Accelerating Molecular Modeling Applications with GPU Computing

John Stone

Theoretical and Computational Biophysics Group

Beckman Institute for Advanced Science and Technology

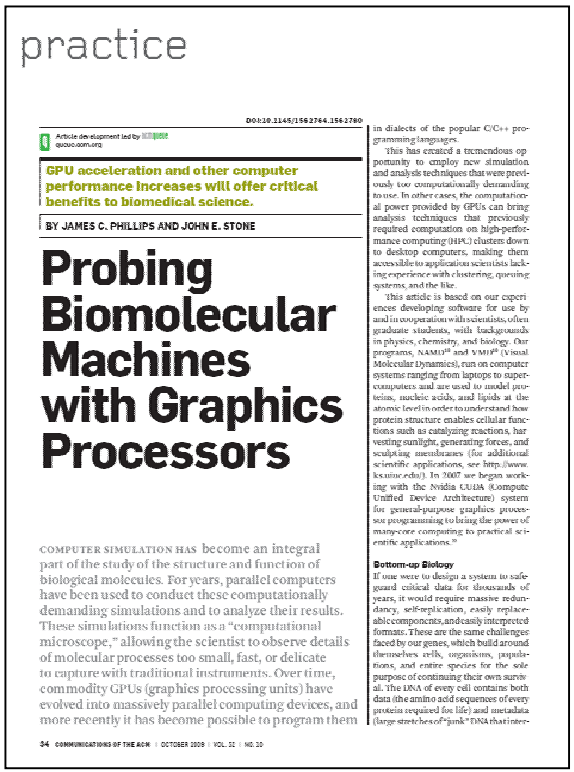
University of Illinois at Urbana-Champaign

http://www.ks.uiuc.edu/Research/gpu/

Supercomputing 2009

Portland, OR, November 18, 2009

See our GPU article in the CACM issue included in your SC2009 attendee bag!



practice

DOI:10.1145/1562764.1562780

Article development led by queue.acm.org

GPU acceleration and other computer performance increases will offer critical benefits to biomedical science.

BY JAMES C. PHILLIPS AND JOHN E. STONE

Probing Biomolecular Machines with Graphics Processors

COMPUTER SIMULATION HAS become an integral part of the study of the structure and function of biological molecules. For years, parallel computers have been used to conduct these computationally demanding simulations and to analyze their results. These simulations function as a "computational microscope," allowing the scientist to observe details of molecular processes too small, fast, or delicate to capture with traditional instruments. Over time, commodity GPUs (graphics processing units) have evolved into massively parallel computing devices, and more recently it has become possible to program them in dialects of the popular C/C++ programming languages.

This has created a tremendous opportunity to employ new simulation and analysis techniques that were previously too computationally demanding to use. In other cases, the computational power provided by GPUs can bring analysis techniques that previously required computation on high-performance computing (HPC) clusters down to desktop computers, making them accessible to application scientists lacking experience with clustering, queuing systems, and the like.

This article is based on our experiences developing software for use by and in cooperation with scientists, often graduate students, with backgrounds in physics, chemistry, and biology. Our programs, NAMD[1] and VMD[2] (Visual Molecular Dynamics), run on computer systems ranging from laptops to supercomputers and are used to model proteins, nucleic acids, and lipids at the atomic level in order to understand how protein structure enables cellular functions such as catalyzing reactions, harvesting sunlight, generating forces, and sculpting membranes (for additional scientific applications, see http://www.ks.uiuc.edu/). In 2007 we began working with the Nvidia CUDA (Compute Unified Device Architecture) system for general-purpose graphics processor programming to bring the power of many-core computing to practical scientific applications.[3]

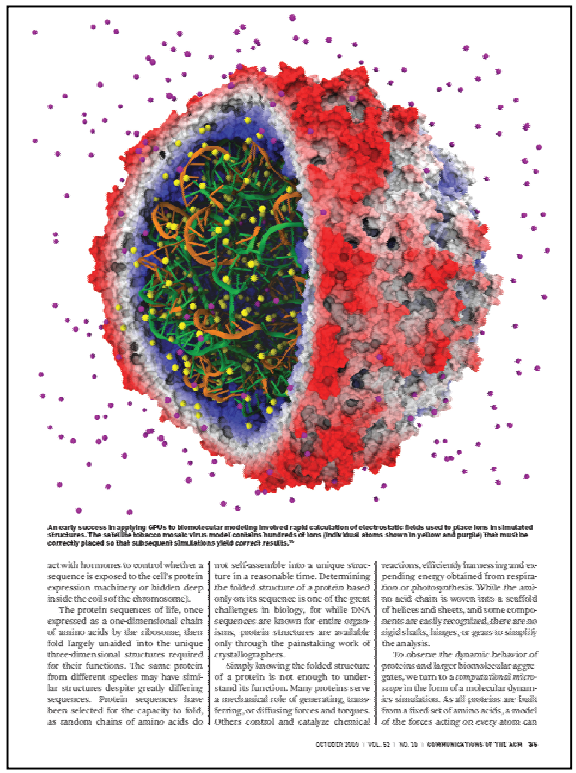
Bottom-up Biology

If one were to design a system to safeguard critical data for thousands of years, it would require massive redundancy, self-replication, easily replaceable components, and easily interpreted formats. These are the same challenges faced by our genes, which build around themselves cells, organisms, populations, and entire species for the sole purpose of continuing their own survival. The DNA of every cell contains both data (the amino acid sequences of every protein required for life) and metadata (large stretches of "junk" DNA that interact with hormones to control whether a sequence is exposed to the cell's protein expression machinery or hidden deep inside the coil of the chromosome).

The protein sequences of life, once expressed as a one-dimensional chain of amino acids by the ribosome, then fold largely unaided into the unique three-dimensional structures required for their functions. The same protein from different species may have similar structures despite greatly differing sequences. Protein sequences have been selected for the capacity to fold, as random chains of amino acids do not self-assemble into a unique structure in a reasonable time. Determining the folded structure of a protein based only on its sequence is one of the great challenges in biology, for while DNA sequences are known for entire organisms, protein structures are available only through the painstaking work of crystallographers.

Simply knowing the folded structure of a protein is not enough to understand its function. Many proteins serve a mechanical role of generating, transferring, or diffusing forces and torques. Others control and catalyze chemical reactions, efficiently harvesting and expending energy obtained from respiration or photosynthesis. While the amino acid chain is woven into a scaffold of helices and sheets, and some components are easily recognized, there are no rigid blocks, hinges, or gears to simplify the analysis.

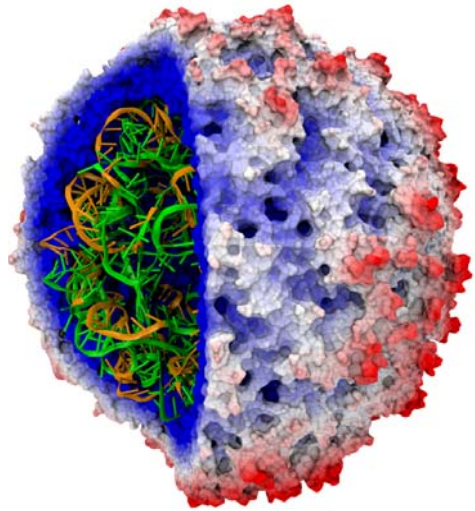
To observe the dynamic behavior of proteins and larger biomolecular aggregates, we turn to a computational microscope in the form of a molecular dynamics simulation. As all proteins are built from a fixed set of amino acids, a model of the forces acting on every atom can



An early success in applying GPUs to biomolecular modeling involved rapid calculation of electrostatic fields used to place ions in simulated structures. The satellite tobacco mosaic virus model contains hundreds of ions (individual atoms shown in yellow and purple) that must be correctly placed so that subsequent simulations yield correct results.[5]

Phillips & Stone, Communications of the ACM, 52(10):34-41, Oct 2009

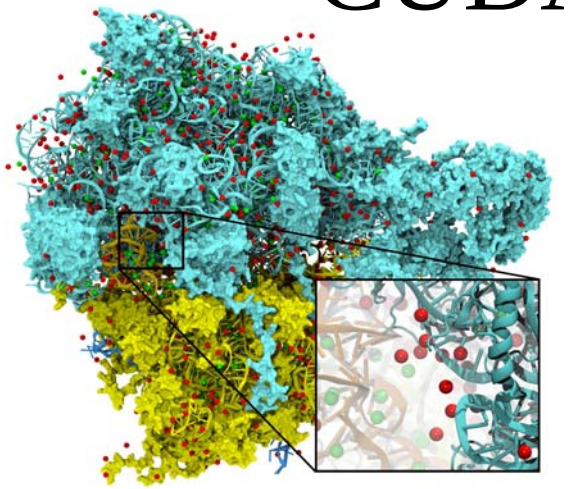
VMD – "Visual Molecular Dynamics"



- Visualization and analysis of:
  - Molecular dynamics simulations
  - Sequence data
  - Volumetric data
  - Quantum chemistry simulations,
  - And much more…
- User extensible with scripting and plugins
- Over 9,400 users running CUDA-enabled versions of VMD
- http://www.ks.uiuc.edu/Research/vmd/

