ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

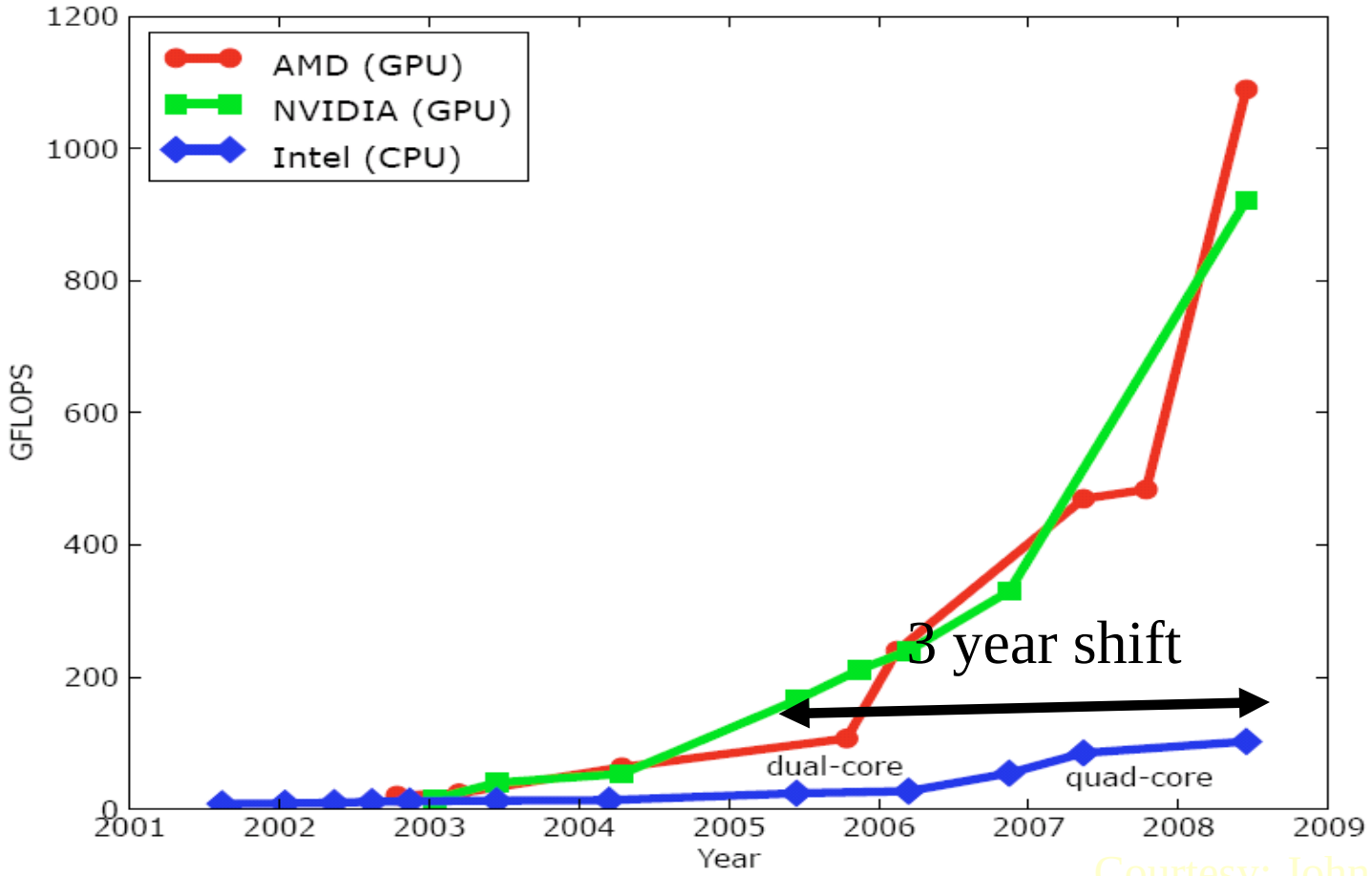
The Parallel Revolution in Computational Science and Engineering

applications, education, tools, and impact

Wen-mei Hwu
University of Illinois, Urbana-Champaign

The Energy Behind Parallel Revolution

- Calculation: 1 TFLOPS vs. 100 GFLOPS
- Memory Bandwidth: 100-150 GB/s vs. 32-64 GB/s



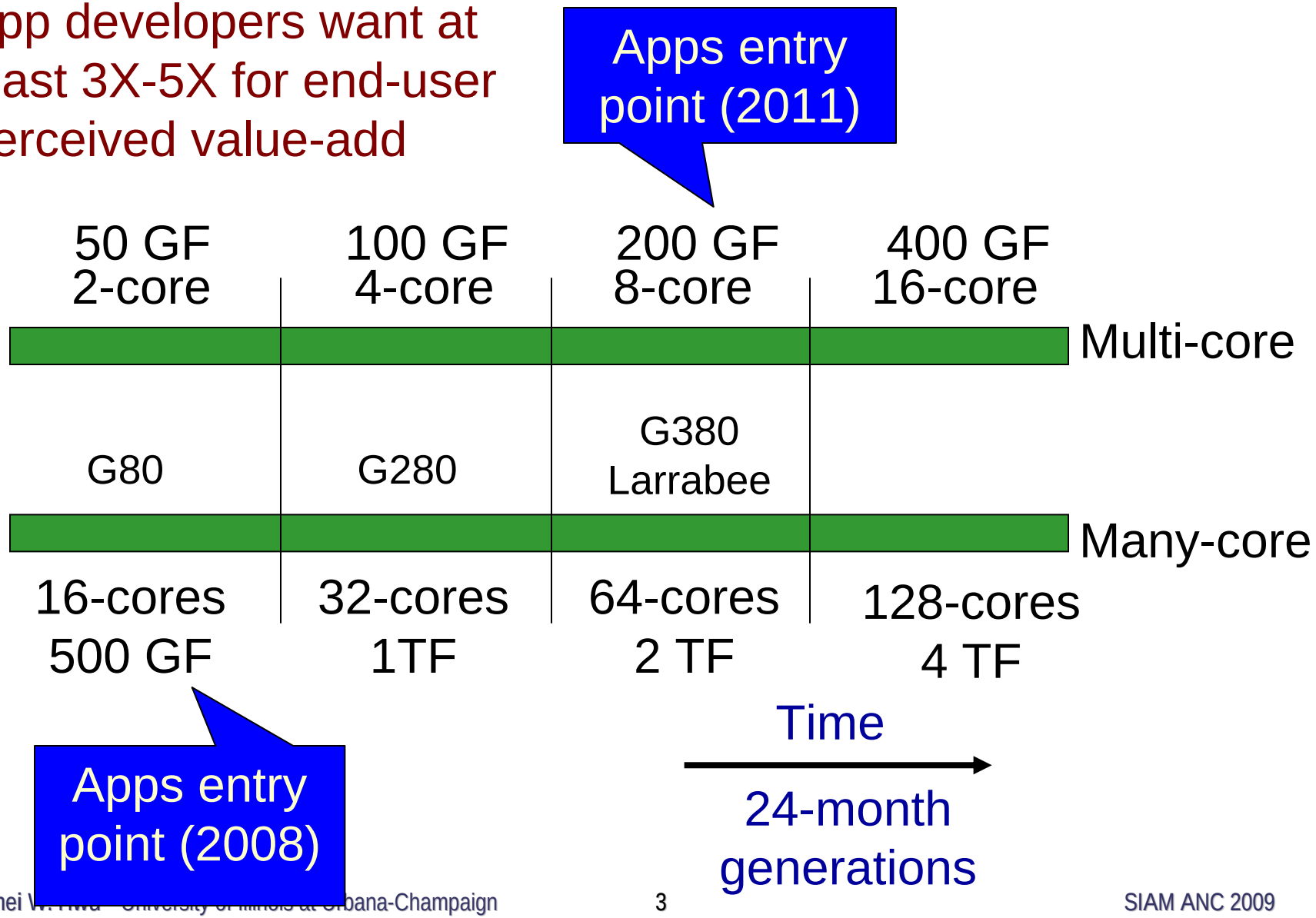
Courtesy:
John Owens

Courtesy: John Owens

- Multi-core and GPU in every PC– massive volume and potential impact

Applications Entry Timeframes

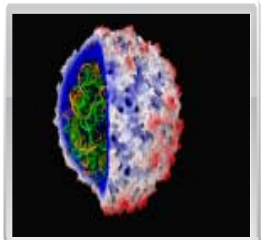
App developers want at least 3X-5X for end-user perceived value-add



What are these applications?

