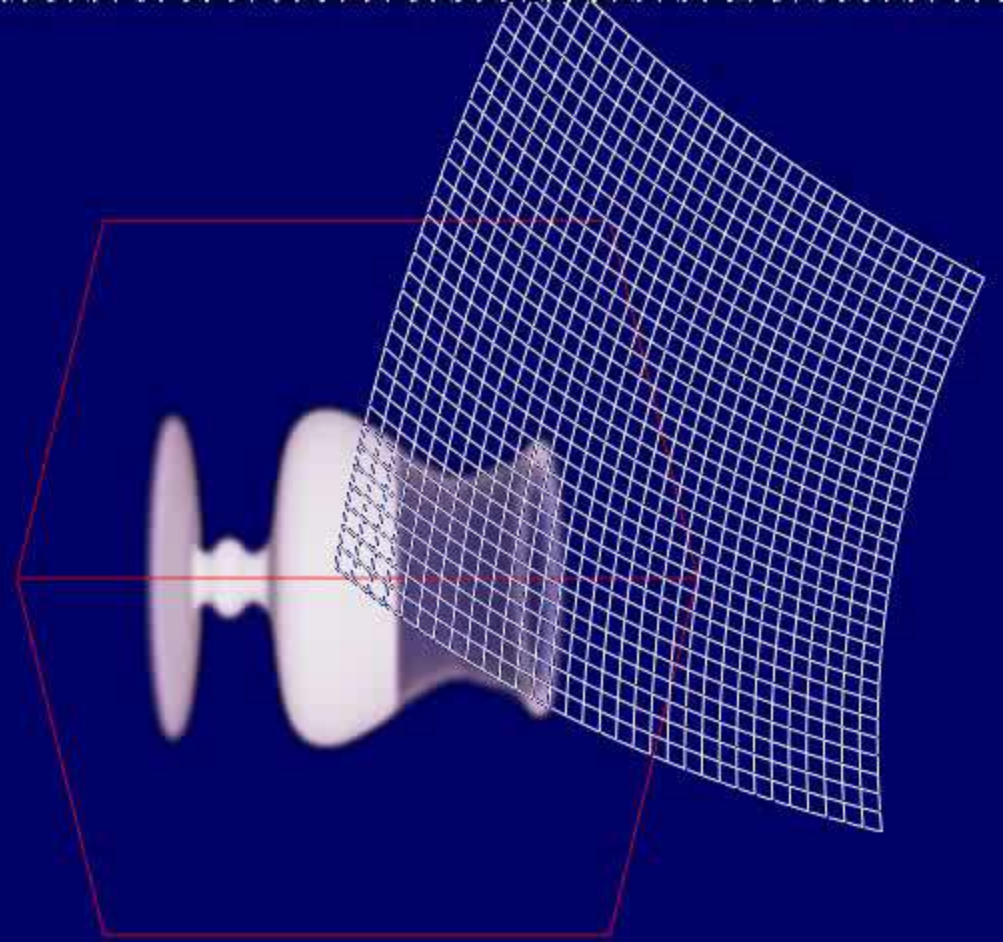
Questions? ...

Further reading

- **Textures, Surfaces and CUDA Array creation:**
Programming Guide, 3.2.10 Texture and Surface Memory
- **Texture lookups in device code:**
Programming Guide, Appendix B.8
- **Specification of texture interpolation modes and clamping:**
Programming Guide, Appendix E
- **Surface read/write operations in device code:**
Programming Guide, Appendix B.9
- **Texture and surface exchange with OpenGL / DirectX:**
Programming Guide, 3.2.11 Graphics Interoperability
- **Texture usage in applications:**
Best Practices Guide, 3.2.4 Texture Memory

