Early Experiences Scaling VMD Molecular Visualization and Analysis Jobs on Blue Waters

John E. Stone, Barry Isralewitz, Klaus Schulten

Theoretical and Computational Biophysics Group

Beckman Institute for Advanced Science and Technology

University of Illinois at Urbana-Champaign

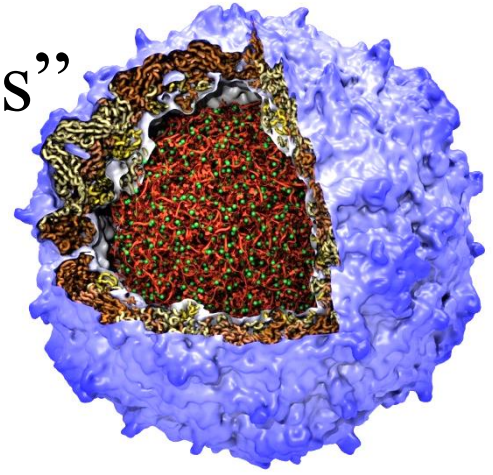
http://www.ks.uiuc.edu/Research/vmd/

XSEDE Extreme Scaling Workshop

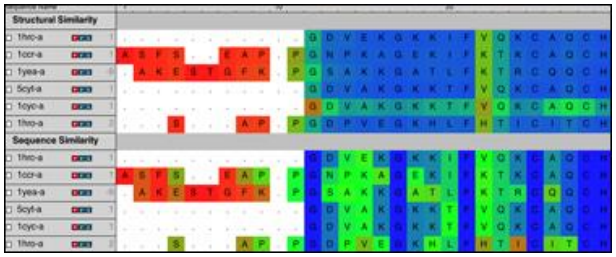
Boulder, CO, August 15, 2013

VMD – "Visual Molecular Dynamics"

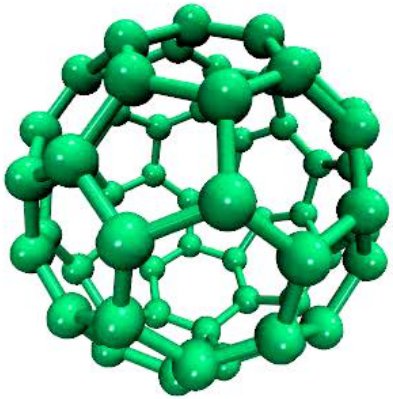
- Visualization and analysis of:
  - molecular dynamics simulations
  - quantum chemistry calculations
  - particle systems and whole cells
  - sequence data
- User extensible w/ scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/