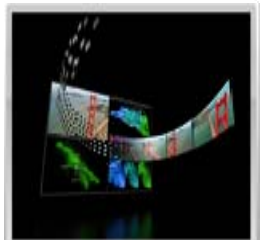
146X

Interactive visualization of volumetric white matter connectivity



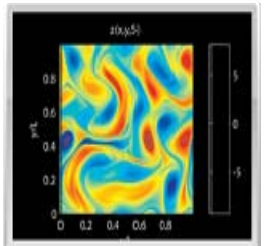
36X

Ionic placement for molecular dynamics simulation on GPU



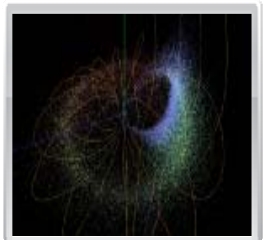
19X

Transcoding HD video stream to H.264



17X

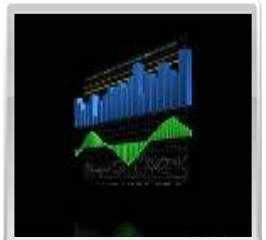
Simulation in Matlab using .mex file CUDA function



100X

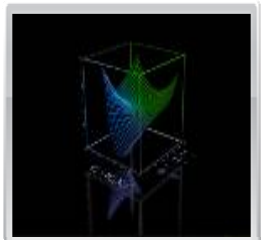
Astrophysics N-body simulation

Courtesy NVIDIA



149X

Financial simulation of LIBOR model with swaptions



47X

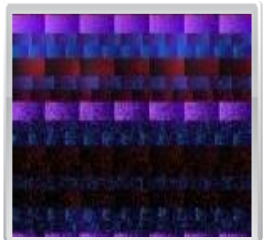
GLAME@lab: An M-script API for linear Algebra operations on GPU

20X

Ultrasound medical imaging for cancer diagnostics

24X

Highly optimized object oriented molecular dynamics

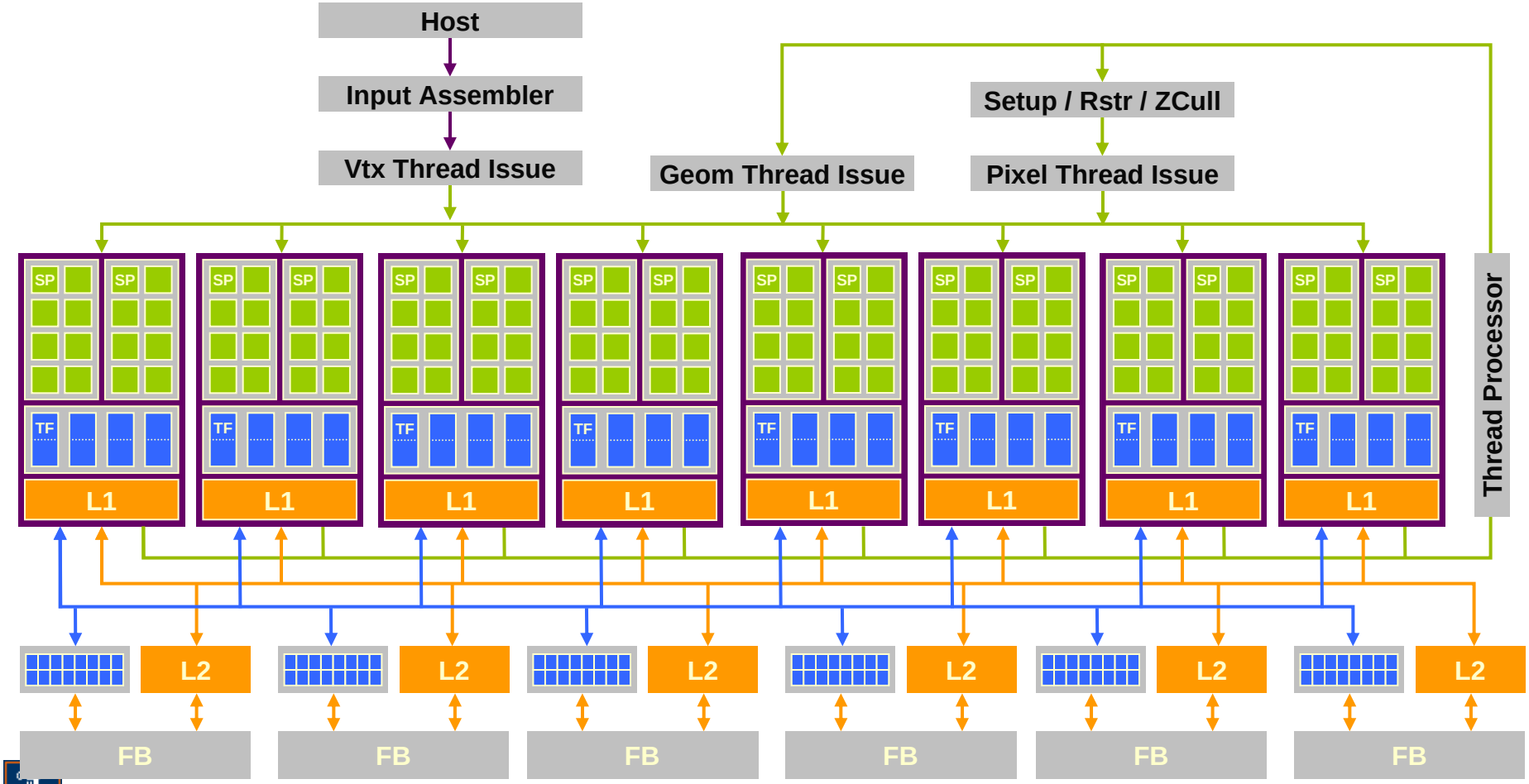


30X

Cmatch exact string matching to find similar proteins and gene sequences

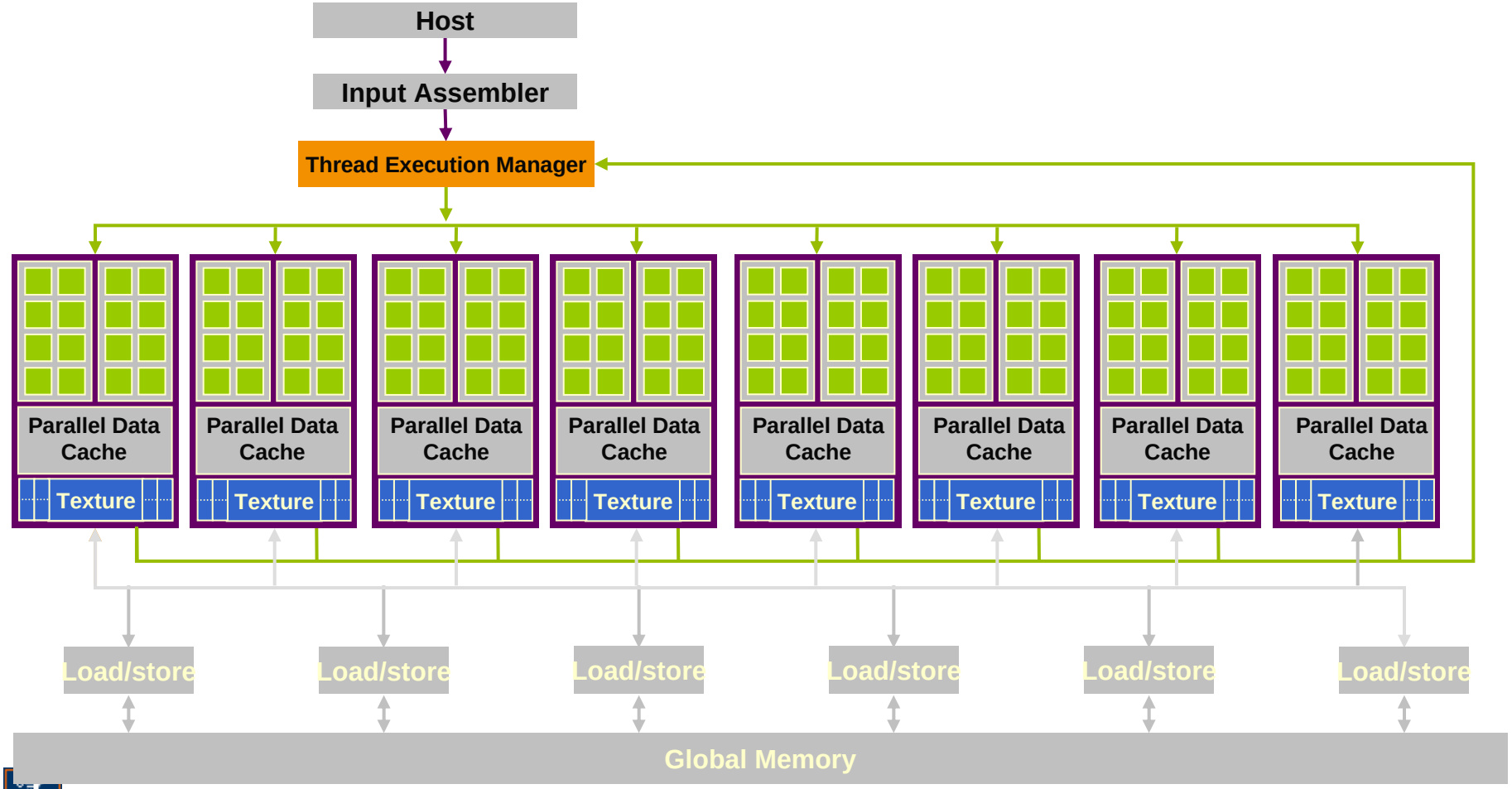
GPU – Graphics Mode

- The future of GPUs is programmable processing
- So – build the architecture around the processor



