

Two vertical lines, one blue and one orange, are positioned on the left side of the slide.

CS/EE 217

GPU Architecture and Parallel Programming

Project Kickoff

Two flavors

- Application
 - Implement/optimize an realistic application on GPGPUs
- Architecture
 - Evaluate application performance at the architecture level and sensitivity to architectural features
 - Propose new features or algorithms

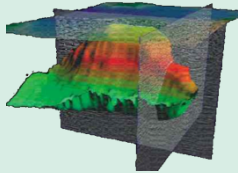
Two tracks

- Basic project—essentially a larger lab.
 - Should involve significant coding/optimization but...
 - does not necessarily have to be original or research related work
 - Could just implement one of the default projects
- Research project
 - Should review related work
 - Should have a goal of publishing a paper
 - Even if you don't get there, that is ok
 - More formal project proposal
 - In return, you don't have to take the final exam!

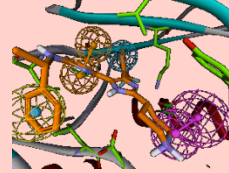
Application projects: Objective

- To Build up your ability to translate parallel computing power into science and engineering breakthroughs
 - Identify applications whose computing structures are suitable for massively parallel execution
 - 10-100X more computing power can have transformative effect on these applications
 - Much better if you have a client
 - You can be the client – bringing an idea from your own research is encouraged.
- Develop algorithm patterns that can result in both better efficiency as well as scalability

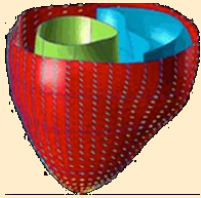
Future Science and Engineering Breakthroughs Hinge on Computing