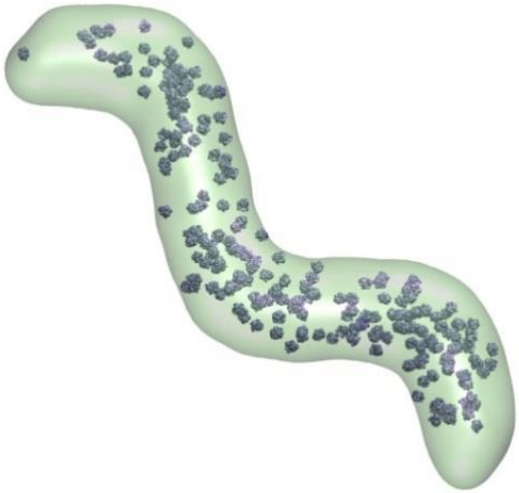
Poliovirus



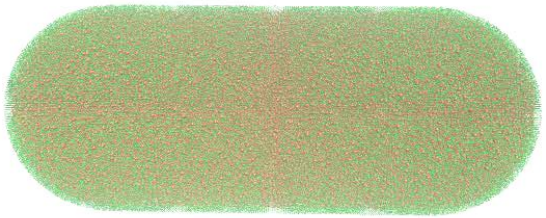
Ribosome Sequences



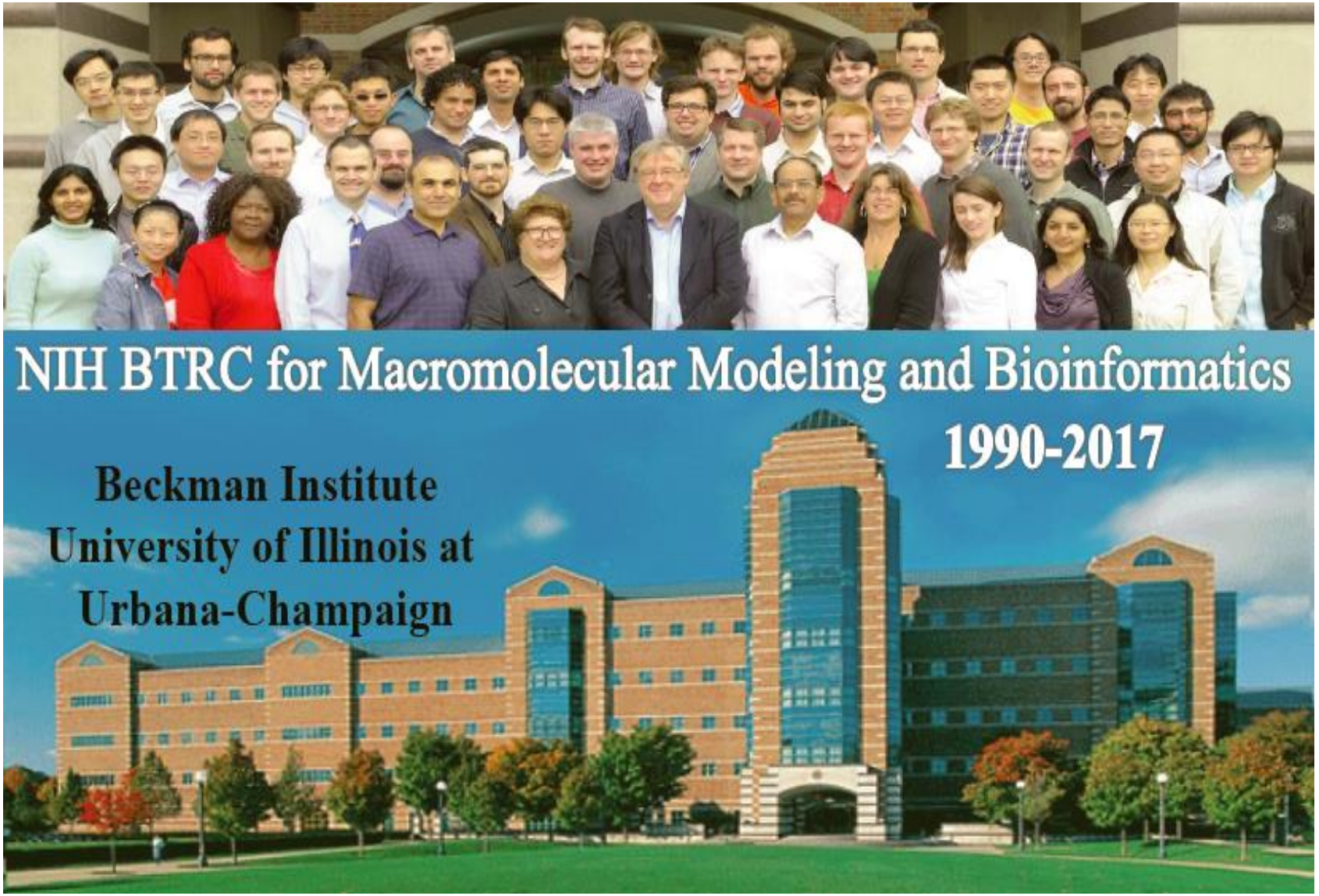
Electrons in
Vibrating Buckyball



Cellular Tomography,
Cryo-electron Microscopy

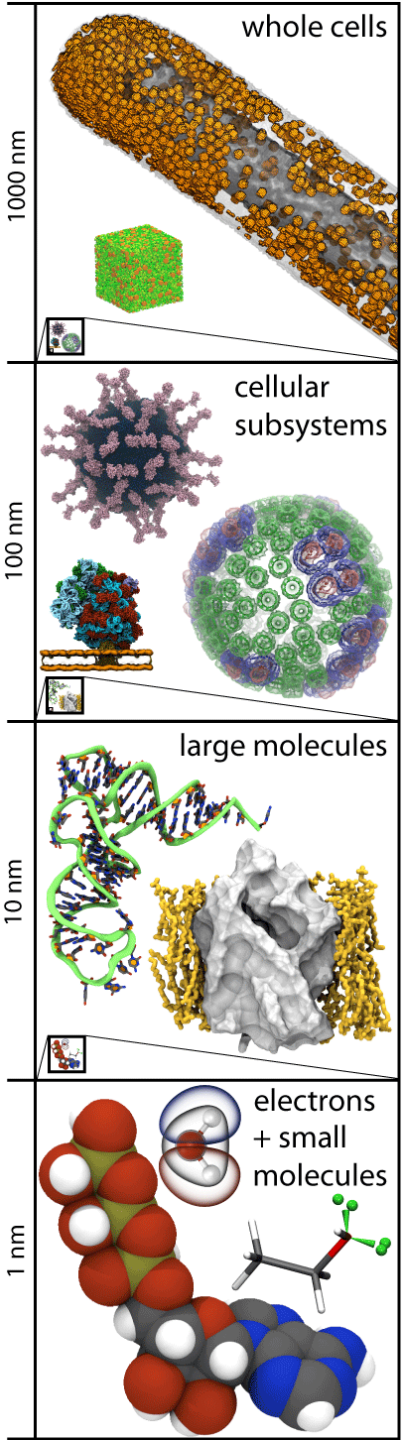


Whole Cell Simulations


NIH BTRC for Macromolecular Modeling and Bioinformatics
1990-2017
Beckman Institute
University of Illinois at
Urbana-Champaign

VMD Interoperability – Linked to Today's Key Research Areas

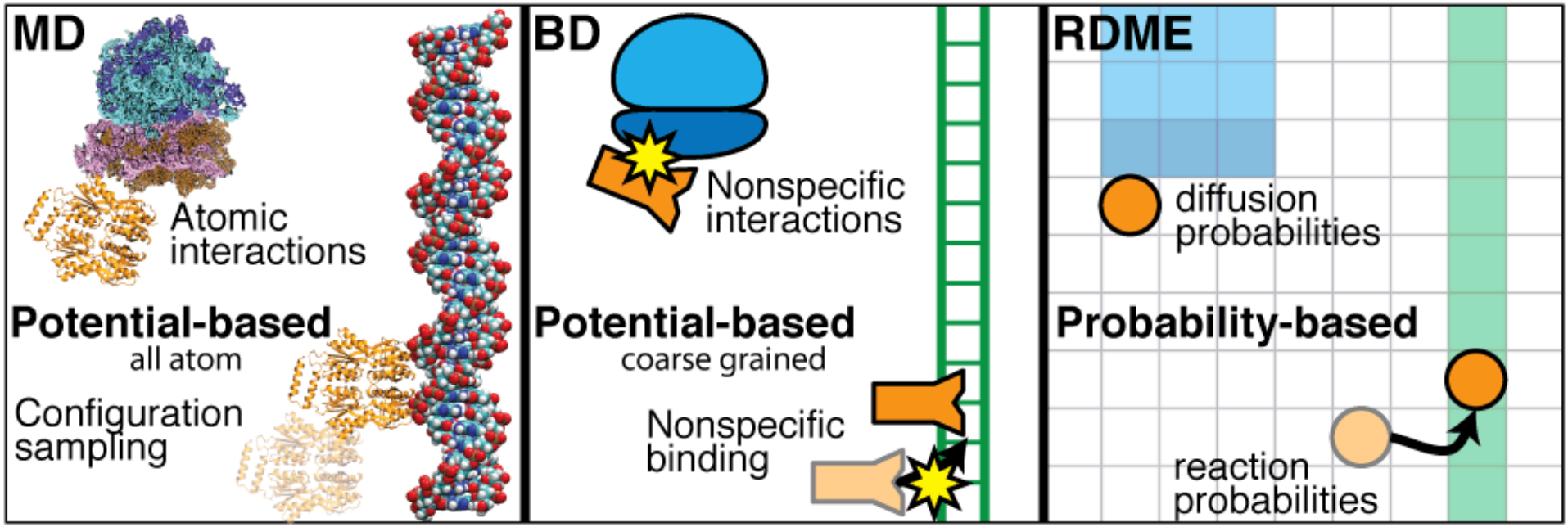
- Unique in its interoperability with a broad range of modeling tools: AMBER, CHARMM, CPMD, DL_POLY, GAMESS, GROMACS, HOOMD, LAMMPS, NAMD, and many more …
- Supports key data types, file formats, and databases, e.g. electron microscopy, quantum chemistry, MD trajectories, sequence alignments, super resolution light microscopy
- Incorporates tools for simulation preparation, visualization, and analysis