GPU CUDA Mode

- Processors execute computing threads
- New operating mode/HW interface for computing



UIUC/NCSA AC Cluster

- 32 nodes
  - 4-GPU (GTX280, Tesla), 1-FPGA, quad-core Opteron node at NCSA
  - GPUs donated by NVIDIA
  - FPGA donated by Xilinx
  - 128 TFLOPS single precision, 10 TFLOPS double precision
- Coulomb Summation:
  - 1.78 TFLOPS/node
  - 271x speedup vs. Intel QX6700 CPU core w/ SSE

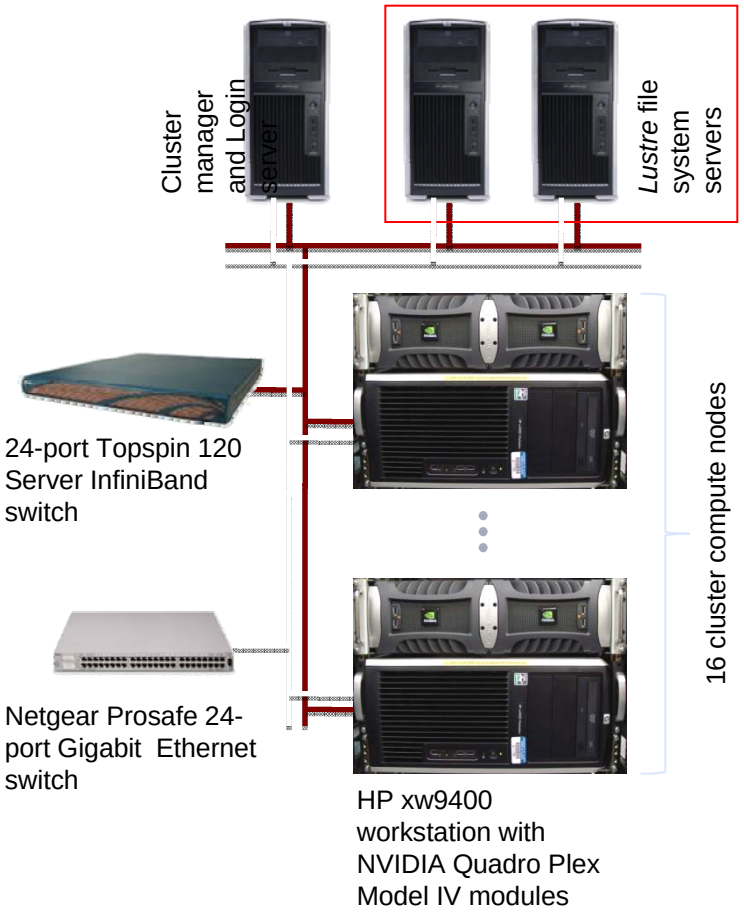
UIUC/NCSA AC Cluster

http://www.ncsa.uiuc.edu/Projects/GPUcluster/

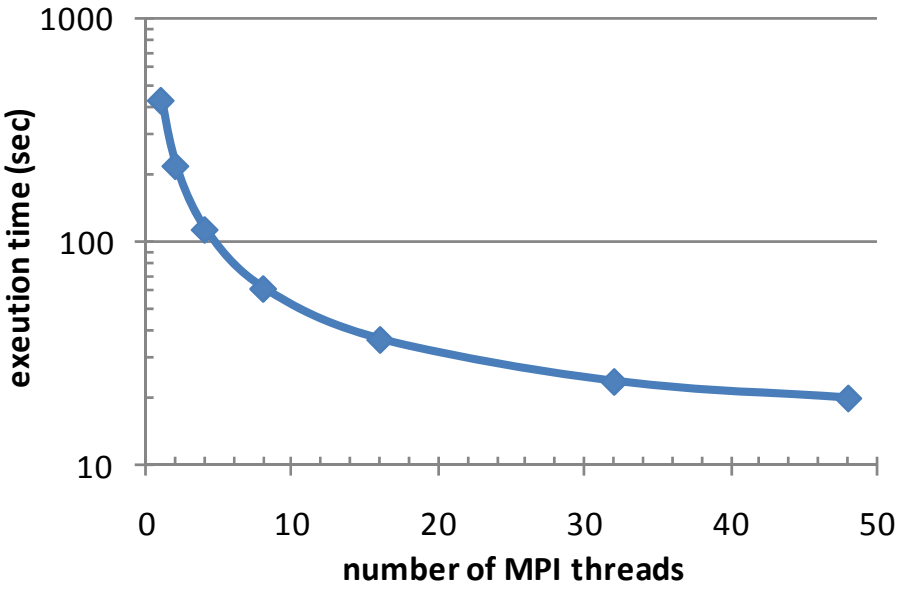
A partnership between NCSA and academic departments.