Eikonal Rendering

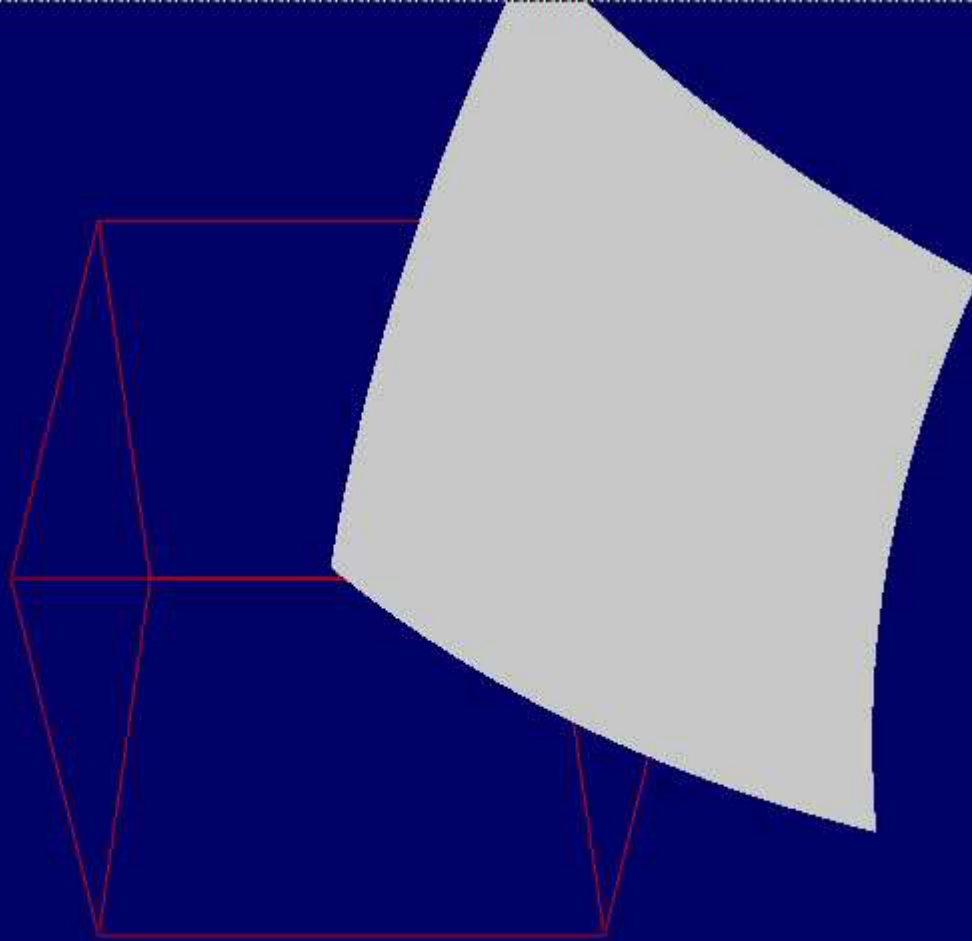
Current frame: 0
CPU load: 0%
Input size: 720x576
Output size: 720x576



- **SIGGRAPH 2007 (Computer Graphics)**
- **Simulates light wavefront passing through volume of refractive indices**
- **All simulation on GPU, esp. *wavefront culling and tessellation***
- **10 secs running, took 10 minutes on CPU**
- **Google and YouTube "Eikonal Rendering"**

Eikonal Rendering

Current frame: 0
CPU load: 0%
Input size: 720x576
Output size: 720x576



CUDA Programming Resources

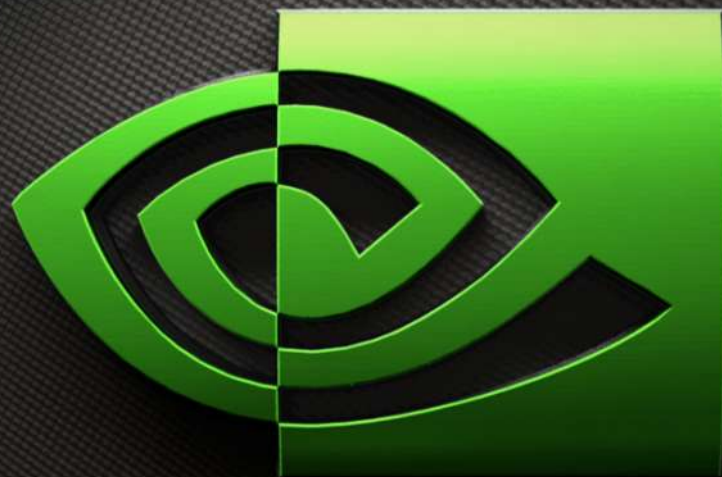
- **CUDA Toolkit**
 - Compiler, free libraries like cuBLAS, cuFFT
 - Documentation (Programming Guide, Best Practices Guide)
 - Free download for Windows, Linux, and MacOS
- **GPU Computing SDK**
 - Code samples
 - Whitepapers
- **Instructional materials on NVIDIA Developer site**
 - CUDA introduction & optimization webinar: slides and audio
 - Parallel programming course at University of Illinois UC
 - Tutorials
 - Forums

GPU Tools

- **Profiler**
 - Available for all supported OSs
 - Command-line or GUI
 - Sampling signals on GPU for:
 - Memory access parameters
 - Execution (serialization, divergence)
- **Debugger**
 - Linux: `cuda-gdb`
 - Windows: Parallel Nsight
 - Runs on the GPU