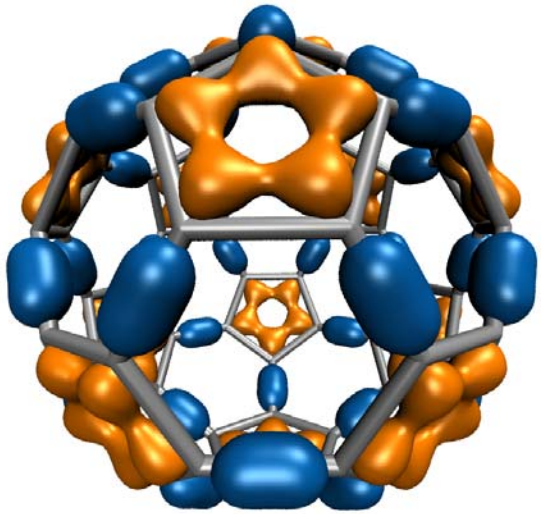
- GPU acceleration provides an opportunity to make some **slow, or batch** calculations capable of being run **interactively, or on-demand…**

CUDA Acceleration in VMD



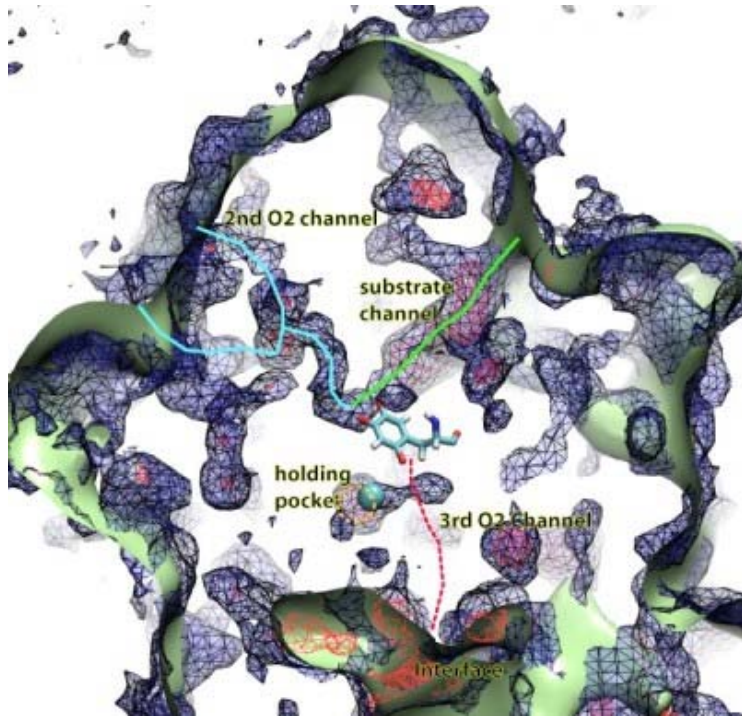
Electrostatic field calculation, ion placement

20x to 44x faster



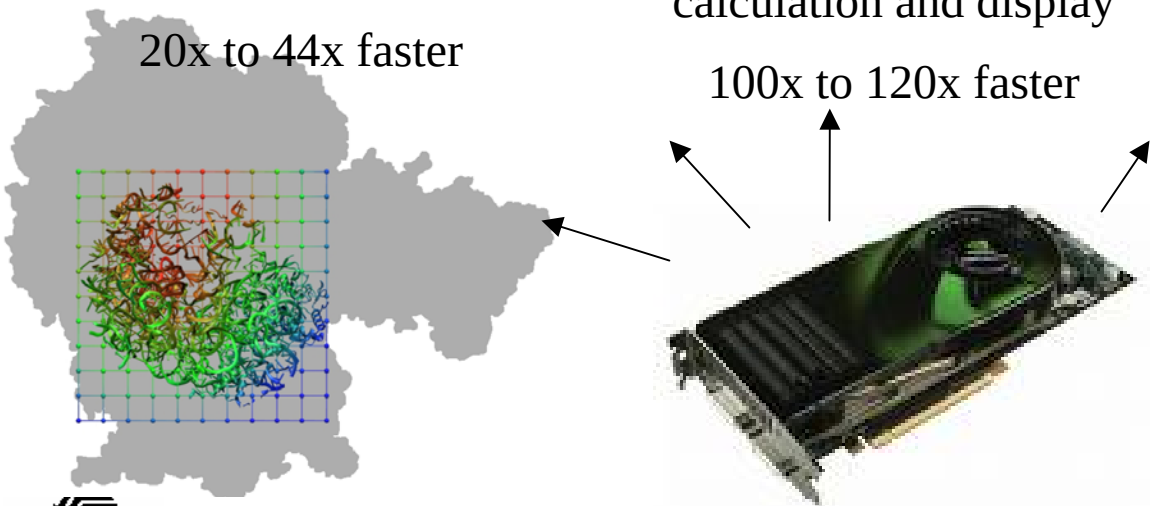
Molecular orbital calculation and display

100x to 120x faster



Imaging of gas migration pathways in proteins with implicit ligand sampling

20x to 30x faster

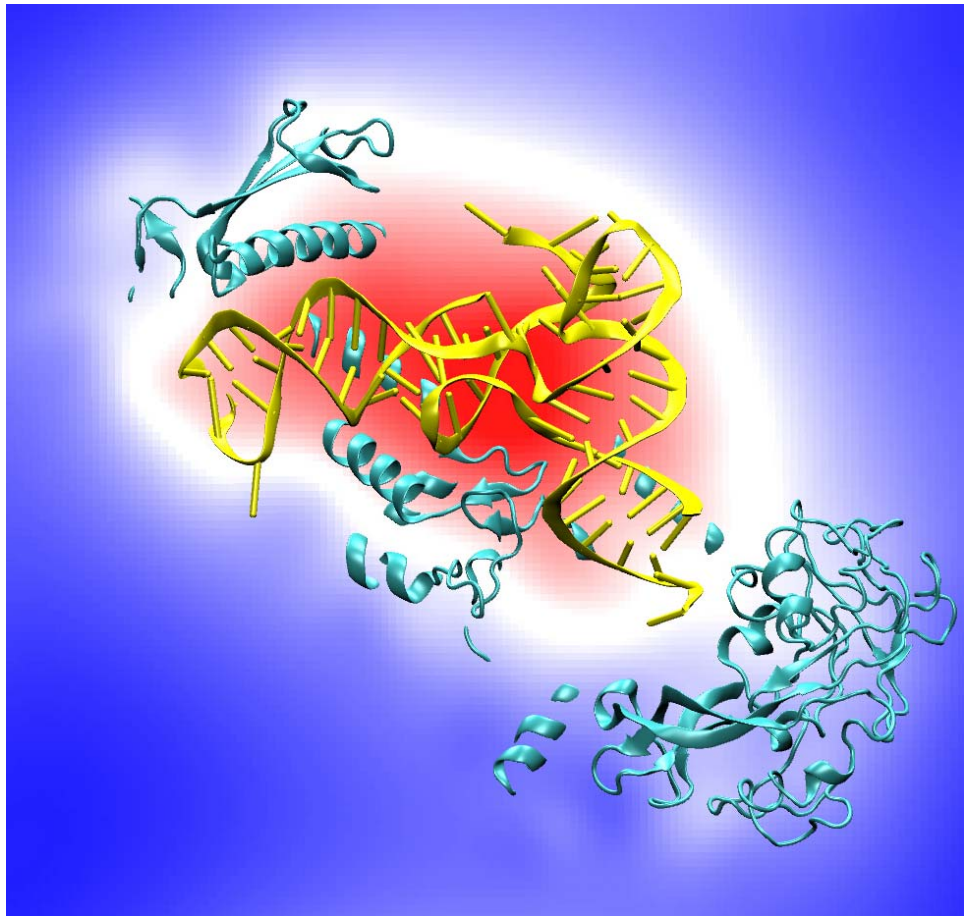


Electrostatic Potential Maps

- Electrostatic potentials evaluated on 3-D lattice:

$$V_i = \sum_j \frac{q_j}{4\pi\epsilon_0 |\mathbf{r}_j - \mathbf{r}_i|}$$

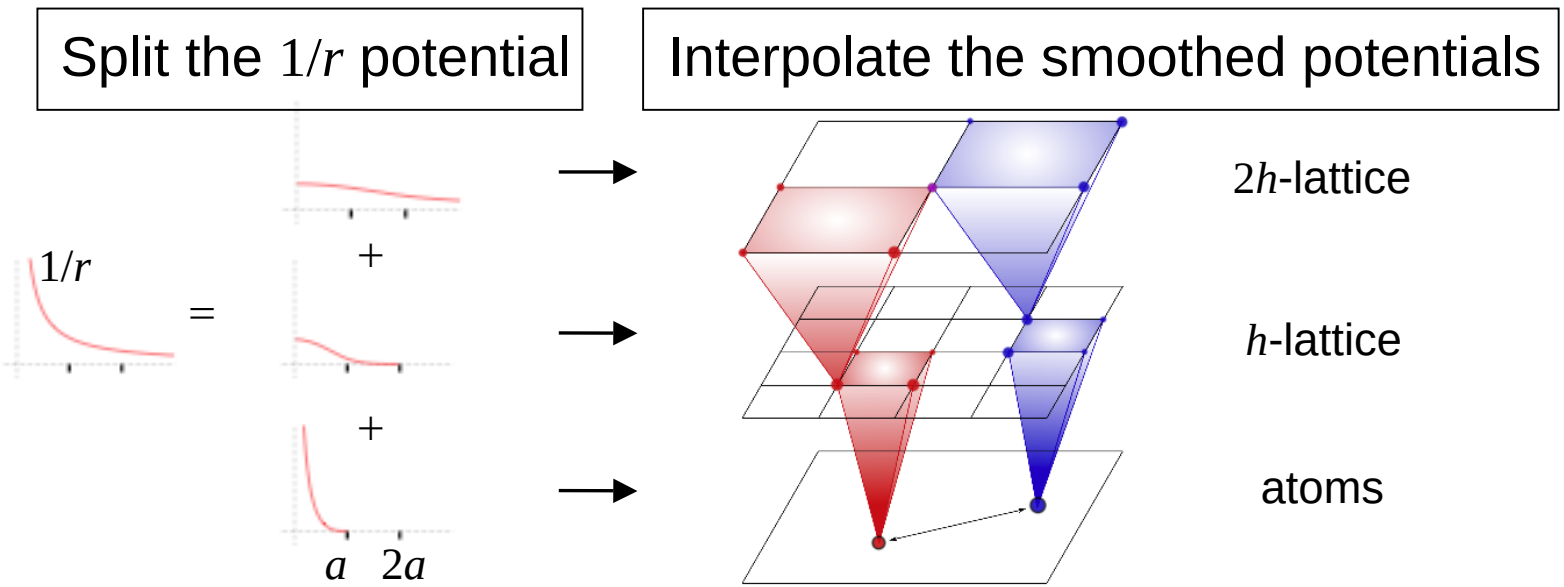
- Applications include:
  - Ion placement for structure building
  - Time-averaged potentials for simulation
  - Visualization and analysis



Isoleucine tRNA synthetase

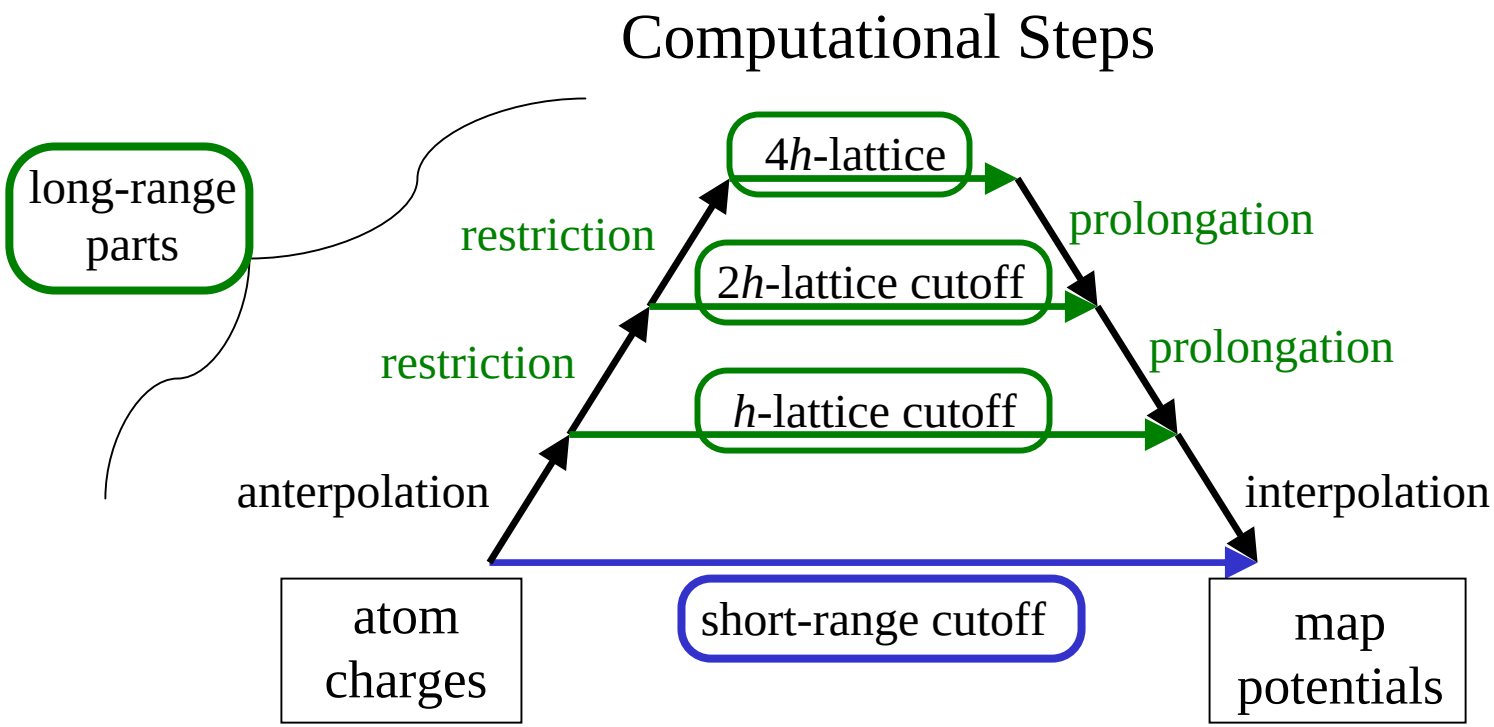
Multilevel Summation Main Ideas

- Split the 1/*r* potential into a short-range cutoff part plus smoothed parts that are successively more slowly varying. All but the top level potential are cut off.
- Smoothed potentials are interpolated from successively coarser lattices.
- Finest lattice spacing *h* and smallest cutoff distance *a* are doubled at each successive level.



Multilevel Summation Calculation

map potential = **exact short-range interactions** + **interpolated long-range interactions**

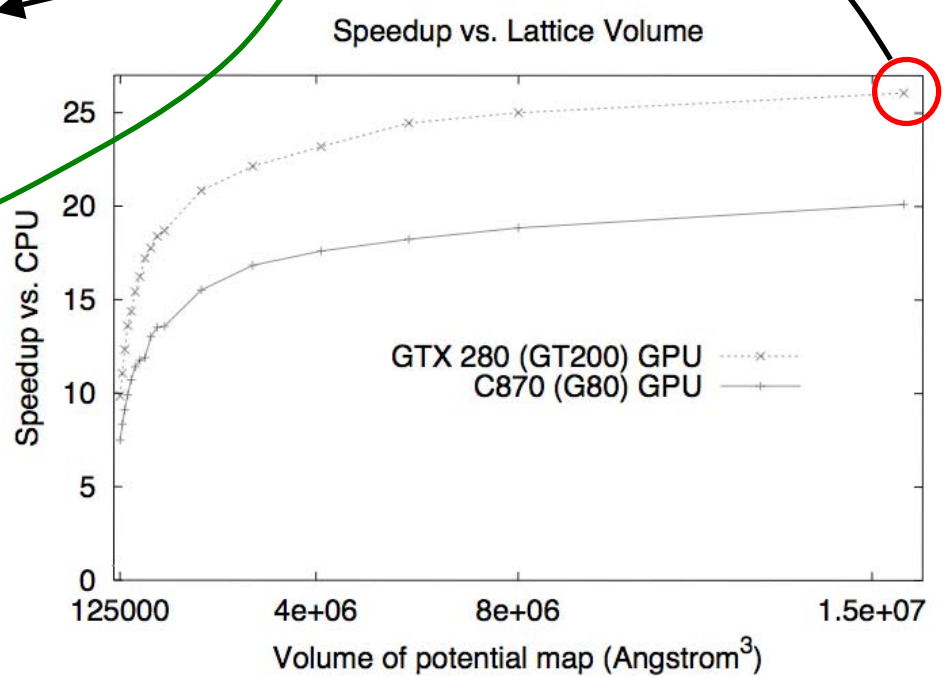


Multilevel Summation on the GPU

Accelerate **short-range cutoff** and **lattice cutoff** parts

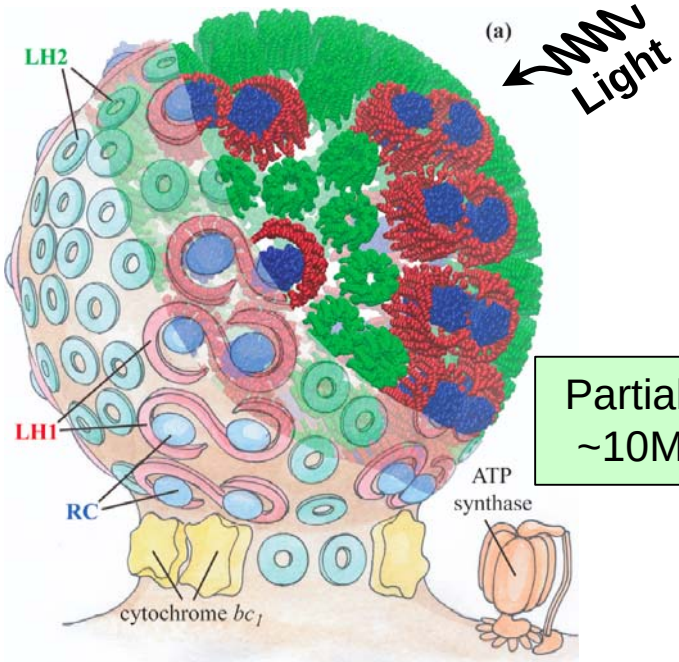
Performance profile for 0.5 Å map of potential for 1.5 M atoms.
Hardware platform is Intel QX6700 CPU and NVIDIA GTX 280.