Multi-GPU MPI Implementation

GPU cluster



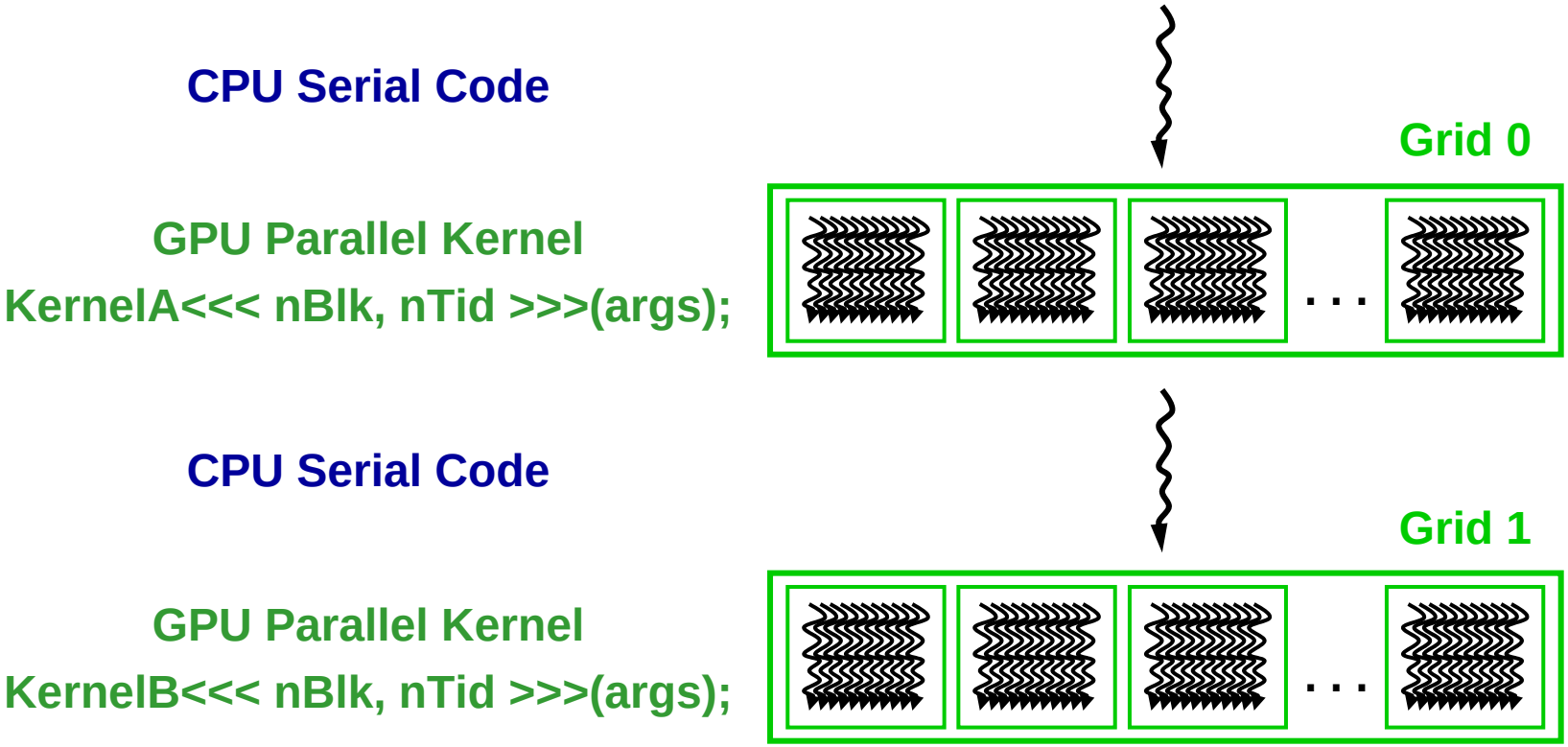
Performance scaling



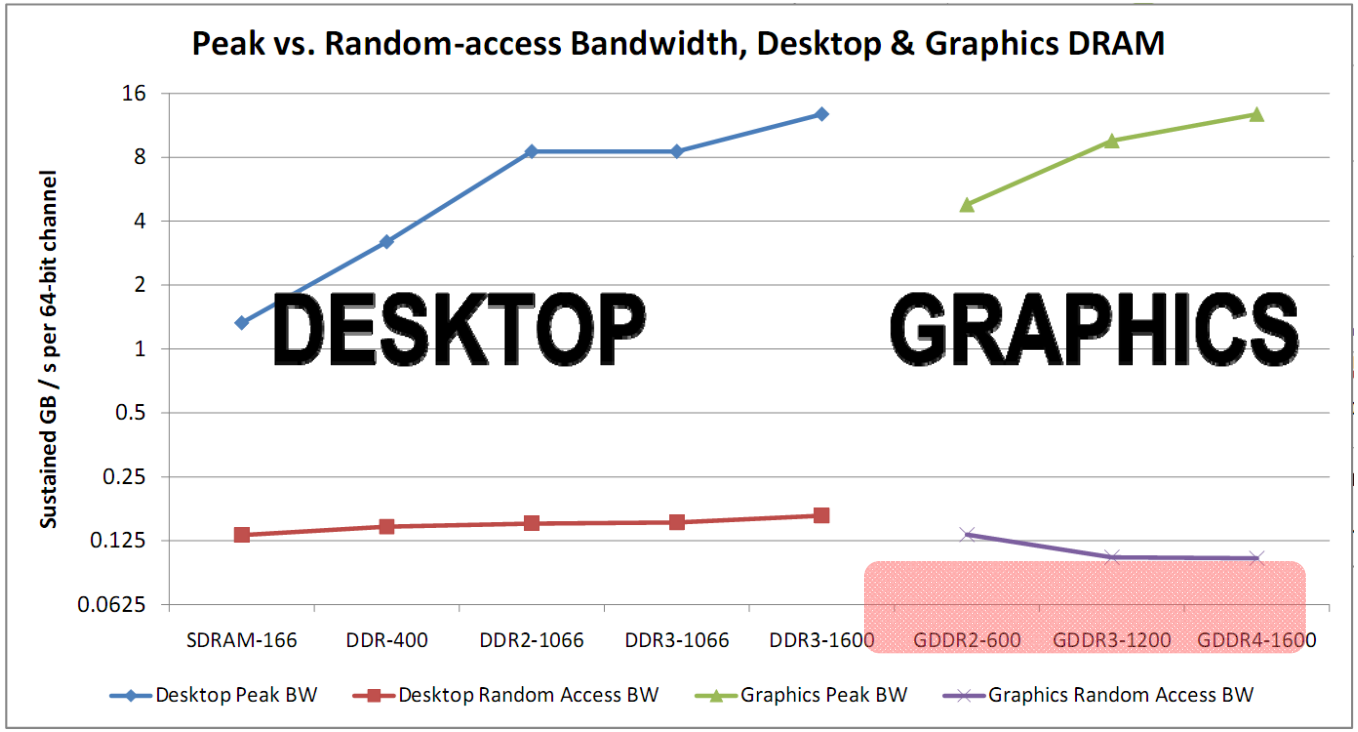
Two Point Angular Correlation

CUDA - No more shader functions.

- CUDA integrated CPU+GPU application C program
  - Serial or modestly parallel C code executes on CPU
  - Highly parallel SPMD kernel C code executes on GPU



DRAM Bandwidth Trends Sets Programming Agenda



- Random access BW 1.2% of peak for DDR3-1600, 0.8% for GDDR4-1600 (and falling)
- 3D stacking and optical interconnects will unlikely help.

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

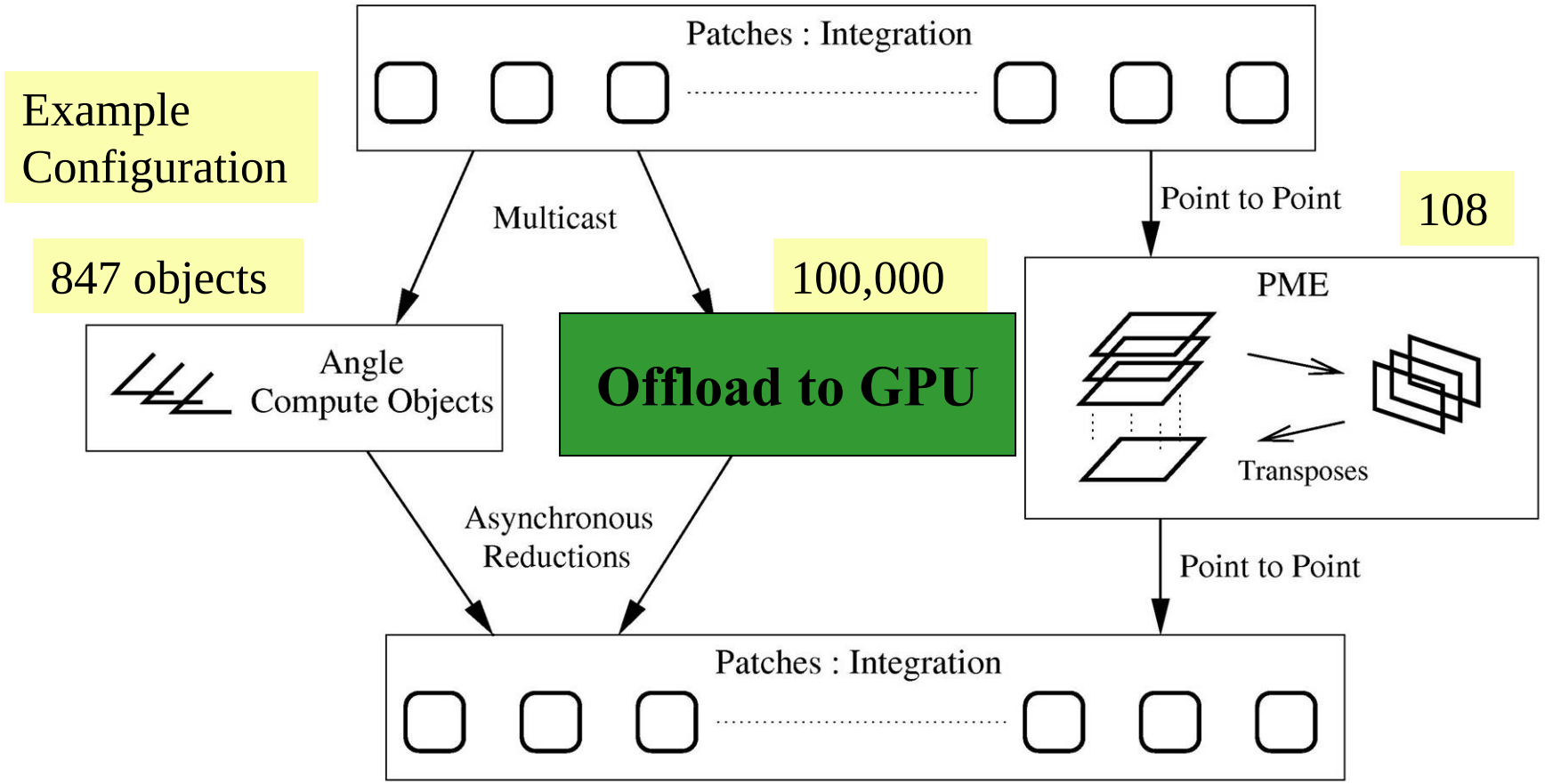
It is all about applications!

Illinois CUDA Center of Excellence
and the IACAT Community

parallel.illinois.edu

NAMD Overlapping Execution

Phillips *et al.*, *SC2002*.