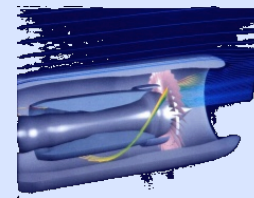
**Computational
Geoscience**



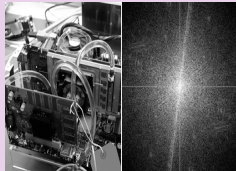
**Computational
Chemistry**



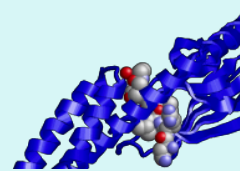
**Computational
Medicine**



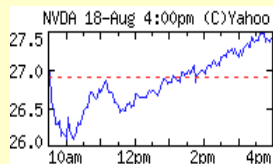
**Computational
Modeling**



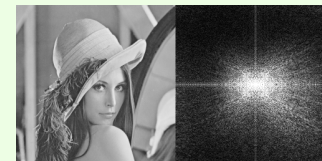
**Computational
Physics**



**Computational
Biology**

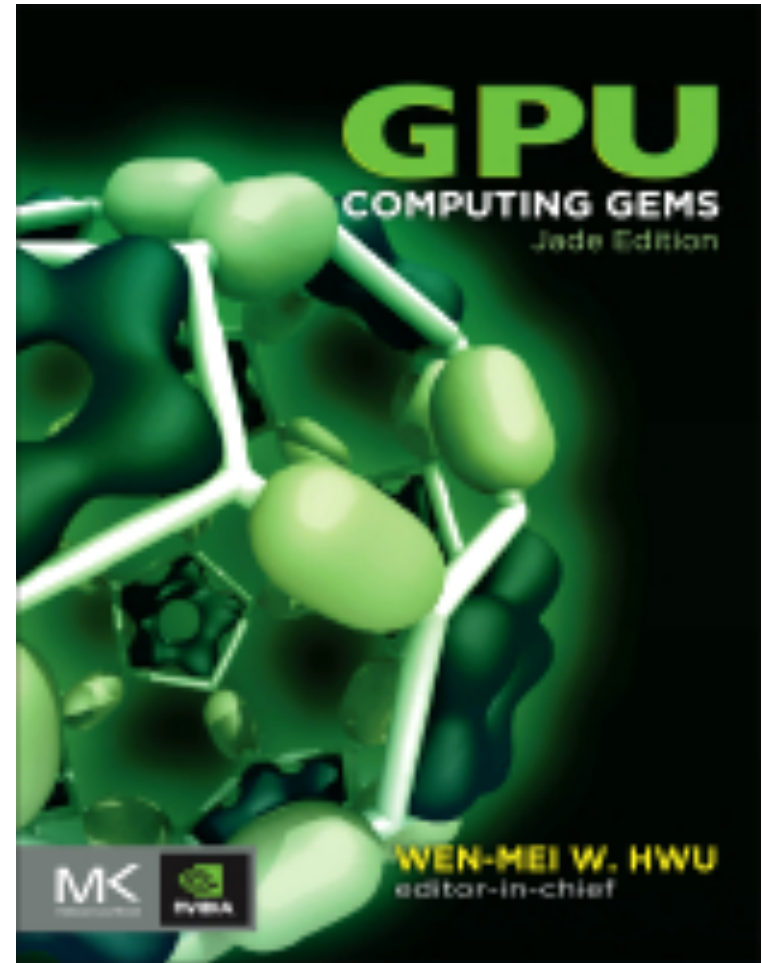
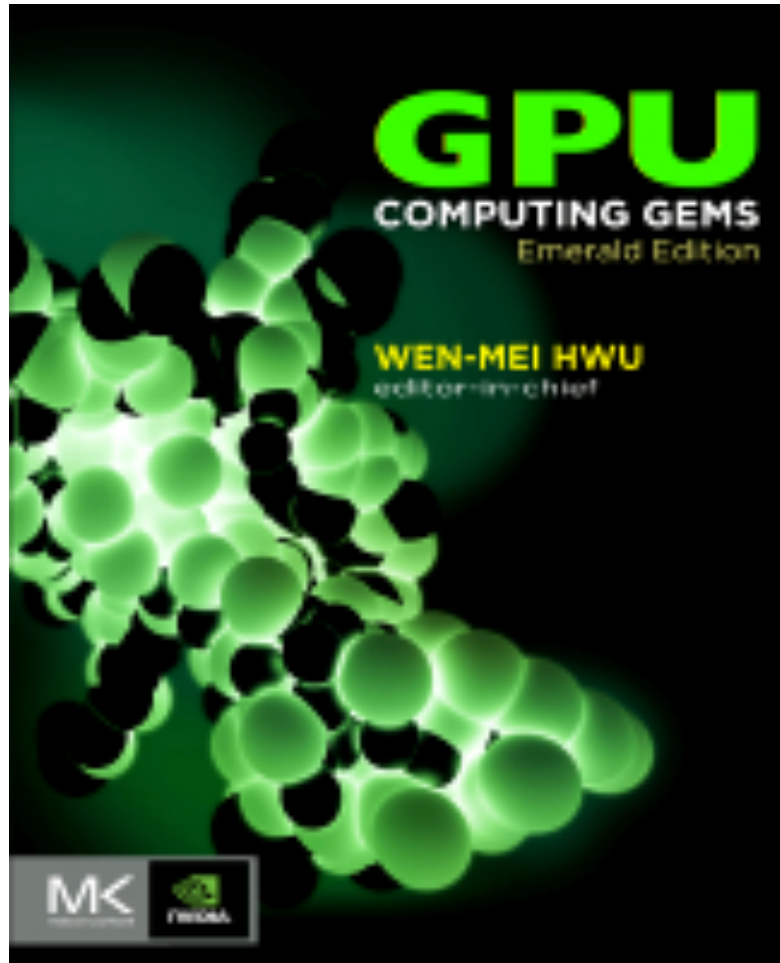


**Computational
Finance**



**Image
Processing**

GPU computing is catching on.



GPU computing is catching on.

**Financial
Analysis**

**Scientific
Simulation**

**Engineering
Simulation**

**Data
Intensive
Analytics**

**Medical
Imaging**

**Digital
Audio
Processing**

**Digital Video
Processing**

**Computer
Vision**

**Biomedical
Informatics**

**Electronic
Design
Automation**

**Statistical
Modeling**

**Ray Tracing
Rendering**

**Interactive
Physics**

**Numerical
Methods**

- 280 submissions to GPU Computing Gems
~90 articles included in two volumes

Faster is not “just Faster”

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

How much computing power is enough?