Computational steps	CPU (s)	w/ GPU (s)	Speedup
Short-range cutoff	480.07	14.87	32.3
Long-range anterpolation	0.18		
restriction	0.16		
lattice cutoff	49.47	1.36	36.4
prolongation	0.17		
interpolation	3.47		
Total	533.52	20.21	26.4

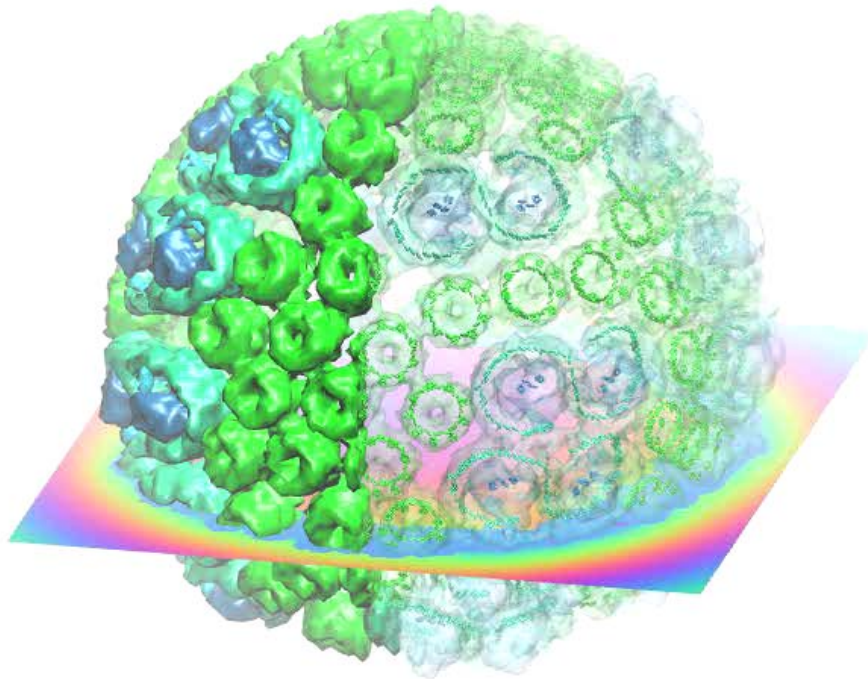


Photobiology of Vision and Photosynthesis

Investigations of the chromatophore, a photosynthetic organelle



Partial model:
~10M atoms



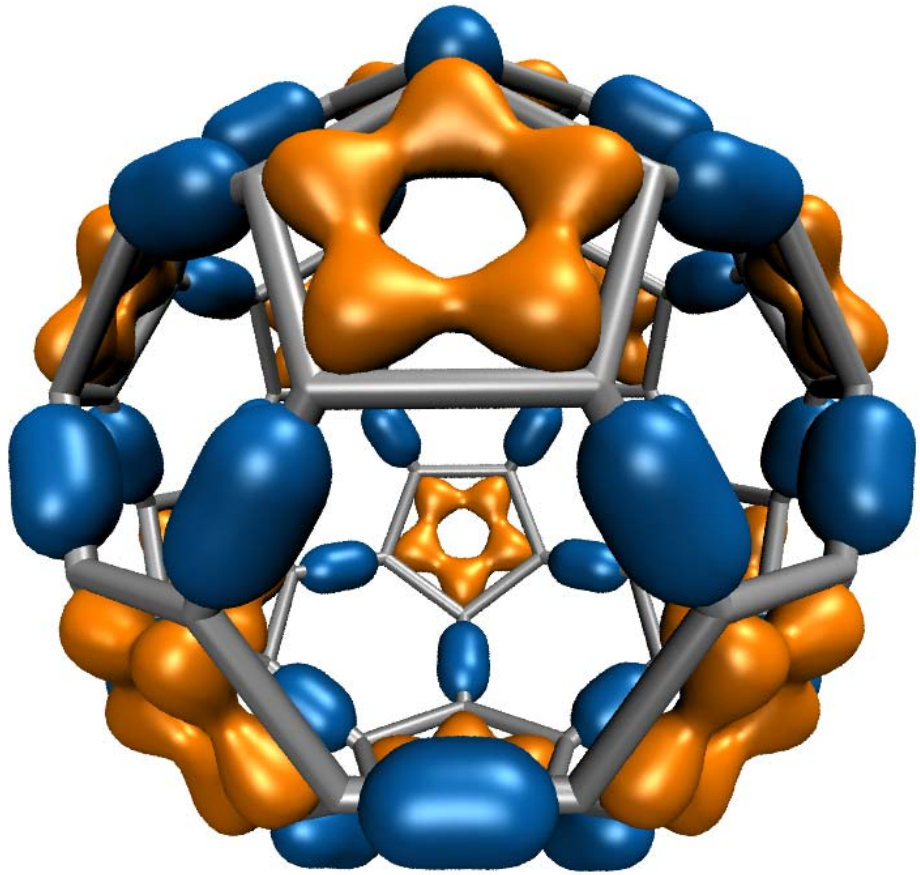
Electrostatics needed to build full structural model, place ions, study macroscopic properties

Electrostatic field of chromatophore model from multilevel summation method: computed with 3 GPUs (G80) in ~90 seconds, 46x faster than single CPU core

Full chromatophore model will permit structural, chemical and kinetic investigations at a structural systems biology level

Computing Molecular Orbitals

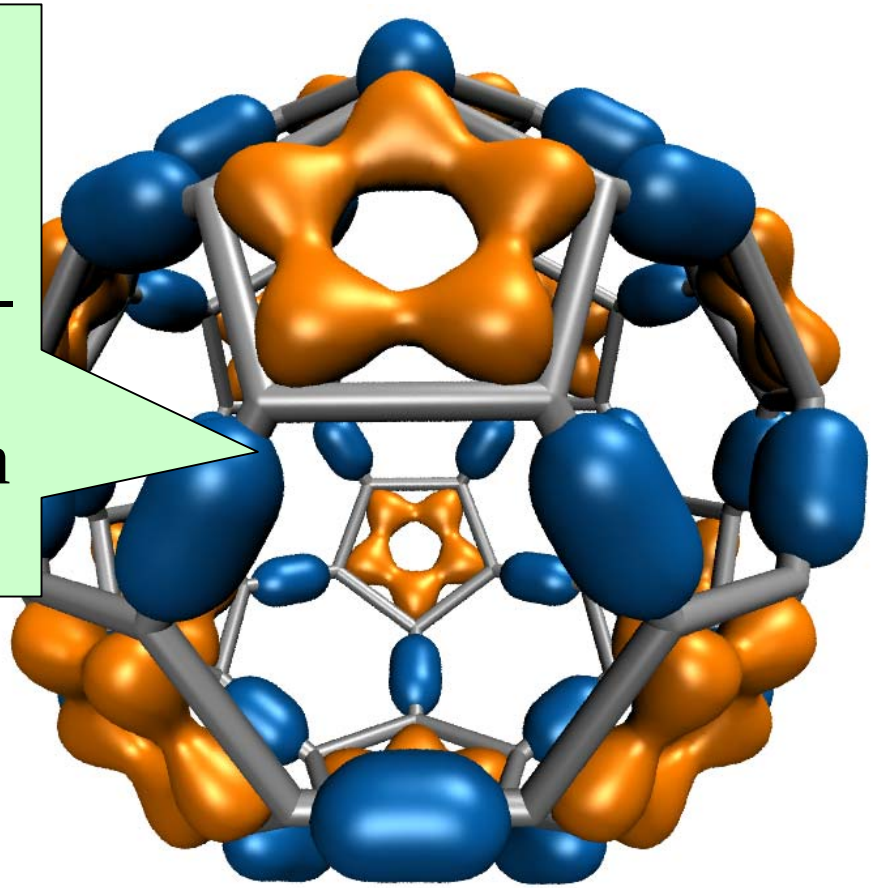
- Visualization of MOs aids in understanding the chemistry of molecular system
- MO spatial distribution is correlated with electron probability density
- Calculation of high resolution MO grids can require tens to hundreds of seconds on CPUs
- >100x speedup allows interactive animation of MOs @ 10 FPS