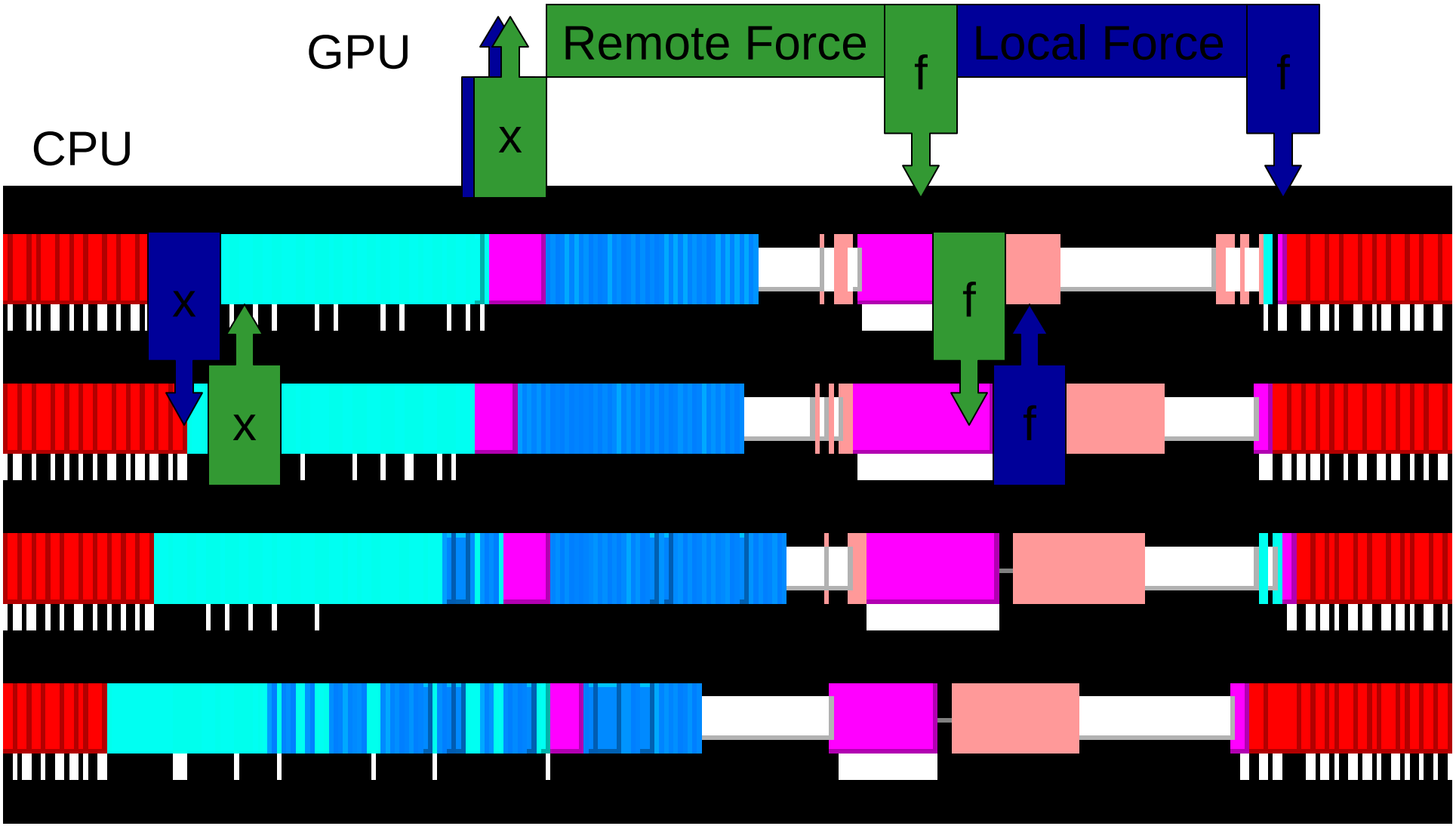


Objects are assigned to processors and queued as data arrives.

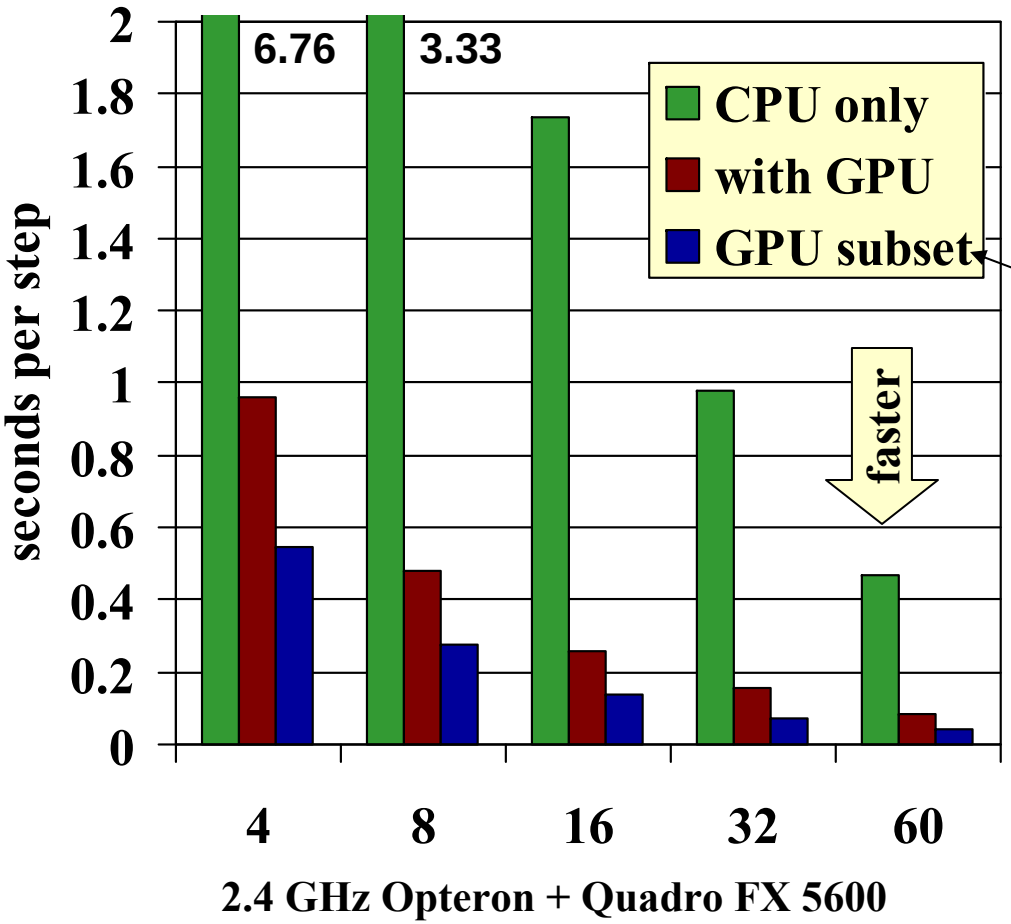
Klaus Schulten and team, over 29,000 registered users

Actual Timelines from NAMD

Generated using Charm++ tool "Projections"



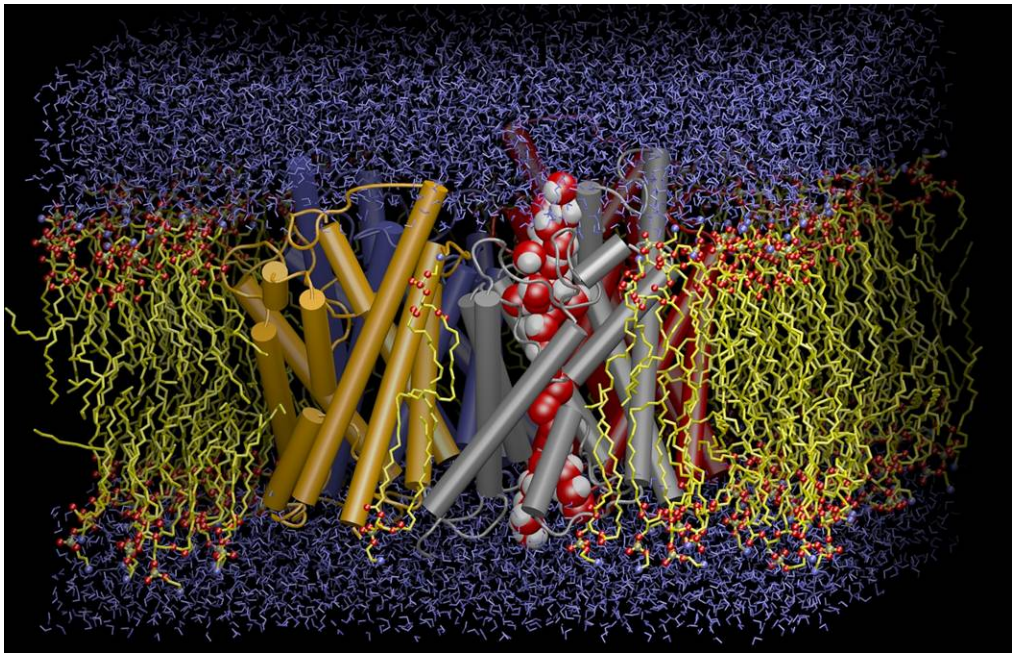
NAMD on QP Cluster – total application time



Amount of time the GPU is actually doing work

VMD

- "Visual Molecular Dynamics" (120,000 registered users)
- Visualization of molecular dynamics simulations, sequence data, volumetric data, quantum chemistry data, particle systems
- http://www.ks.uiuc.edu/Research/vmd/



Molecular Orbital Computation/Display - Molekel, MacMolPlt, and VMD

Program/Kernel	cores	C60-a	C60-b	Thr-a	Thr-b	Kr-a	Kr-b
Molekel CPU	1	39	25	175	108	617	138
MacMolPlt CPU	4	97	66	361	265	2668	632
VMD gcc-cephes	4	126	100	518	374	2655	892
VMD icc-sse-cephes	4	658	429	2428	1366	10684	2968
VMD gcc-approx	4	841	501	2641	1828	11055	4060
VMD icc-sse-approx	4	2314	1336	8829	5319	33818	9631
VMD CUDA 8800 GTX	1	14166	8565	45015	32614	104576	61358
VMD CUDA GTX 280	1	21540	13338	62277	45498	119167	78884