Modeling of Molecules to Cells

Biomolecular interactions span many orders of magnitude in space and time

Molecular Dynamics Brownian Dynamics Reactions and Diffusion

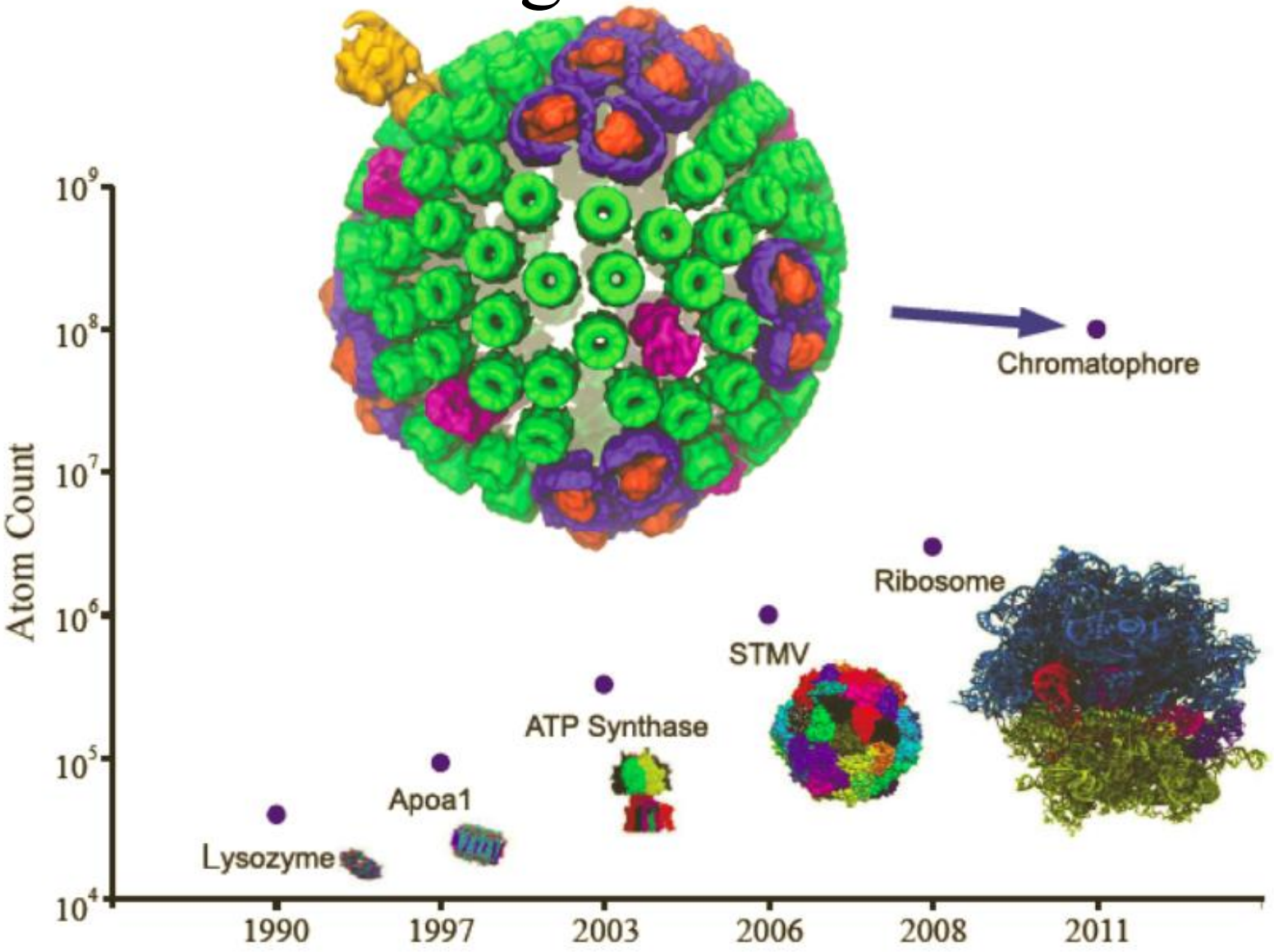


Femtoseconds **Picoseconds** **Microseconds**

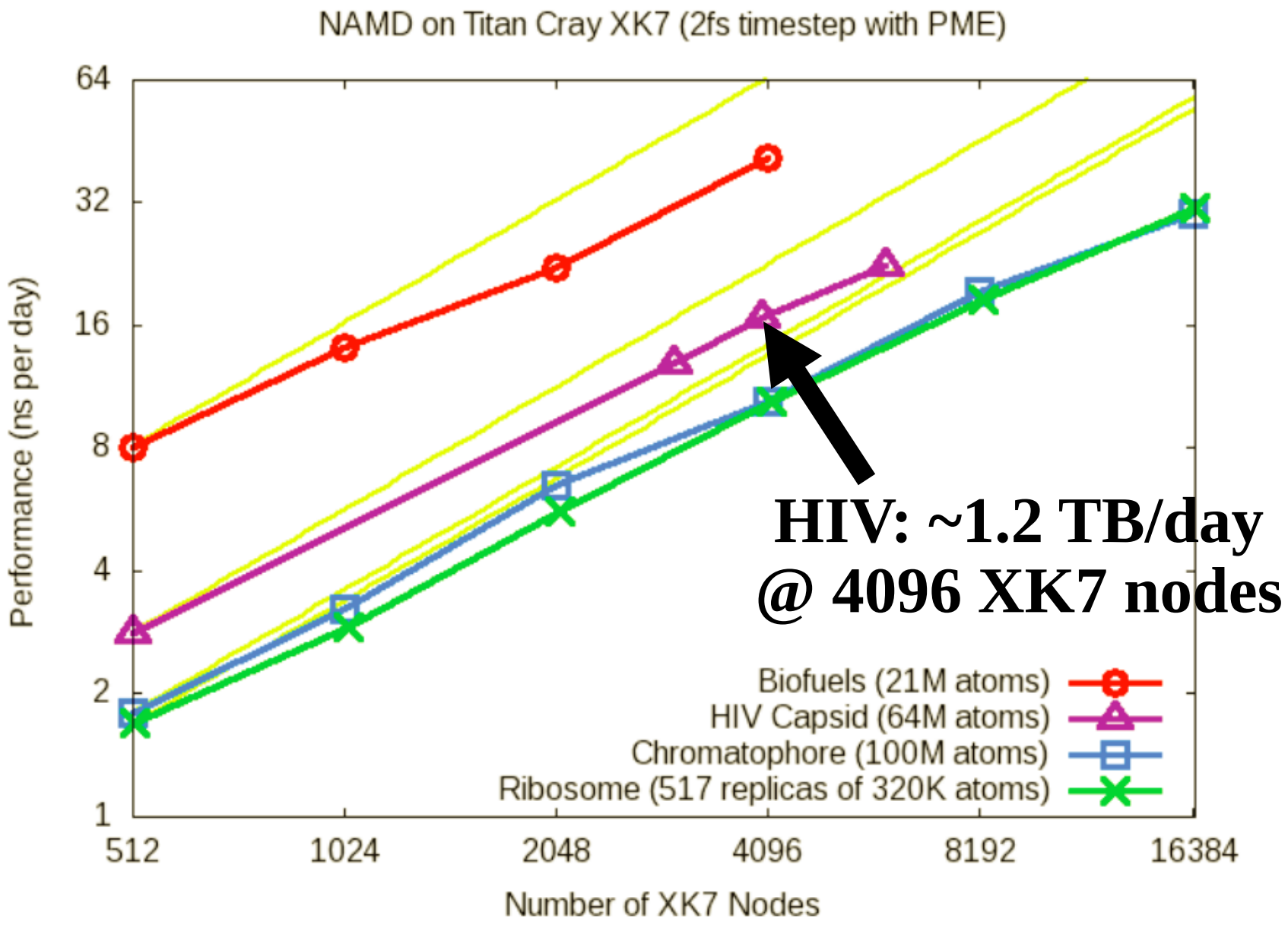
Molecules to Macromolecular assemblies

Whole Cells

Larger Supercomputers Enable Larger Simulations…



NAMD Titan XK7 Performance August 2013



VMD Petascale Visualization and Analysis

- Analyze/visualize large trajectories too large to transfer off-site:
  - Compute time-averaged electrostatic fields, MDFF quality-of-fit, etc.