C_{60}

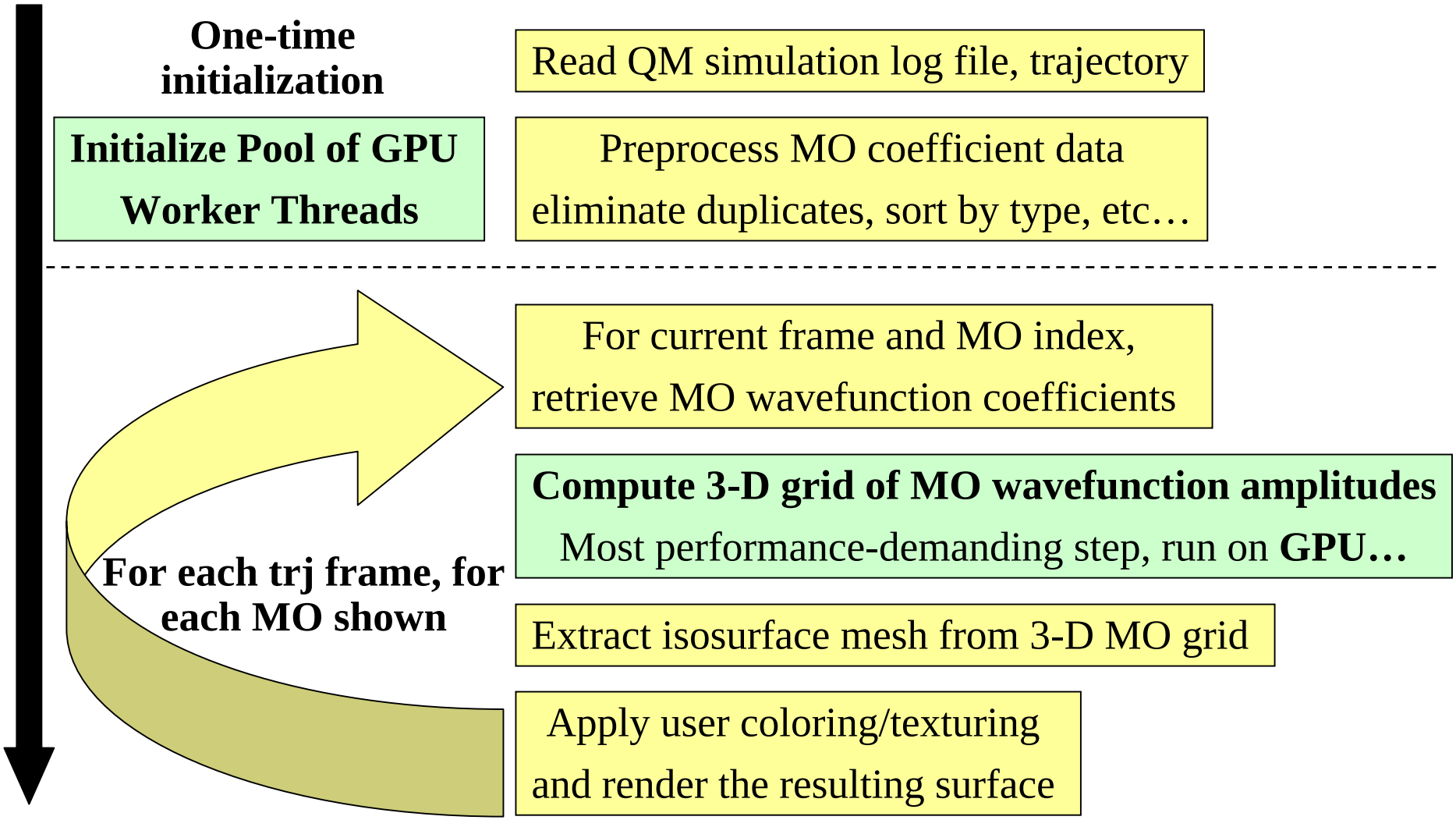
Computing Molecular Orbitals

See a live demo of VMD GPU-accelerated MO calc./display, showing the results of a TeraChem GPU-accelerated QM calculation in the Stanford SLAC booth here at SC2009!



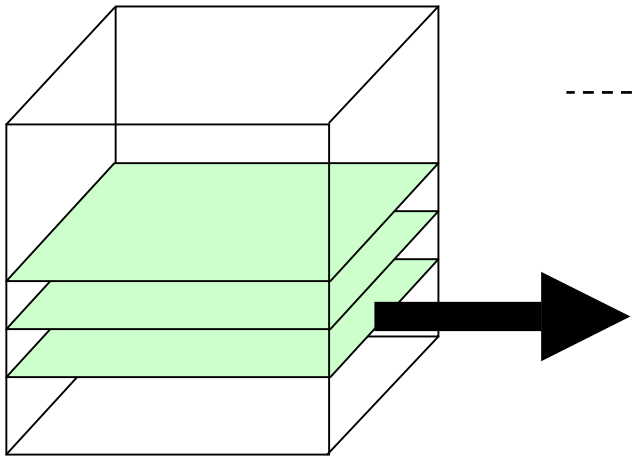
C_{60}

Molecular Orbital Computation and Display Process

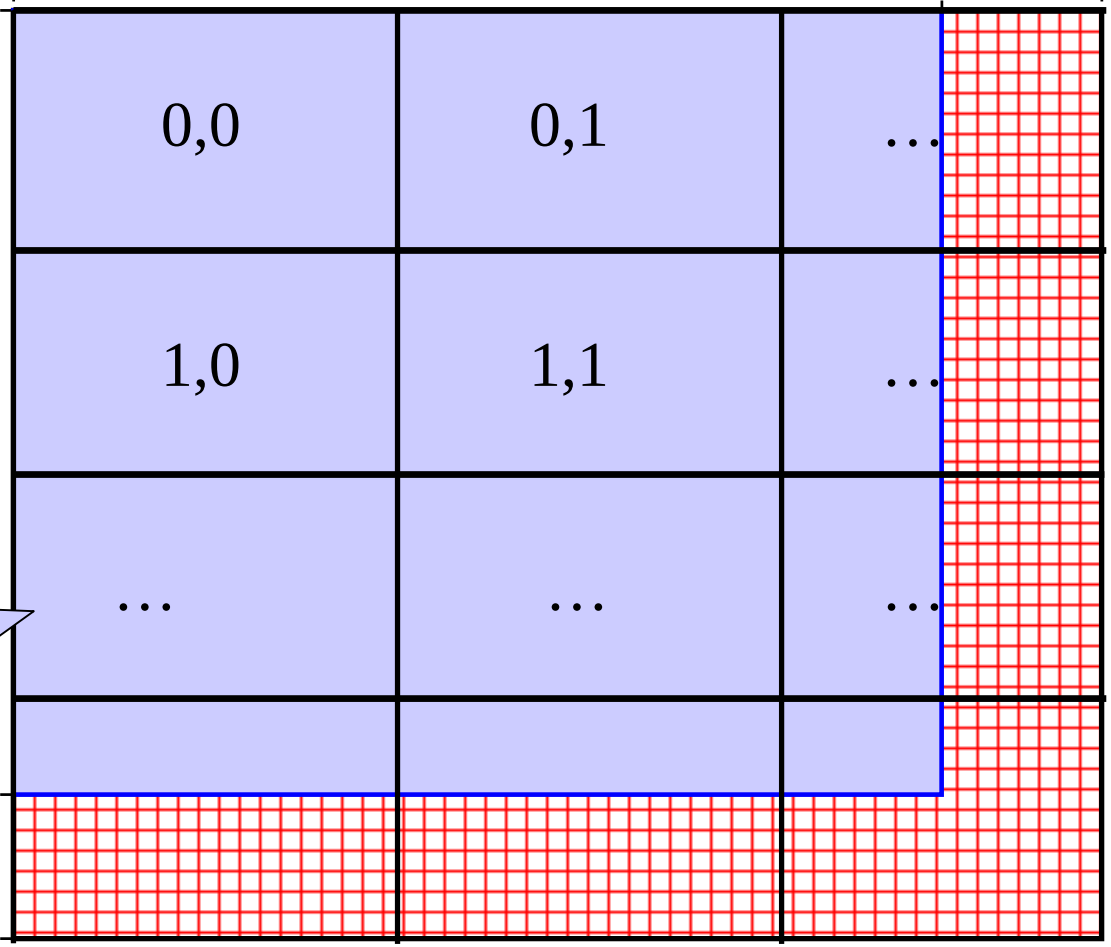


CUDA Block/Grid Decomposition

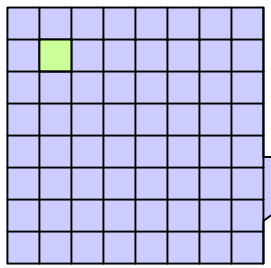
MO 3-D lattice decomposes into 2-D slices (CUDA grids)



Grid of thread blocks:



Small 8x8 thread blocks afford large per-thread register count, shared mem. Threads compute one MO lattice point each.



Padding optimizes glob. mem perf, guaranteeing coalescing

MO Kernel for One Grid Point (Naive C)

```
…
for (at=0; at<numatoms; at++) {
   int prim_counter = atom_basis[at];
   calc_distances_to_atom(&atompos[at], &xdist, &ydist, &zdist, &dist2, &xdiv);
   for (contracted_gto=0.0f, shell=0; shell < num_shells_per_atom[at]; shell++) {
      int shell_type = shell_symmetry[shell_counter];
      for (prim=0; prim < num_prim_per_shell[shell_counter];  prim++) {
         float exponent       = basis_array[prim_counter    ];
         float contract_coeff = basis_array[prim_counter + 1];
         contracted_gto += contract_coeff * expf(-exponent*dist2);
         prim_counter += 2;
      }
      for (tmpshell=0.0f, j=0, zdp=1.0f; j<=shell_type; j++, zdp*=zdist) {
        int imax = shell_type - j;
        for (i=0, ydp=1.0f, xdp=pow(xdist, imax); i<=imax; i++, ydp*=ydist, xdp*=xdiv)
          tmpshell += wave_f[ifunc++] * xdp * ydp * zdp;
      }
      value += tmpshell * contracted_gto;
      shell_counter++;
   }
} …..
```

Loop over atoms

Loop over shells

Loop over primitives: largest component of runtime, due to expf()