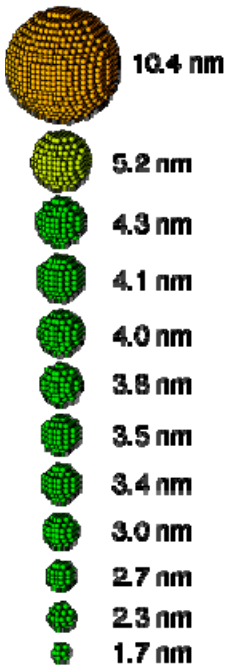
Units: 10^3 grid points/sec

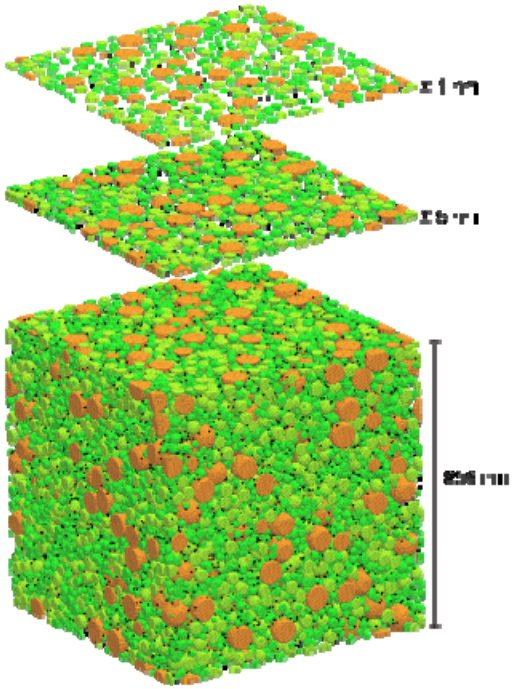
Larger numbers indicate higher performance.

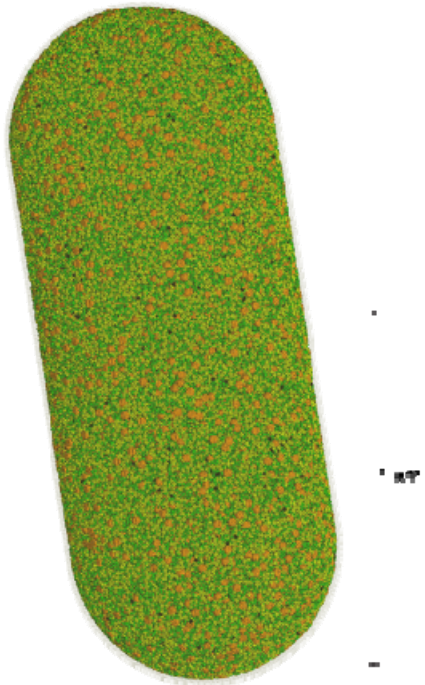
High Performance Computation and Interactive Display of Molecular Orbitals on GPUs and Multi-core CPUs. J. Stone, J. Saam, D. Hardy, K. Vandivort, W. Hwu, and K. Schulten, 2nd Workshop on General-Purpose Computation on Graphics Pricessing Units (GPGPU-2) (in press)

Whole-cell Diffusion Modeling

- Zan Luthey-Schulten group (Chemistry, UIUC) is using CUDA to simulate reaction diffusion in three dimensions on a lattice.
- GPUs and CUDA enable simulations at cellular length and time scales, enabling study of stochastic biochemical networks *in vivo*.
- Simulations sample the cell state's probability distribution according to the reaction-diffusion master equation.







Using size and population distributions from proteomic data, an approximation of an in vivo cellular environment is constructed on a lattice.

[left] A cell model in which 30% of the total volume is occupied by obstacles.

Particle diffuse around the stationary obstacles, reacting according to kinetic rates.

Scalability on GPUs

Calculation	FX5600			GTX280			
	Time (ms)	%	Performance†	Time (ms)	%	Performance†	Speedup
Load lattice block	5.2	20	13 GB/s	2.2	16	30 GB/s	2.4X
Random number generation‡	7.0	27	48 GOPS	3.8	28	88 GOPS	1.8X
Particle movement decision	7.7	29	109 GOPS	4.4	33	191 GOPS	1.8X
Particle propagation	3.6	13	94 GOPS	1.6	12	209 GOPS	2.2X
Store lattice block	2.9	11	23 GB/s	1.5	11	44 GB/s	1.9X
Total	26.4	100		13.5	100		1.9X

†Bandwidth rates were calculated as four bytes times the number of lattice sites being transferred divided by the runtime. Operation rates were calculated as the number of logical operations per site times the number of lattice sites divided by the runtime.

‡Operation count was calculated using 64-bit operations, but current hardware implements 64-bit operations using 32-bit instructions. Performance calculated using the 32-bit instruction count (88) yields 210 GIPS and 387 GIPS, respectively.

Roberts, Stone, Sepulveda, Hwu, Luthey-Schulten (2009) *The Eighth IEEE International Workshop on High-Performance Computational Biology, in press.*

Large Eddy Simulations (LES) on GPUs using CUDA

- 3D incompressible Navier-Stokes equations

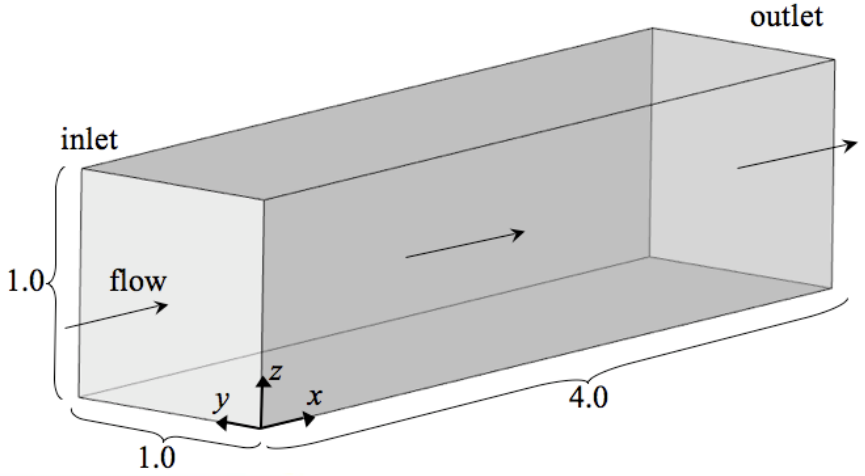
$$\nabla \cdot \mathbf{u} = 0$$

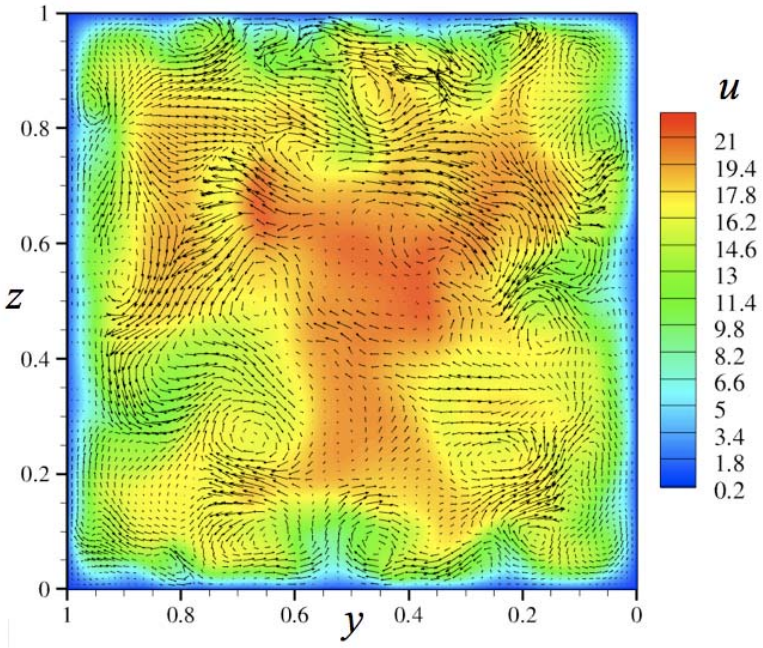
$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u}$$

- Solved using fractional-step method.
- Used geometric multigrid for pressure-Poisson solver.
- Used Smagorinsky sub-grid scale model.
- Used Red-Black Gauss-Seidel for linear solver.