  - User-defined parallel analysis operations, data types
  - Parallel rendering, movie making
- Parallel I/O rates up to **275 GB/sec** on 8192 Cray XE6 nodes – can read in **231 TB in 15 minutes!**
- Multi-level dynamic load balancing tested with up to 262,144 CPU cores
- **Supports GPU-accelerated Cray XK7 nodes for both visualization and analysis usage**

NCSA Blue Waters Hybrid Cray XE6 / XK7 Supercomputer

22,640 XE6 CPU nodes

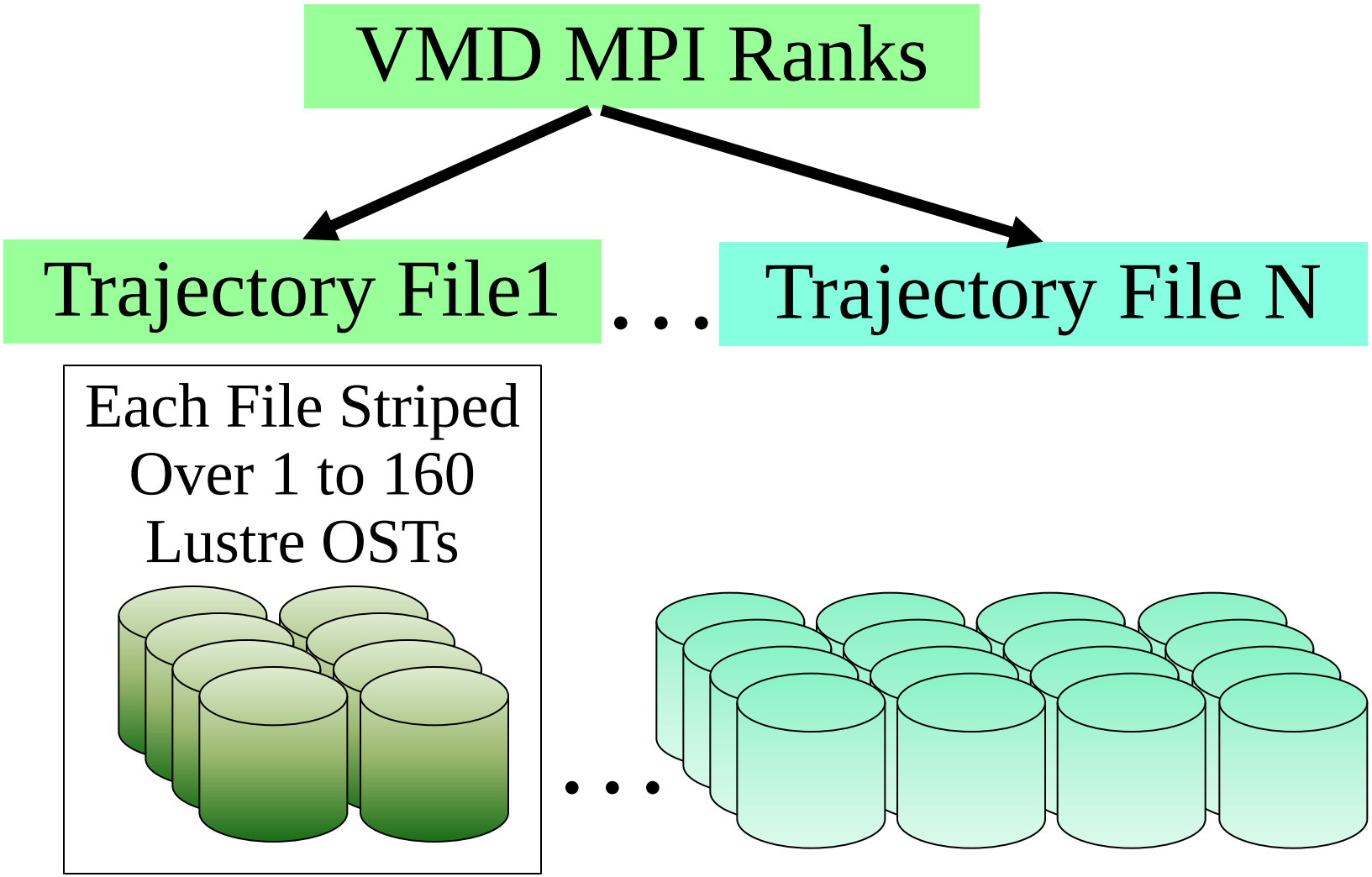
3,072 XK7 nodes w/ GPUs support fast VMD OpenGL movie rendering and visualization