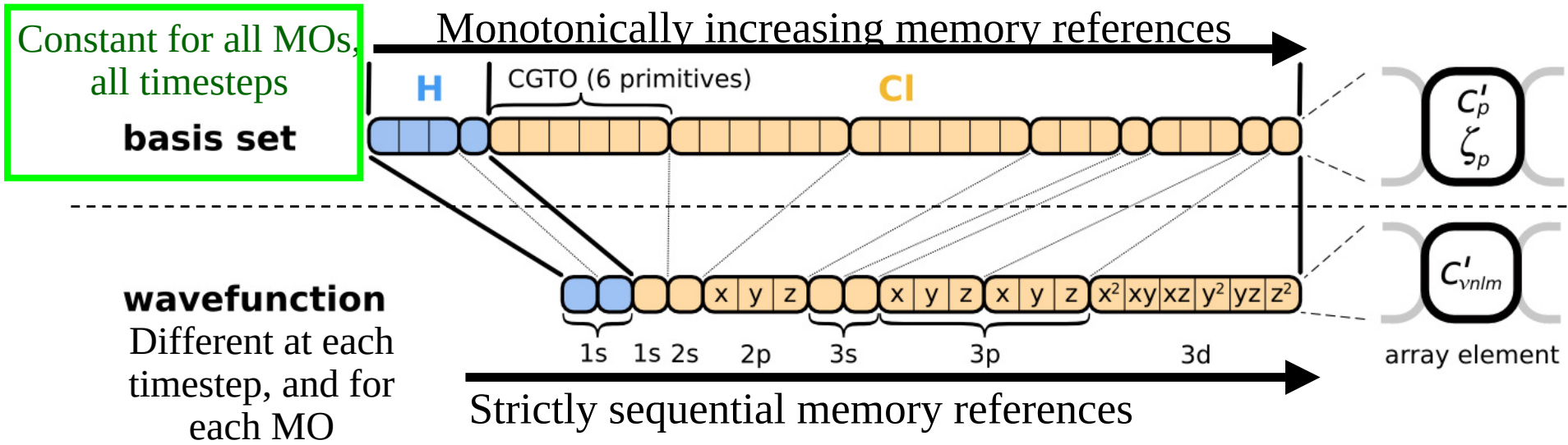
Loop over angular momenta

(unrolled in real code)

Preprocessing of Atoms, Basis Set, and Wavefunction Coefficients

- Must make effective use of high bandwidth, low-latency GPU on-chip memory, or CPU cache:
  - Overall storage requirement reduced by eliminating duplicate basis set coefficients
  - Sorting atoms by element type allows re-use of basis set coefficients for subsequent atoms of identical type
- Padding, alignment of arrays guarantees coalesced GPU global memory accesses, CPU SSE loads

GPU Traversal of Atom Type, Basis Set, Shell Type, and Wavefunction Coefficients

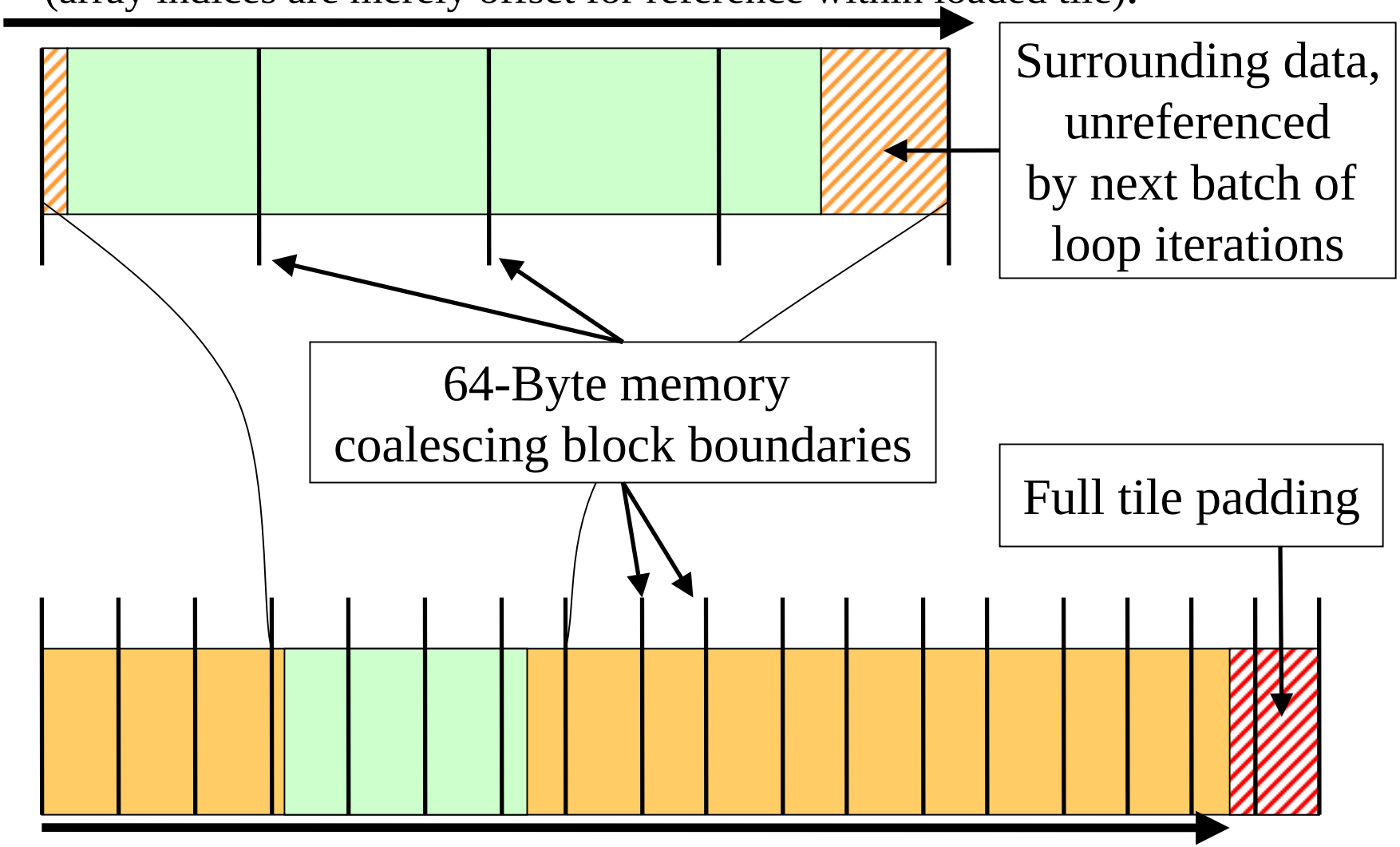


- Loop iterations always access same or consecutive array elements for all threads in a thread block:
  - Yields good constant memory cache performance
  - Increases shared memory tile reuse

Use of GPU On-chip Memory

- If total data less than 64 kB, use only const mem:
  - Broadcasts data to all threads, no global memory accesses!
- For large data, shared memory used as a program-managed cache, coefficients loaded on-demand:
  - Tiles sized large enough to service entire inner loop runs, broadcast to all 64 threads in a block
  - Complications: nested loops, multiple arrays, varying length
  - Key to performance is to locate tile loading checks outside of the two performance-critical inner loops
  - Only 27% slower than hardware caching provided by constant memory (GT200)
  - Next-gen “Fermi” GPUs will provide larger on-chip shared memory, L1/L2 caches, reduced control overhead

Array tile loaded in GPU shared memory. Tile size is a power-of-two, multiple of coalescing size, and allows simple indexing in inner loops (array indices are merely offset for reference within loaded tile).



VMD MO Performance Results for C_{60}

Sun Ultra 24: Intel Q6600, NVIDIA GTX 280

Kernel	Cores/GPUs	Runtime (s)	Speedup
CPU ICC-SSE	1	46.58	1.00
CPU ICC-SSE	4	11.74	3.97
CPU ICC-SSE-approx**	4	3.76	12.4
CUDA-tiled-shared	1	0.46	100.
CUDA-const-cache	1	0.37	126.
CUDA-const-cache-JIT*	**1**	**0.27**	**173.** **(JIT 40% faster)**

C_{60} basis set 6-31Gd. We used an unusually-high resolution MO grid for accurate timings. A more typical calculation has $1/8^{th}$ the grid points.