- Each jump in computing power motivates new ways of computing
 - Many apps have approximations or omissions that arose from limitations in computing power
 - Every 10x jump in performance allows app developers to innovate
 - Example: graphics, medical imaging, physics simulation, etc.

Application developers do not take it seriously until they see real results.

Why didn't this happen earlier?

- Computational experimentation is just reaching critical mass
 - Simulate large enough systems
 - Simulate long enough system time
 - Simulate enough details
- Computational instrumentation is also just reaching critical mass
 - Reaching high enough accuracy
 - Cover enough observations

A Great Opportunity for Many

- New massively parallel computing is enabling
 - Drastic reduction in “time to discovery”
 - 1st principle-based simulation at meaningful scale
 - New, 3rd paradigm for research: computational experimentation
- The “democratization” of power to discover
 - \$2,000/Teraflop in personal computers today
 - \$5,000,000/Petaflops in clusters today
 - HW cost will no longer be the main barrier for big science
- This is once-in-a-career opportunity for many!

VMD/NAMD Molecular Dynamics

- 240X speedup over sequential code
- Computational biology

THEORETICAL and COMPUTATIONAL BIOPHYSICS GROUP
NEW RESOURCES FOR MACROMOLECULAR MODELING AND RECONSTRUCTION
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Home
Overview
Publications
Research
Software
VMD Molecular Graphics Viewer
NAMD Molecular Dynamics Simulator
Hardware Collaboration Environment
MD Service Center
Structural Biology Software Database
Course/Abstract Facility
Outreach

Download NAMD

Download VMD

Parallel Programming Laboratory

NAMD

Scalable Molecular Dynamics

NAMD, recipient of a 2002 Gordon Bell Award, is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems. Used on **Charm++ parallel objects**, NAMD **scales** to hundreds of processors on high-end parallel platforms and tens of processors on commodity **clusters** using gigabit ethernet. NAMD uses the popular molecular graphics program **VMD** for simulation setup and trajectory analysis, but is also file-compatible with AMBER, CHARMM, and X-PLOR. NAMD is distributed **free of charge** with source code. You can **build NAMD** yourself or download **binaries** for a wide variety of platforms. Our **tutorials** show you how to use NAMD and VMD for biomolecular modeling.

High performance computing in biology: Multimillion atom simulations of nanoscale systems. K.Y. Sanbonmatsu and C.-S. Tung. *Journal of Structural Biology*, 157:370-400, 2007.

Supercomputer Simulations May Pinpoint Causes of Parkinson's, Alzheimer's Diseases (SDSC article referring to NAMD simulations on Blue Gene/L reported in *Tsigelny et al., FEBS Journal*, 274:1862-1877, 2007.)

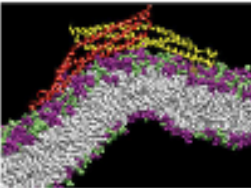
Single search:

Spotlight: Step Up to the BAR Domain (Apr 2007) Other Spotlights

PSC News Release: University of Utah chemist Gregory Voth and grad student Phil Blood are using PSC's Cray XT3 to tackle a basic question of embryology—the life-changing process by which cells absorb material from outside the cell by bending their membrane to form a "vesicle" and engulf it. All animal cells depend on endocytosis, which involves various steps, but begins with curvature of the membrane.

BAR domains are a family of banana-shaped proteins shown to bind to cellular membrane as it curves. Experiments suggest that BAR domains mold their concave surface to a section of membrane and induce a corresponding curvature. Voth and Blood undertook molecular dynamics simulations to look more closely. **With the XT3 they've been able to run efficiently, using software called NAMD, with as many as 1,024 processors.** "The XT3 has been amazing," says Blood. "We haven't found a hard limit on scaling up the number of processors."