BW Upgrades adding 1,152 more XK7 nodes for 4,224 total

BW VMD Parallelization Overview

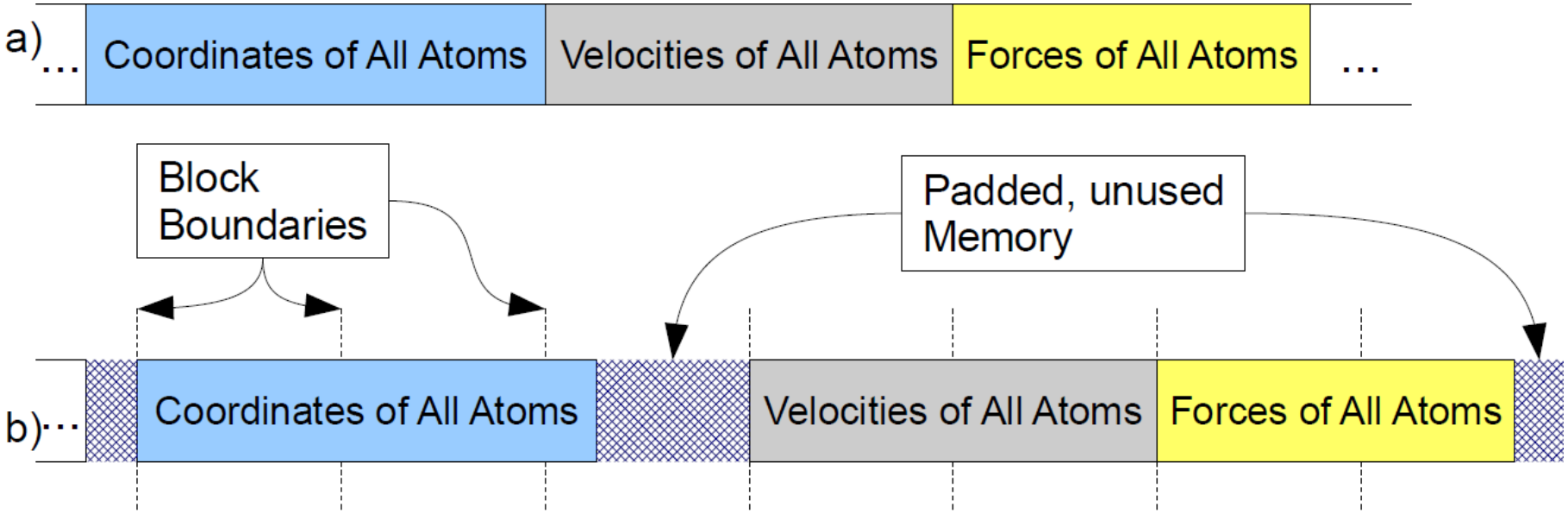
- Main approach is to launch VMD with one MPI rank per node – maximizing RAM/rank
- VMD uses POSIX threads internally to exploit multi-core CPUs on the nodes
- GPU acceleration provided by CUDA
- VMD uses its own GPU-aware work scheduler for dynamic load balancing, error recovery, CPU-fallback, **lock-free inner loops** on CPUS with *atomic-fetch-and-add*

VMD Parallel I/O on Blue Waters



Total of 1440 OSTs in Blue Waters
Scratch Filesystem

Trajectory File Format Structure Required for Direct I/O



Immersive out-of-core visualization of large-size and long-timescale molecular dynamics trajectories
J. Stone, K. Vandivort, and K. Schulten.
Lecture Notes in Computer Science, 6939:1-12, 2011.

Parallel Read Performance

- Used unbuffered VM page-multiple and VM page-aligned I/Os **(O_DIRECT)** to eliminate extra buffer copies in the OS kernel
- Read from hundreds of 10-20GB files to ensure utilization of all 1440 OSTs
- **Peak I/O rate 275/GB/s on 8192 nodes**
- VMD XE6 I/O rates scale as:

~1TB/sec peak × fraction of machine, better for smaller node counts

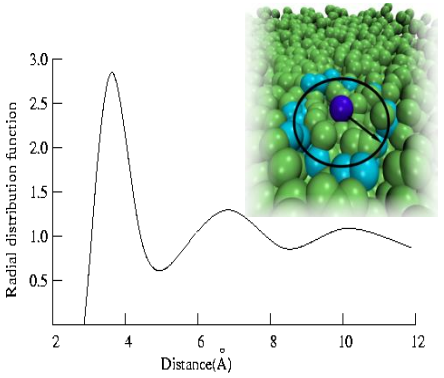
- 512 node **XK7 I/O rates (59GB/s) half** that of **XE6 rates (109GB/s)** – initial BW config had no XIO (LNET) nodes in the XK7 partition