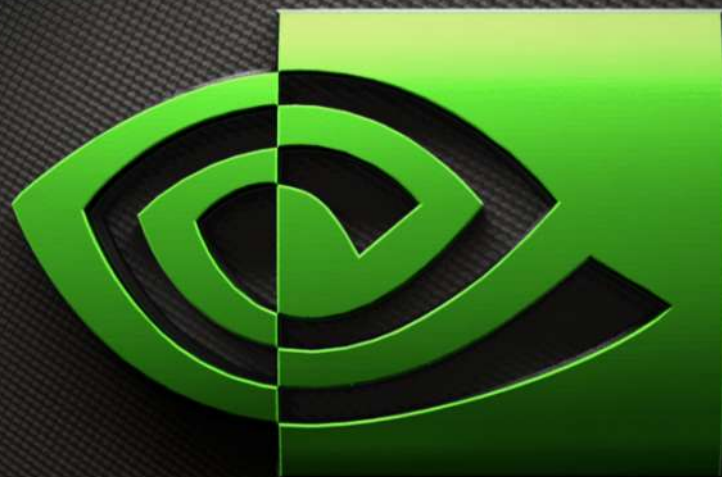


**(Application)
Questions?**



nVIDIA

Latest Developments in CUDA Eco-System



nVIDIA

CUDA 4.0

CUDA 4.0

Application Porting Made Simpler

Faster Multi-GPU Programming
GPUDirect 2.0

Rapid Application Porting
Unified Virtual Addressing

Easier Parallel Programming in C++
Thrust

NVIDIA CUDA Overview

New in
CUDA 4.0

Platform

GPUDirect 2.0
Fast Path to Data

Hardware Support

ECC Memory
Double Precision
Native 64-bit Architecture
Concurrent Kernel Execution
Dual Copy Engines
Multi-GPU support
6GB per GPU supported

Operating System Support

MS Windows 32/64
Linux 32/64 support
Mac OSX support

Cluster Management
NVIDIA GPUDirect
Tesla Compute Cluster (TCC)
Graphics Interoperability

Programming Model

Unified Virtual Addressing

C++ new/delete
C++ Virtual Functions

C support

- NVIDIA C Compiler
- CUDA C Parallel Extensions
- Function Pointers
- Recursion
- Atomics
- malloc/free

C++ support

- Classes/Objects
- Class Inheritance
- Polymorphism
- Operator Overloading
- Class Templates
- Function Templates
- Virtual Base Classes
- Namespaces

Fortran, OpenCL

Libraries

Thrust C++ Library
Templated Performance
Primitives

NVIDIA Library Support

Complete math.h
Complete BLAS Library (1, 2 and 3)
Sparse Matrix Math Library
RNG Library
FFT Library (1D, 2D and 3D)
Image Processing Library (NPP)
Video Processing Library (NPP)

3rd Party Math Libraries

- CULA Tools
- MAGMA
- IMSL
- VSIPL

Tools

Parallel Nsight Pro

NVIDIA Tools Support

Parallel Nsight 1.0 IDE
cuda-gdb Debugger with multi-GPU
CUDA/OpenCL Visual Profiler
CUDA Memory Checker
CUDA C SDK
CUDA Disassembler

CUDA Partner Tools

Allinea DDT
RogueWave /Totalview
Vampir
Tau
CAPS HMPP

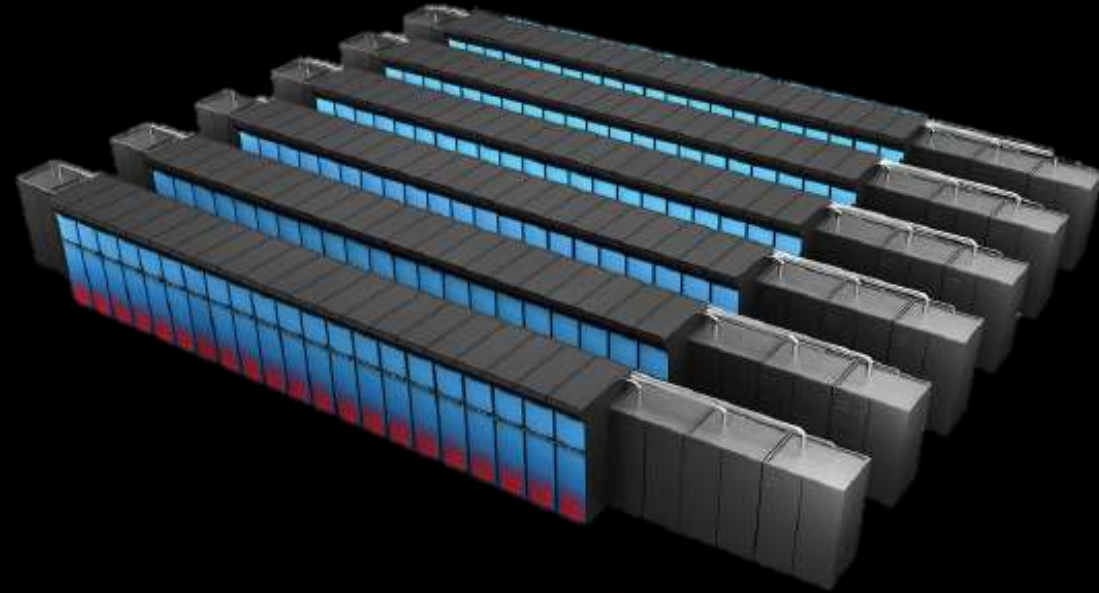
World's Top Open Science Computing Research Facility



18,000+ Tesla GPUs

20+ PetaFlops

3x More Energy Efficient than
Current #1 (K Computer)



TITAN