* Runtime-generated JIT kernel compiled using batch mode CUDA tools

**Reduced-accuracy approximation of expf(), cannot be used for zero-valued MO isosurfaces

Performance Evaluation:
Molekel, MacMolPlt, and VMD
Sun Ultra 24: Intel Q6600, NVIDIA GTX 280

	C_{60}-A	C_{60}-B	Thr-A	Thr-B	Kr-A	Kr-B
Atoms	60	60	17	17	1	1
Basis funcs (unique)	**300 (5)**	**900 (15)**	**49 (16)**	**170 (59)**	**19 (19)**	**84 (84)**

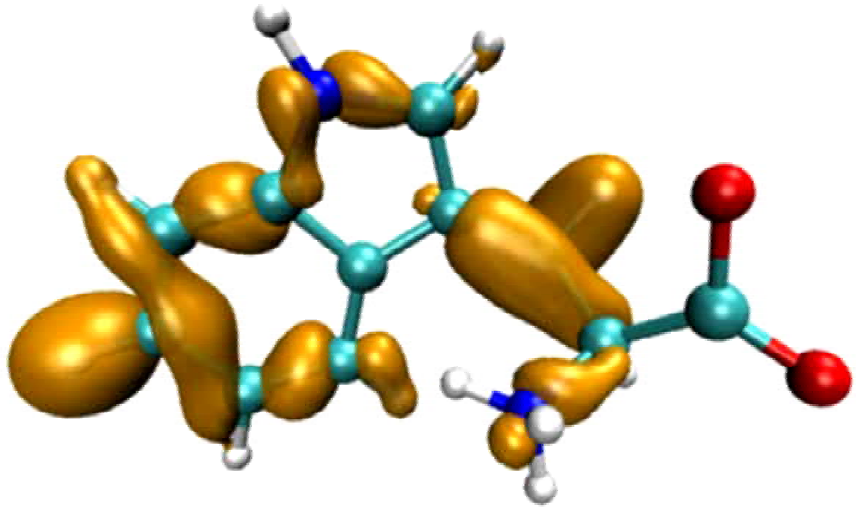
Kernel	**Cores GPUs**	Speedup vs. Molekel on 1 CPU core					
Molekel	1*	1.0	1.0	1.0	1.0	1.0	1.0
MacMolPlt	4	2.4	2.6	2.1	2.4	4.3	4.5
VMD GCC-cephes	4	3.2	4.0	3.0	3.5	4.3	6.5
VMD ICC-SSE-cephes	4	16.8	17.2	13.9	12.6	17.3	21.5
VMD ICC-SSE-approx**	4	59.3	53.4	50.4	49.2	54.8	69.8
VMD CUDA-const-cache	1	552.3	533.5	355.9	421.3	193.1	571.6

VMD Orbital Dynamics Proof of Concept

One GPU can compute and animate this movie on-the-fly!

CUDA const-cache kernel,
Sun Ultra 24, GeForce GTX 285

GPU MO grid calc.	**0.016 s**
CPU surface gen, volume gradient, and GPU rendering	**0.033 s**
Total runtime	**0.049 s**
Frame rate	**20 FPS**

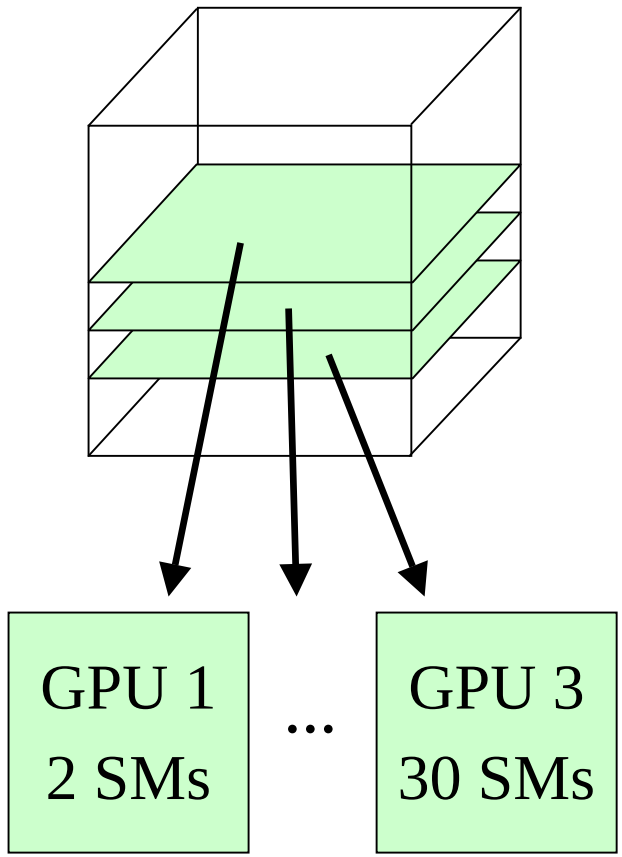


threonine

With GPU speedups over **100x**, previously insignificant CPU surface gen, gradient calc, and rendering are now **66%** of runtime. Need GPU-accelerated surface gen next…

Multi-GPU Load Balance

- All new NVIDIA cards support CUDA, so a typical machine may have a diversity of GPUs of varying capability
- Static decomposition works poorly for non-uniform workload, or diverse GPUs, e.g. w/ 2 SM, 16 SM, 30 SM
- VMD uses a multithreaded dynamic GPU work distribution and error handling system



VMD Multi-GPU Molecular Orbital Performance Results for C_{60}

Kernel	Cores/GPUs	Runtime (s)	Speedup	Parallel Efficiency
CPU-ICC-SSE	1	46.580	1.00	100%
CPU-ICC-SSE	4	11.740	3.97	99%
CUDA-const-cache	1	0.417	112	100%
CUDA-const-cache	2	0.220	212	94%
CUDA-const-cache	3	0.151	308	92%
CUDA-const-cache	4	0.113	412	92%

Intel Q6600 CPU, 4x Tesla C1060 GPUs,

Uses persistent thread pool to avoid GPU init overhead, dynamic scheduler distributes work to GPUs

VMD Multi-GPU Molecular Orbital Performance Results for C_{60} Using Mapped Host Memory

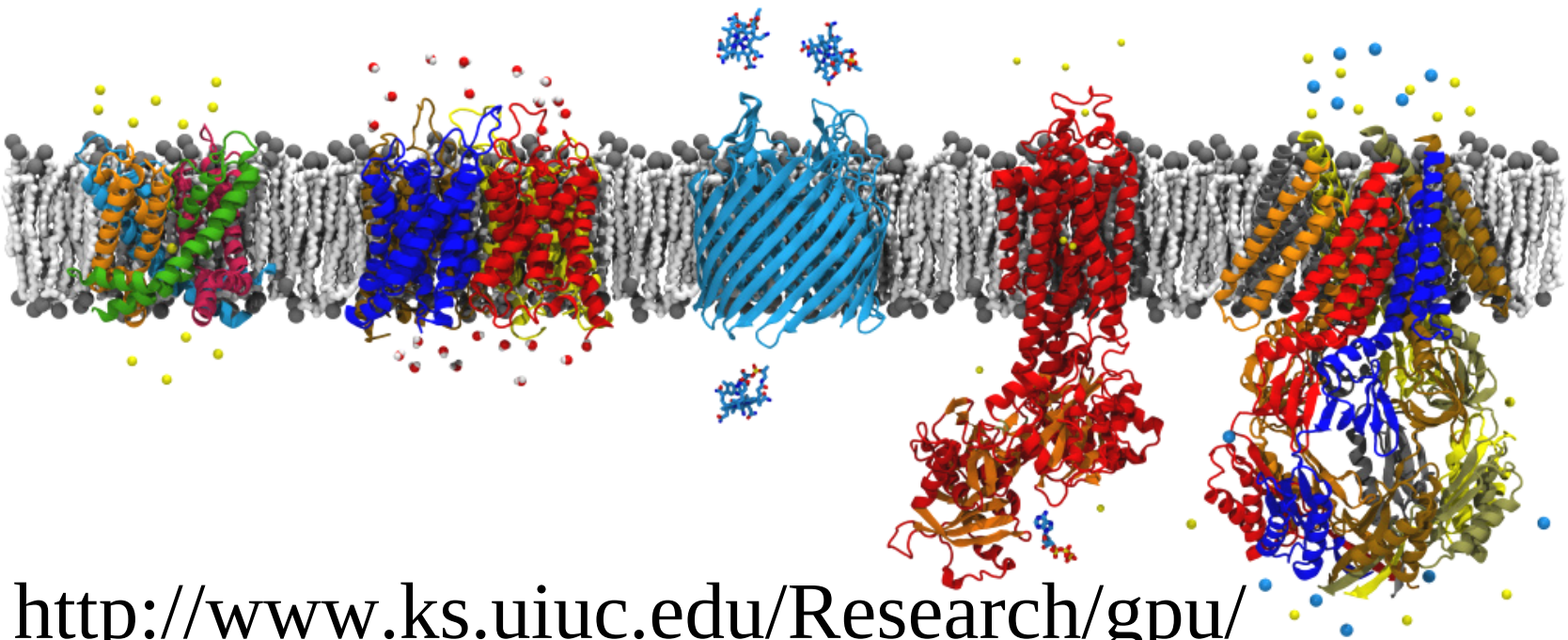
Kernel	Cores/GPUs	Runtime (s)	Speedup
CPU-ICC-SSE	1	46.580	1.00
CPU-ICC-SSE	4	11.740	3.97
CUDA-const-cache	3	0.151	308.
CUDA-const-cache w/ mapped host memory	**3**	**0.137**	**340.**

Intel Q6600 CPU, 3x Tesla C1060 GPUs,

GPU kernel writes output directly to host memory, no extra cudaMemcpy() calls to fetch results!