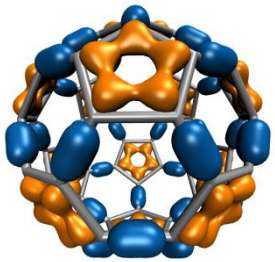
VMD Parallel Read Performance

Test Type	Node Type	Nodes	Per-file Data Size	Test Size	Parallel I/O Rate	
					Average	Peak
Combined Structure and Trajectory, Static Load Balancing	XK7	256	20,092 MB	5 TB	63 GB/sec	65 GB/sec
	XK7	512	20,092 MB	10 TB	55 GB/sec	59 GB/sec
	XE6	256	20,092 MB	5 TB	57 GB/sec	60 GB/sec
	XE6	512	20,092 MB	10 TB	101 GB/sec	109 GB/sec
	XE6	1,024	20,092 MB	20 TB	126 GB/sec	137 GB/sec
	XE6	2,048	20,092 MB	40 TB	185 GB/sec	202 GB/sec
	XE6	4,096	20,092 MB	80 TB	217 GB/sec	218 GB/sec
Trajectory Only, Dynamic Load Balancing	XE6	1,024	9,641 MB	58 TB	123 GB/sec	127 GB/sec
	XE6	2,048	9,641 MB	115 TB	186 GB/sec	194 GB/sec
	XE6	4,096	9,641 MB	115 TB	211 GB/sec	230 GB/sec
	XE6	8,192	9,641 MB	154 TB	256 GB/sec	275 GB/sec
	XE6	8,192	9,641 MB	231 TB	-	234 GB/sec

CUDA GPU Accelerated Trajectory Analysis and Visualization in VMD

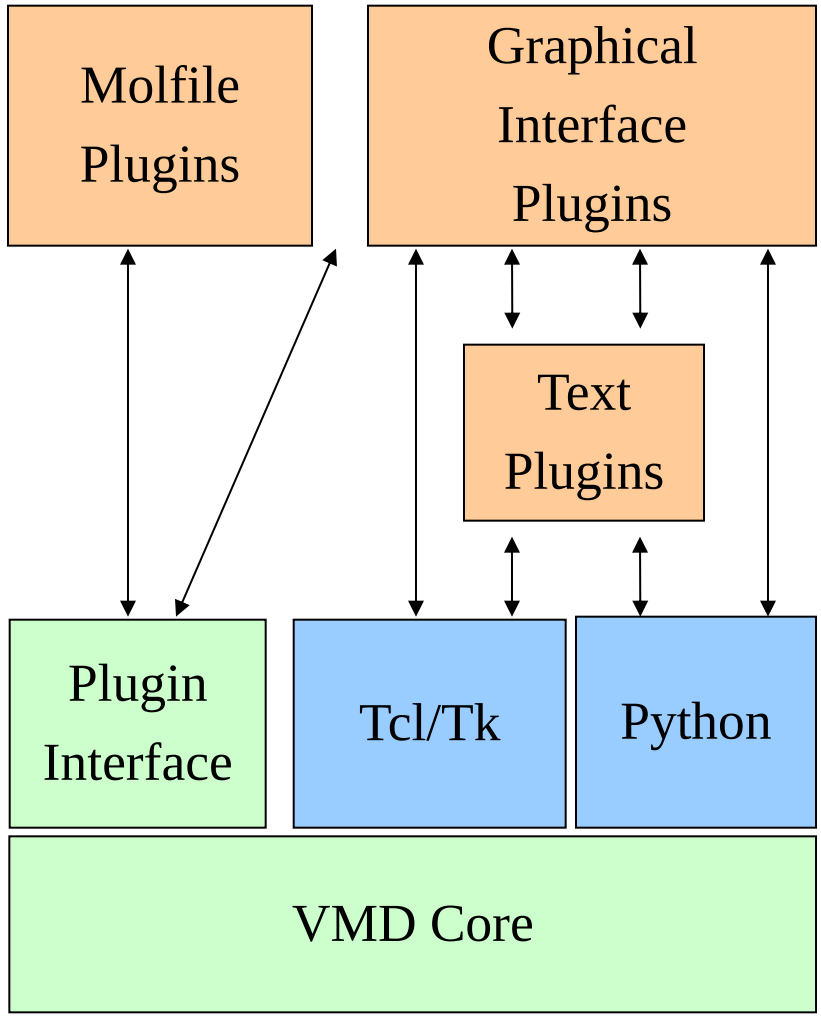
GPU-Accelerated Feature or Kernel	Typical speedup vs. a single CPU core
Molecular orbital display	120x
Radial distribution function	92x
Electrostatic field calculation	44x
Molecular surface display	40x
Ion placement	26x
MDFF density map synthesis	26x
Implicit ligand sampling	25x
Root mean squared fluctuation	25x
Radius of gyration	21x
Close contact determination	20x
Dipole moment calculation	15x





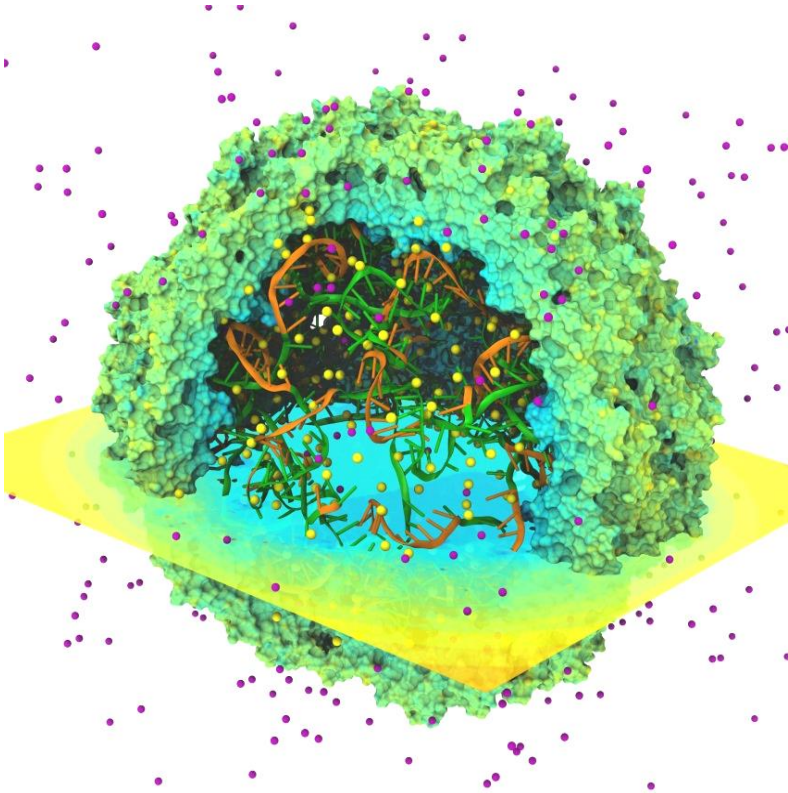
VMD as an Analysis Platform
Over 60 VMD Plugins Developed by Users