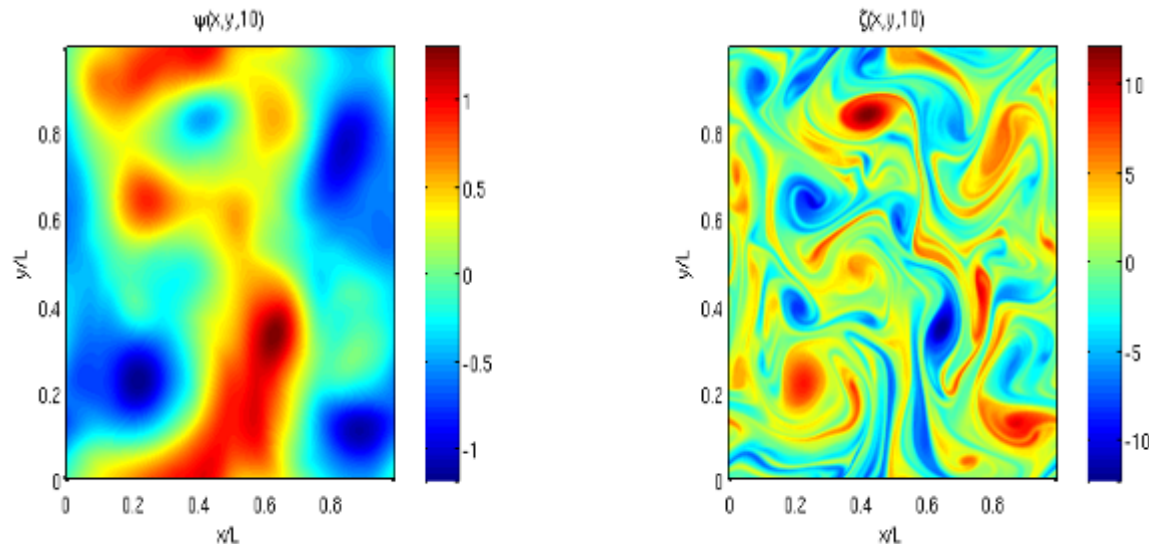
They used TeraGrid systems at SDSC, NCSA and University of Chicago/Argonne to construct a model and to explore how long a stretch of membrane they needed for curvature to occur. Their final simulations used the XT3 to include the protein with a 50-nanometer length of membrane—probably the longest patch of



Matlab: Language of Science

15X with MATLAB CPU+GPU

http://developer.nvidia.com/object/matlab_cuda.html



Pseudo-spectral simulation of 2D Isotropic turbulence

Prevalent Performance Limits

Some microarchitectural limits appear repeatedly across the benchmark suite:

- Global memory bandwidth saturation
 - Tasks with intrinsically low data reuse, e.g. vector-scalar addition or matrix-vector multiplication product
 - Computation with frequent global synchronization
 - Converted to short-lived kernels with low data reuse
 - Common in simulation programs
- Thread-level optimization vs. latency tolerance
 - Since hardware resources are divided among threads, low per-thread resource use is necessary to furnish enough simultaneously-active threads to tolerate long-latency operations
 - Making individual threads faster generally increases register and/or shared memory requirements
 - Optimizations trade off single-thread speed for exposed latency

What you will likely need to hit hard.

- Parallelism extraction requires global understanding
 - Most programmers only understand parts of an application
- Algorithms need to be re-designed
 - Algorithmic effect on parallelism and locality is often hard to maneuver
- Real but rare dependencies often need to be pushed aside
 - Error checking code, etc., parallel code is often not equivalent to sequential code
- Getting more than a small speedup over sequential code is very tricky
 - ~20 versions typically experimented for each application

Ideas for projects

- Your own research!
- An application you are interested in.
- GPU computing Gems
- Parboil benchmark: Collections of best parallel programming projects for CPU/GPU computing

<http://impact.crhc.illinois.edu/Parboil/parboil.aspx>

- Rodinia Benchmark
- Talk to me if you are interested in architecture projects

Architecture Proposals

- Either a workload evaluation study running workloads in a simulator
 - Explore impact of different architecture parameters on performance
 - Provide insights into what causes the differences in behavior
 - Propose and evaluate improvements
 - E.g., warp scheduler competition
- <http://www.sigarch.org/2015/08/06/call-for-papers-1st-gpu-warpwavefront-scheduling-championship/>

Project website come up soon

- Deadlines and details of deliverables
- Standard programming project
- Standard architecture project (scheduler competition)

Two vertical lines, one blue and one orange, are positioned on the left side of the slide.

ANY MORE QUESTIONS?