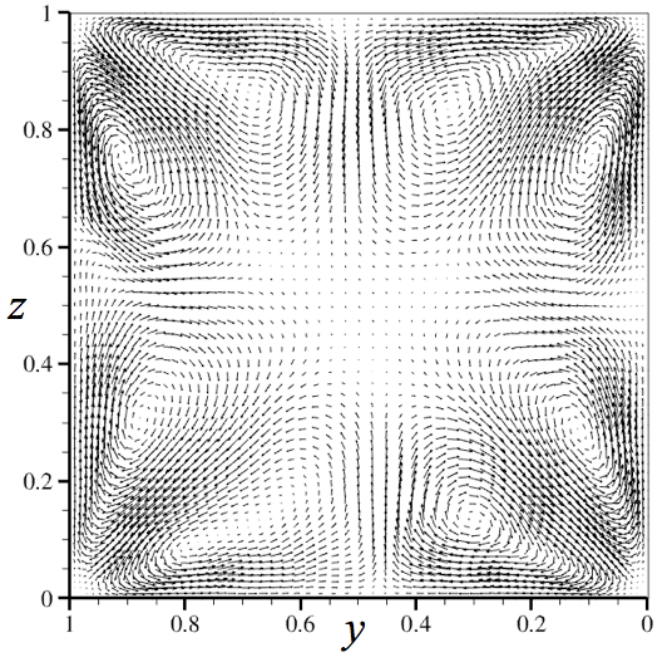
Aaron Shinn, Pratap Vanka, Wen-mei Hwu

Turbulent flow in 3D square duct





Instantaneous flow in cross-section



Mean flow in cross-section

GPU (Tesla C1060) versus 3.0 GHz Intel Xeon Core

First 100 time steps

laminar lid-driven cube simulation

mesh	Fortran code (sec)	CUDA code (sec)	speedup (CPU/GPU)
16x16x16	0.334	0.390	0.856
32x32x32	3.082	1.236	2.494
64x64x64	31.141	6.484	4.803
128x128x128	291.051	50.921	5.716

turbulent flow in a square duct

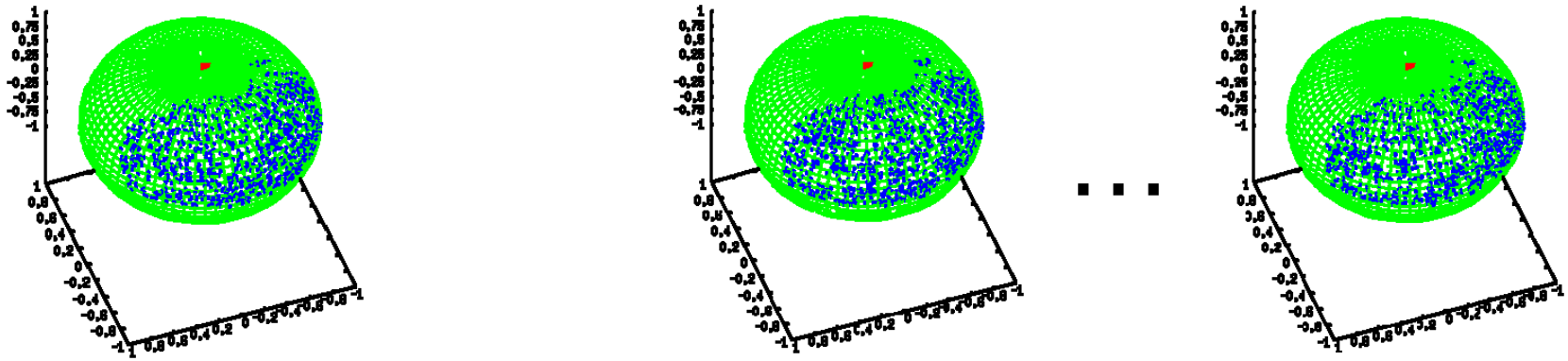
mesh	Fortran code (sec)	CUDA code (sec)	speedup (CPU/GPU)
256x64x64	208.088	34.555	6.022
512x64x64	448.463	64.283	6.976