- VMD/NAMD sister programs, VMD is crucial for simulation analysis
- VMD user-extensible scripting w/ Tcl/Tk, Python
- Compiled C/C++ plugins loaded from shared libraries at runtime via **dlopen()**
- 70 molfile plugins provide access to molecular file formats
- **Built-in analysis commands exploit XE6 multi-core CPUs, XK7Tesla K20X GPUs**
- **New VMD collective ops and work scheduling interfaces enable existing code to be parallelized easily**



Time-Averaged Electrostatics Analysis on Energy-Efficient GPU Cluster

- **1.5 hour** job (CPUs) reduced to **3 min** (CPUs+GPU)
- Electrostatics of thousands of trajectory frames averaged
- Per-node power consumption on NCSA "AC" GPU cluster:
  - CPUs-only: 299 watts
  - CPUs+GPUs: 742 watts
- GPU Speedup: **25.5x**
- Power efficiency gain: **10.5x**



Quantifying the Impact of GPUs on Performance and Energy Efficiency in HPC Clusters. J. Enos, C. Steffen, J. Fullop, M. Showerman, G. Shi, K. Esler, V. Kindratenko, J. Stone, J. Phillips. *The Work in Progress in Green Computing*, pp. 317-324, 2010.

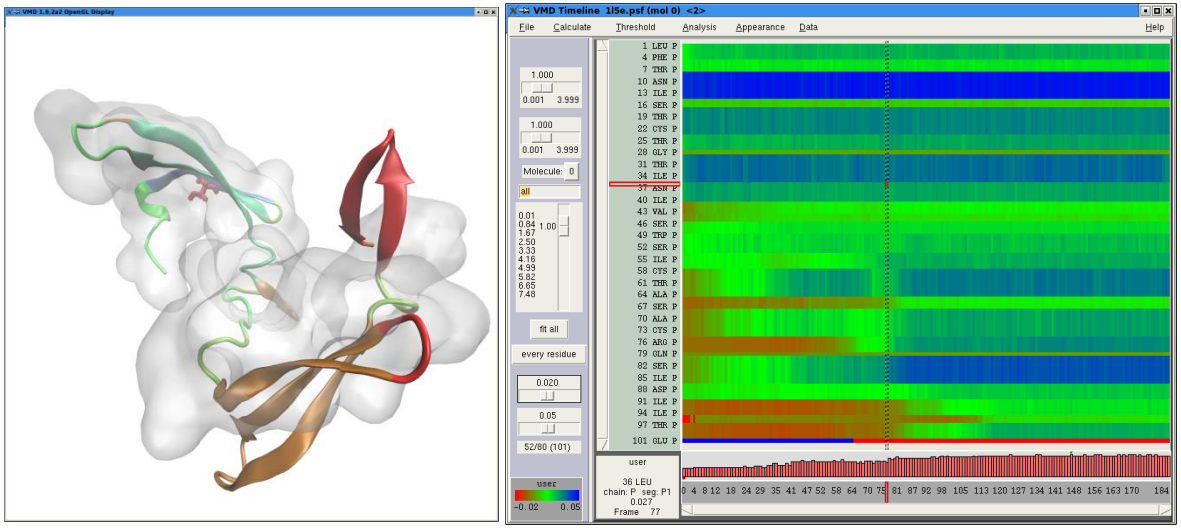
Time-Averaged Electrostatics Analysis on NCSA Blue Waters

NCSA Blue Waters Node Type	**Seconds per trajectory frame for one compute node**
Cray XE6 Compute Node: 32 CPU cores (2xAMD 6200 CPUs)	9.33
Cray XK6 GPU-accelerated Compute Node: 16 CPU cores + **NVIDIA X2090 (Fermi) GPU**	2.25
Speedup for GPU XK6 nodes vs. CPU XE6 nodes	**XK6 nodes are 4.15x faster overall**
Tests on XK7 nodes indicate MSM is CPU-bound with the Kepler K20X GPU. **Performance is not much faster (yet) than Fermi X2090** **Need to move spatial hashing, prolongation, interpolation onto the GPU…**	**In progress….** **XK7 nodes 4.3x faster overall**

Preliminary performance for VMD time-averaged electrostatics w/ Multilevel Summation Method on the NCSA Blue Waters Early Science System