See cudaHostAlloc() + cudaGetDevicePointer()

NAMD: Molecular Dynamics on GPUs

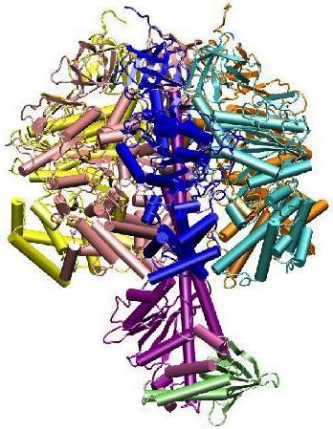


http://www.ks.uiuc.edu/Research/gpu/

http://www.ks.uiuc.edu/Research/namd/

NAMD Highlights

2002 Gordon Bell Award



ATP synthase

PSC Lemieux

34,000 Users, 1200 Citations

Computational Biophysics Summer School

Blue Waters Target Application

Illinois Petascale Computing Facility

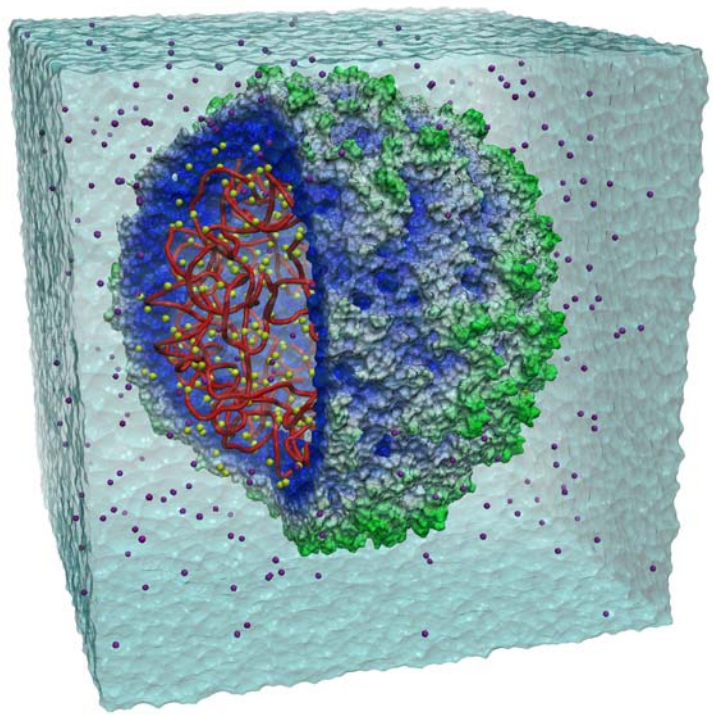
GPU Acceleration

NVIDIA Tesla

NCSA Lincoln

NAMD 2.7b2 w/ CUDA Released

- CUDA-enabled NAMD binaries for 64-bit Linux are available on the NAMD web site now!
  http://www.ks.uiuc.edu/Research/namd/2.7b2/announce.html
- Direct download link:
  http://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD

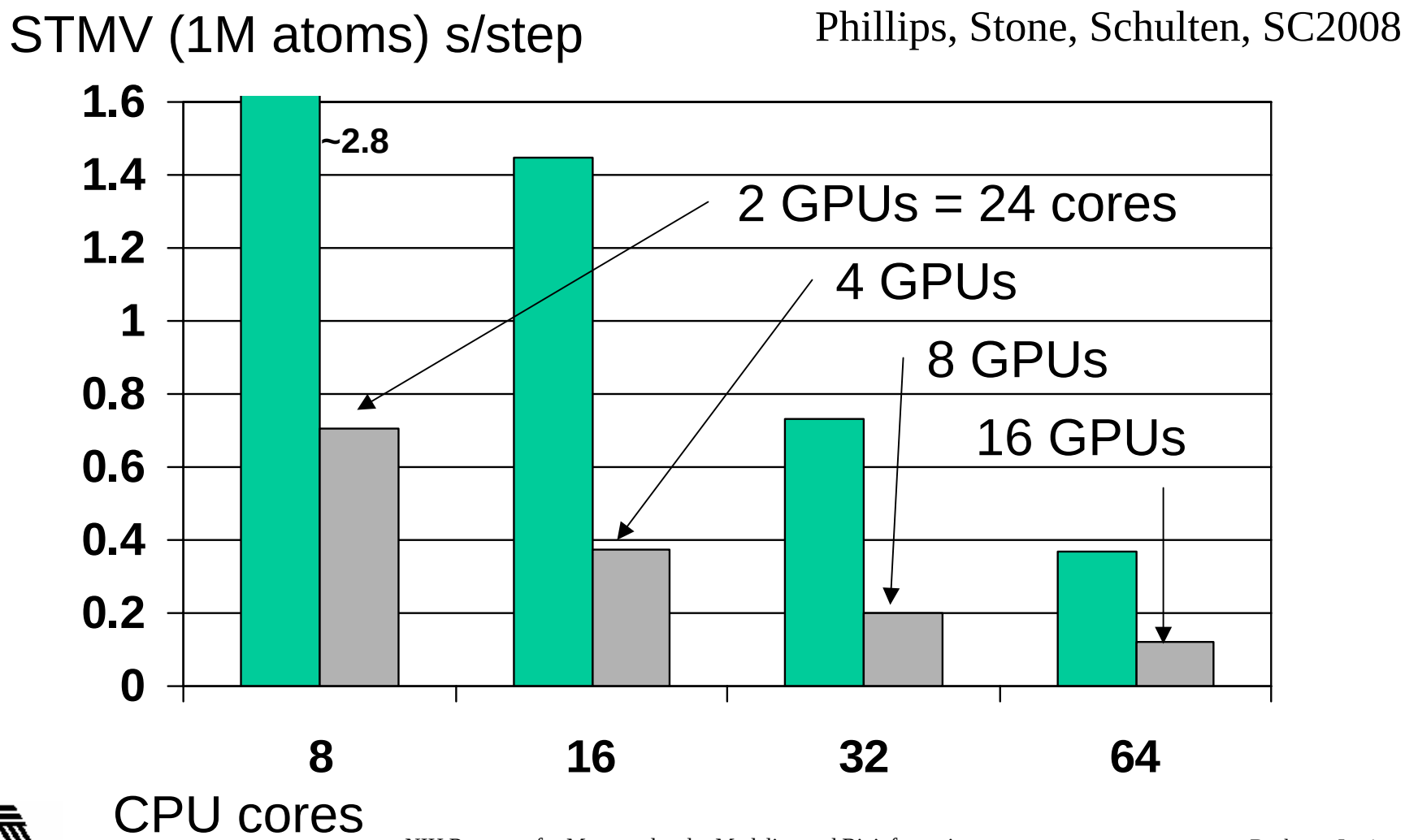


National Center for Research Resources

Beckman Institute, UIUC

NCSA Lincoln Cluster Performance

(8 Intel cores and 2 NVIDIA Telsa GPUs per node)



GPU-Accelerated NAMD Plans

- Serial performance
  - Target NVIDIA Fermi architecture
  - Revisit GPU kernel design decisions made in 2007
  - Improve performance of remaining CPU code
- Parallel scaling
  - Target NSF Track 2D Keeneland cluster at ORNL
  - Finer-grained work units on GPU (feature of Fermi)
  - One process per GPU, one thread per CPU core
  - Dynamic load balancing of GPU work
- Wider range of simulation options and features

Acknowledgements

- Additional Information and References:
  - http://www.ks.uiuc.edu/Research/gpu/
- Questions, source code requests:
  - John Stone: johns@ks.uiuc.edu
- Acknowledgements:
  - J. Phillips, D. Hardy, J. Saam,
    UIUC Theoretical and Computational Biophysics Group,
    NIH Resource for Macromolecular Modeling and Bioinformatics
  - Prof. Wen-mei Hwu, Christopher Rodrigues, UIUC IMPACT Group
  - CUDA team at NVIDIA
  - UIUC NVIDIA CUDA Center of Excellence
  - NIH support: P41-RR05969

Publications

http://www.ks.uiuc.edu/Research/gpu/

- Probing Biomolecular Machines with Graphics Processors. J. Phillips, J. Stone. *Communications of the ACM*, 52(10):34-41, 2009.
- GPU Clusters for High Performance Computing. V. Kindratenko, J. Enos, G. Shi, M. Showerman, G. Arnold, J. Stone, J. Phillips, W. Hwu. *Workshop on Parallel Programming on Accelerator Clusters (PPAC)*, IEEE Cluster 2009. In press.
- Long time-scale simulations of in vivo diffusion using GPU hardware. E. Roberts, J. Stone, L. Sepulveda, W. Hwu, Z. Luthey-Schulten. In *IPDPS'09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Computing*, pp. 1-8, 2009.
- 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, K. Schulten, *2nd Workshop on General-Purpose Computation on Graphics Pricessing Units (GPGPU-2), ACM International Conference Proceeding Series*, volume 383, pp. 9-18, 2009.
- Multilevel summation of electrostatic potentials using graphics processing units. D. Hardy, J. Stone, K. Schulten. *J. Parallel Computing*, 35:164-177, 2009.

Publications (cont)

http://www.ks.uiuc.edu/Research/gpu/

- Adapting a message-driven parallel application to GPU-accelerated clusters. J. Phillips, J. Stone, K. Schulten. *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, IEEE Press, 2008.
- GPU acceleration of cutoff pair potentials for molecular modeling applications. C. Rodrigues, D. Hardy, J. Stone, K. Schulten, and W. Hwu. *Proceedings of the 2008 Conference On Computing Frontiers*, pp. 273-282, 2008.
- GPU computing. J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, J. Phillips. *Proceedings of the IEEE*, 96:879-899, 2008.
- Accelerating molecular modeling applications with graphics processors. J. Stone, J. Phillips, P. Freddolino, D. Hardy, L. Trabuco, K. Schulten. *J. Comp. Chem.*, 28:2618-2640, 2007.
- Continuous fluorescence microphotolysis and correlation spectroscopy. A. Arkhipov, J. Hüve, M. Kahms, R. Peters, K. Schulten. *Biophysical Journal*, 93:4006-4017, 2007.