Cosmological Data Analysis: Two Point Angular Correlation Function

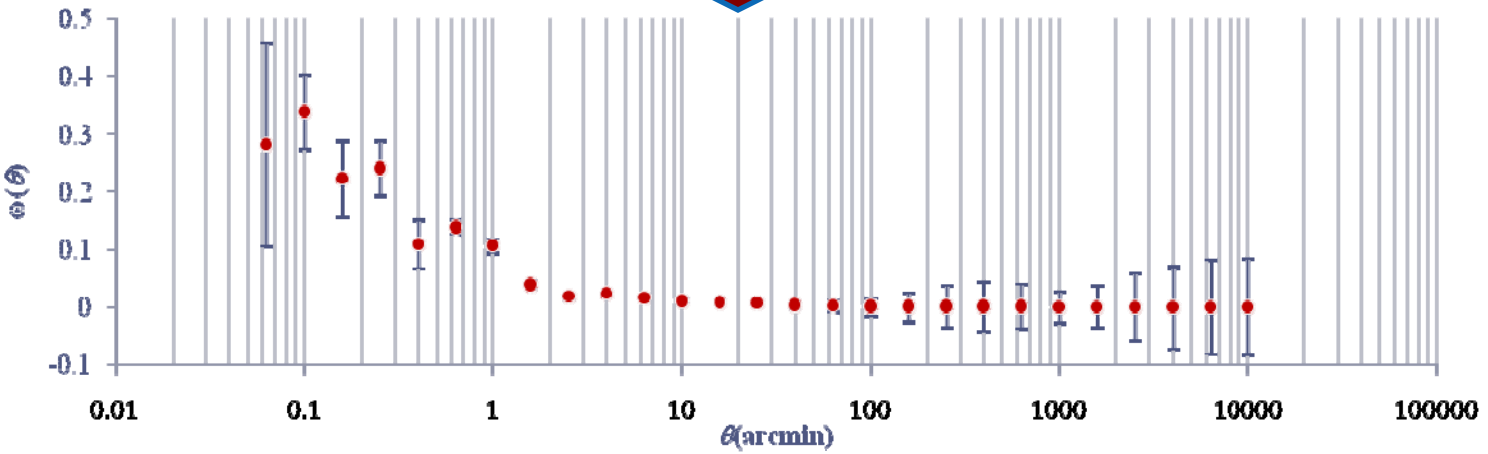
observed data

random dataset 1

random dataset n_R

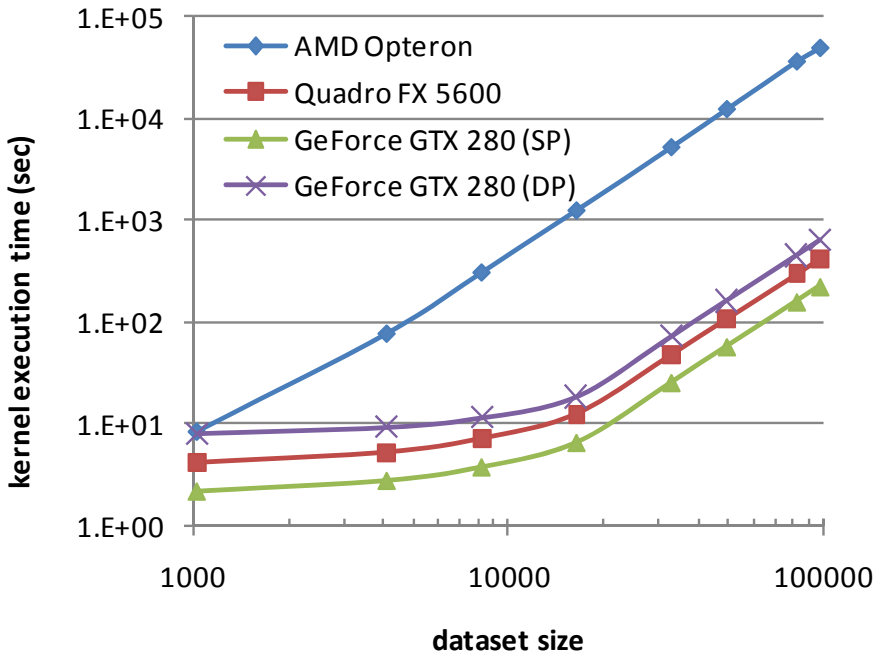


Two-point angular correlation function

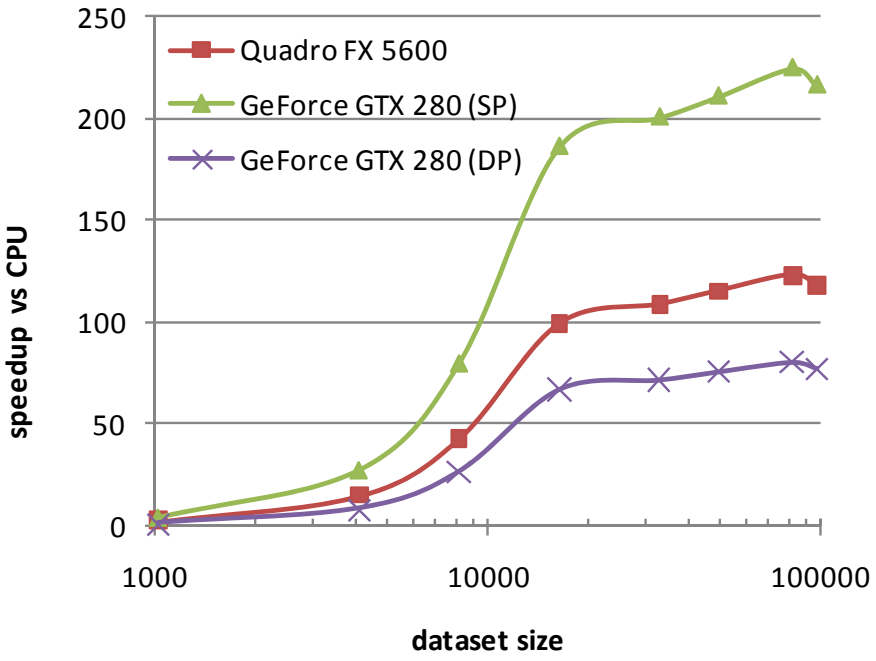


Performance on a Single GPU

Execution time



Speedup



Dylan Roeh, Volodymyr V. Kindratenko, Robert J. Brunner

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

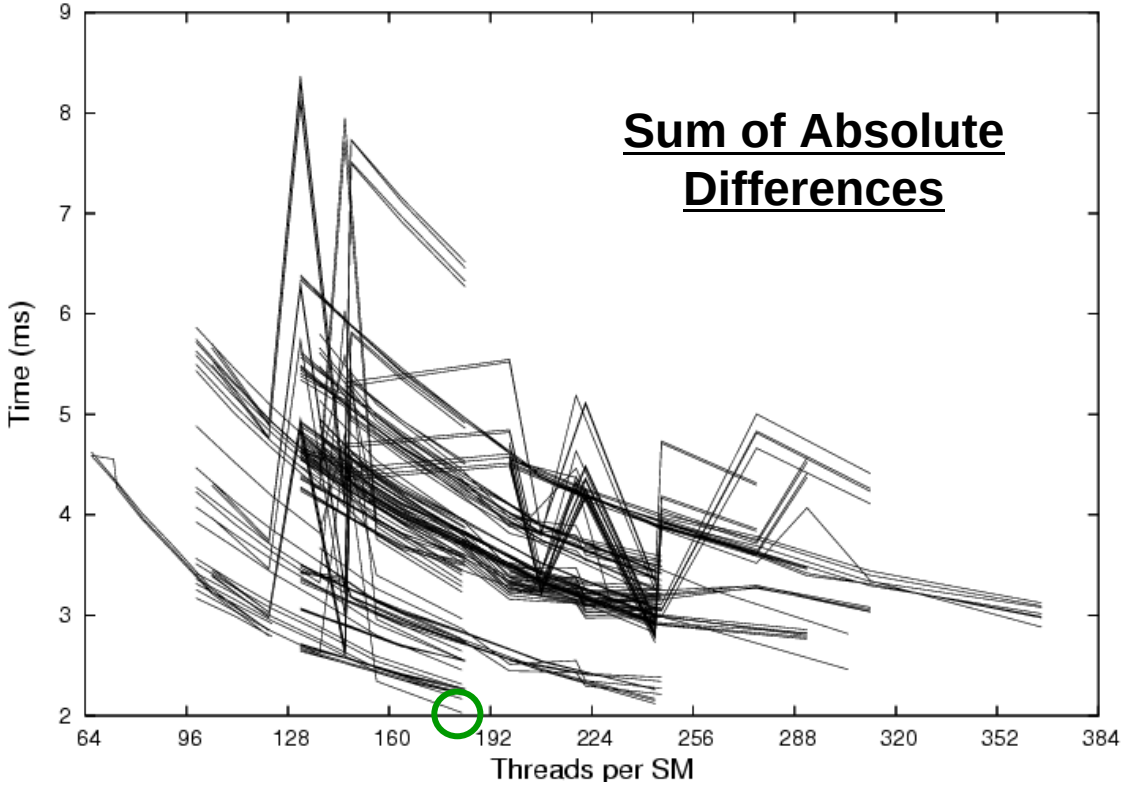
Tools are coming.

"There is always hope."

– In the eve of the Battle of the Helms Deep

Auto-tuning Tools do heavy lifting!

- Programmers are doing too much heavy lifting
- Too many memory organizational details are exposed to the programmers



- Pareto optimal curve based on analytical performance model of the source code allows automatic identification of optimal code arrangement

S. Ryoo, et al, "Program Optimization Space Pruning for a Multithreaded GPU, CGO, April 2008.

Implicitly parallel programming with data structure and function property annotations to enable auto parallelization

CUDA-auto

Locality annotation programming to eliminate need for explicit management of memory types and data transfers

CUDA-lite

Parameterized CUDA programming using auto-tuning and optimization space pruning

CUDA-tune

1st generation CUDA programming with explicit, hardwired thread organizations and explicit management of memory types and data transfers

MCUDA/ OpenMP

NVIDIA SDK 1.1

IA multi-core & Larrabe

NVIDIA GPU

Towards Systematic Code Reuse – MRI Example

- Two main approaches to MRI Reconstruction:
  - Riemann approximation of the continuous inverse FT:

$$\hat{\boldsymbol{\rho}} = \mathbf{F}^{H}\left(\mathbf{w} \odot \mathbf{d}\right) = \sum_{m=1}^{M} w_m d\left[m\right] e^{i2\pi \mathbf{k}_m \cdot \mathbf{x}}$$

  - Solve a regularized inverse problem, e.g.,

$$\hat{\boldsymbol{\rho}} = \arg\min_{\boldsymbol{\rho}} \left\| \mathbf{F}\boldsymbol{\rho} - \mathbf{d} \right\|_2^2 + R\left(\boldsymbol{\rho}\right)$$

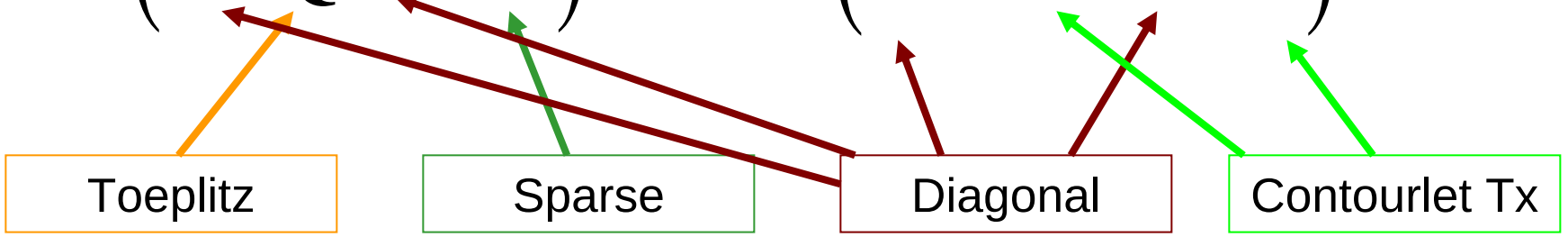
    Solutions often derived by solving one or more matrix inversions, e.g.,

$$\hat{\boldsymbol{\rho}} = \left(\mathbf{F}^{H}\mathbf{F} + \mathbf{H}\right)^{-1}\mathbf{F}^{H}\mathbf{d}$$

Towards a Reusable Code Base

- Certain computations appear frequently in inverse problems, e.g.,
  - Matrix inversion
  - Matrix multiplication
- It is rare that two similar applications have exactly the same structure

$$\left(\mathbf{S}^H\mathbf{Q}\mathbf{S}+\mathbf{H}\right)^{-1}\mathbf{d} \qquad \left(\mathbf{I}+\mathbf{W}^H\mathbf{D}\mathbf{W}\right)^{-1}\mathbf{d}$$



- Better library interfaces and tools are needed to allow for easy code reuse across different applications.

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Education is key.

To Learn More

- UIUC ECE498AL – Programming Massively Parallel Processors (http://courses.ece.uiuc.edu/ece498/al/)
  - David Kirk (NVIDIA) and Wen-mei Hwu (UIUC) co-instructors
  - CUDA programming, GPU computing, lab exercises, and projects
  - Lecture slides and voice recordings
- More than 500 students worldwide follow the course each semester.

Graduate Summer School

- Aimed at CSE grad students
  - Full registration
  - 180 applicants from 3 continents
  - 44 accepted from 25 universities, 3 continents
  - 50+ remote participants
  - 9 lectures, 3 keynotes, 1 panel, hands-on lab
- Sponsored by UIUC, NCSA, Microsoft, NVIDIA, VSCSE
- Will be offered again, August 10-14 2009

www.greatlakesconsortium.org/events/GPUMulticore

Wen-mei Hwu and David Kirk Co-instructor

Conclusion - A Great Opportunity for Many

- Many-core accelerates science discovery
  - Current challenges are in parallelizing sparse and graph- based computation for very large models – memory bandwidth!
  - Incorporating many-core technology into large scale systems such as Blue Waters presents major challenges in software tool chains
- Software engineering issues must be better addressed
  - It is still hard to extract and integrate parallel kernels from real applications
  - Effective code reuse requires a very large library code base and new tools and interfaces.
- Cross-application fertilization key to future success – vision for a global developer community

Thank you! Any questions?