Timeline Plugin: Analyze MD Trajectories for Events

MDFF quality-of-fit for cyanovirin-N



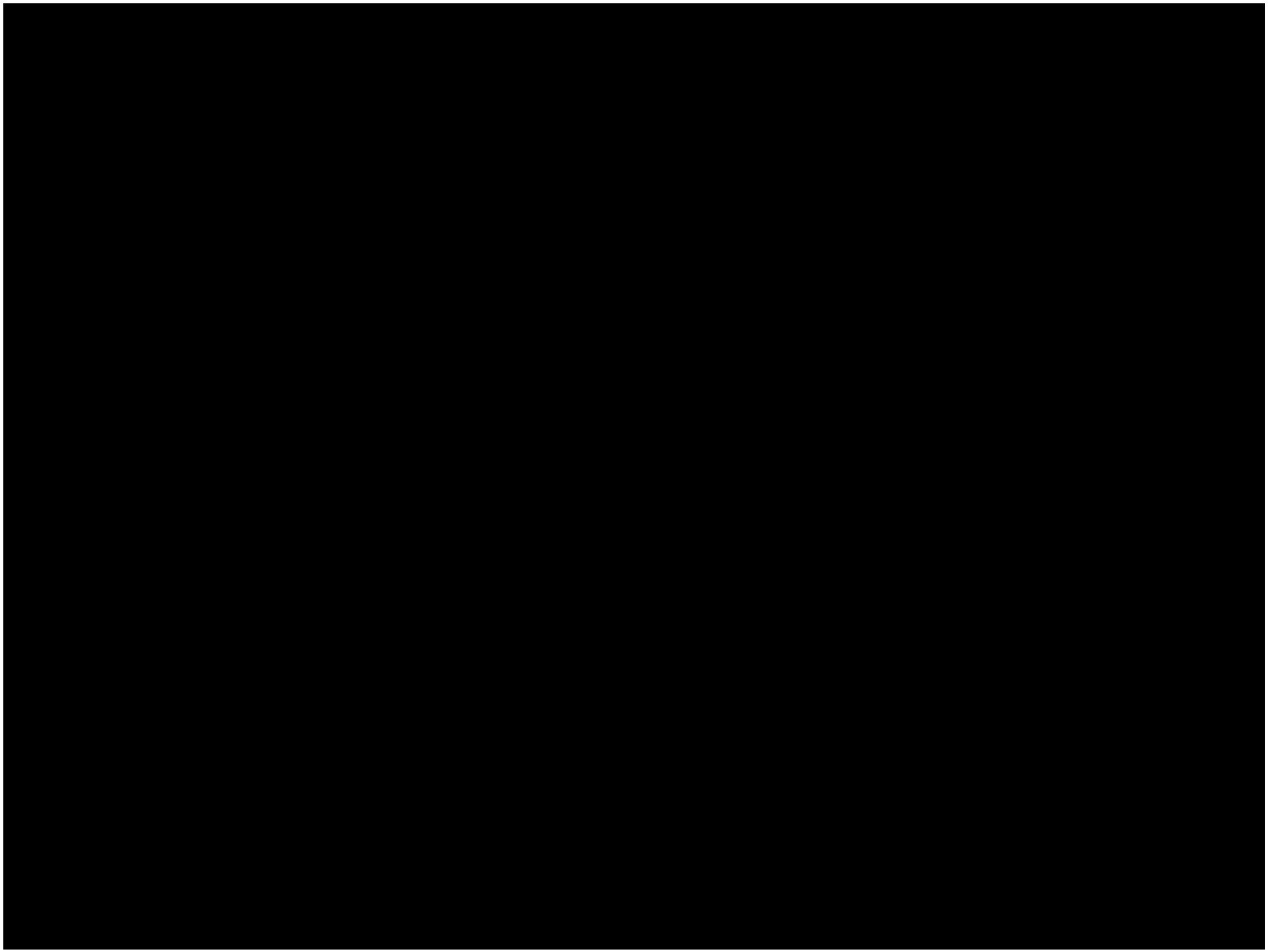
VMD Timeline plugin: live 2D plot linked to 3D structure

- Single picture shows changing properties across entire structure+trajectory
- Explore time vs. per-selection attribute, linked to molecular structure
- Many analysis methods available; user-extendable

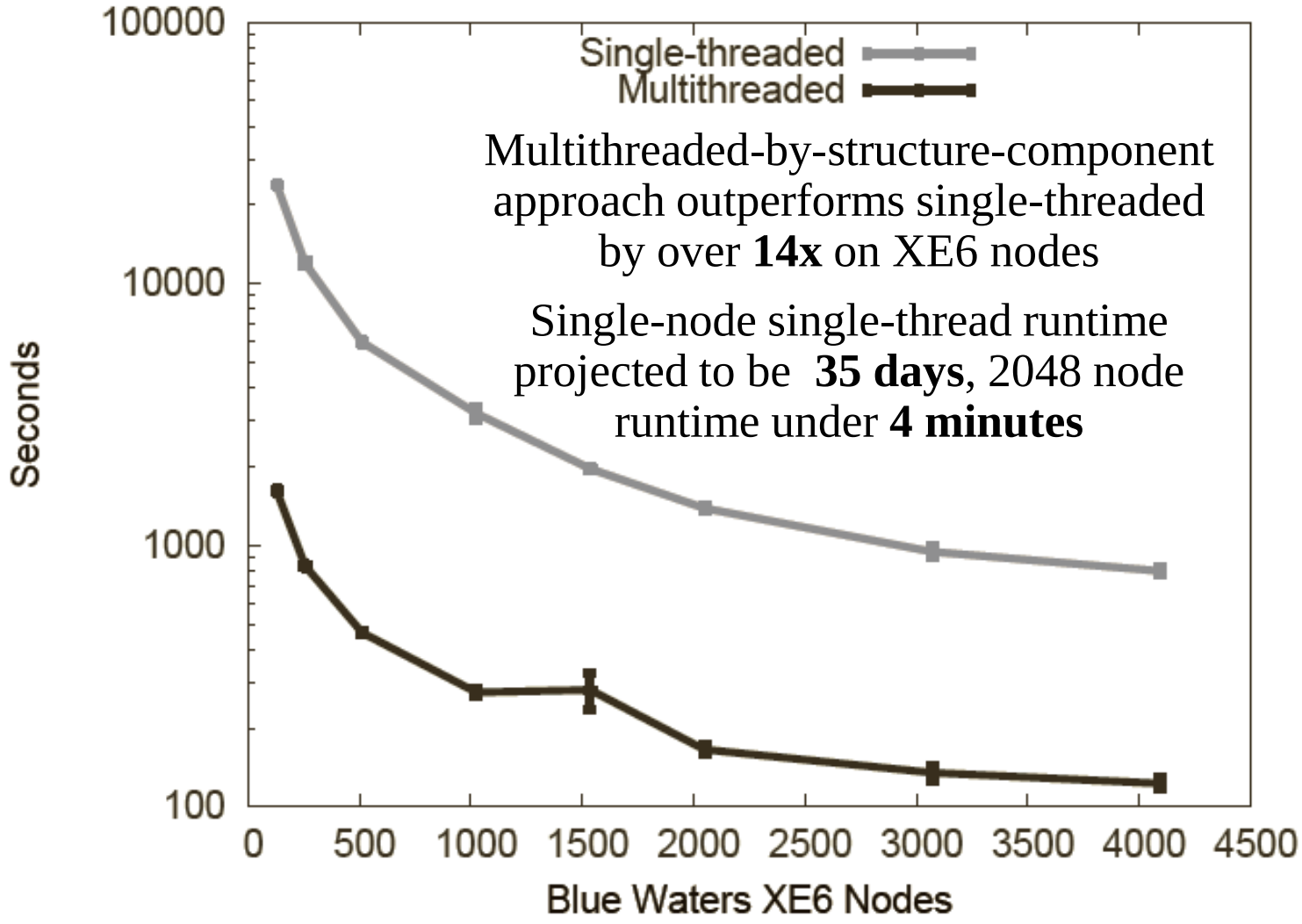
Recent progress:

- Adapted for data parallel analysis runs on Blue Waters
- Developed new approach for parallelizing Timeline calculations that involve larger numbers of very small atom selections

Interactive Timeline Screencast

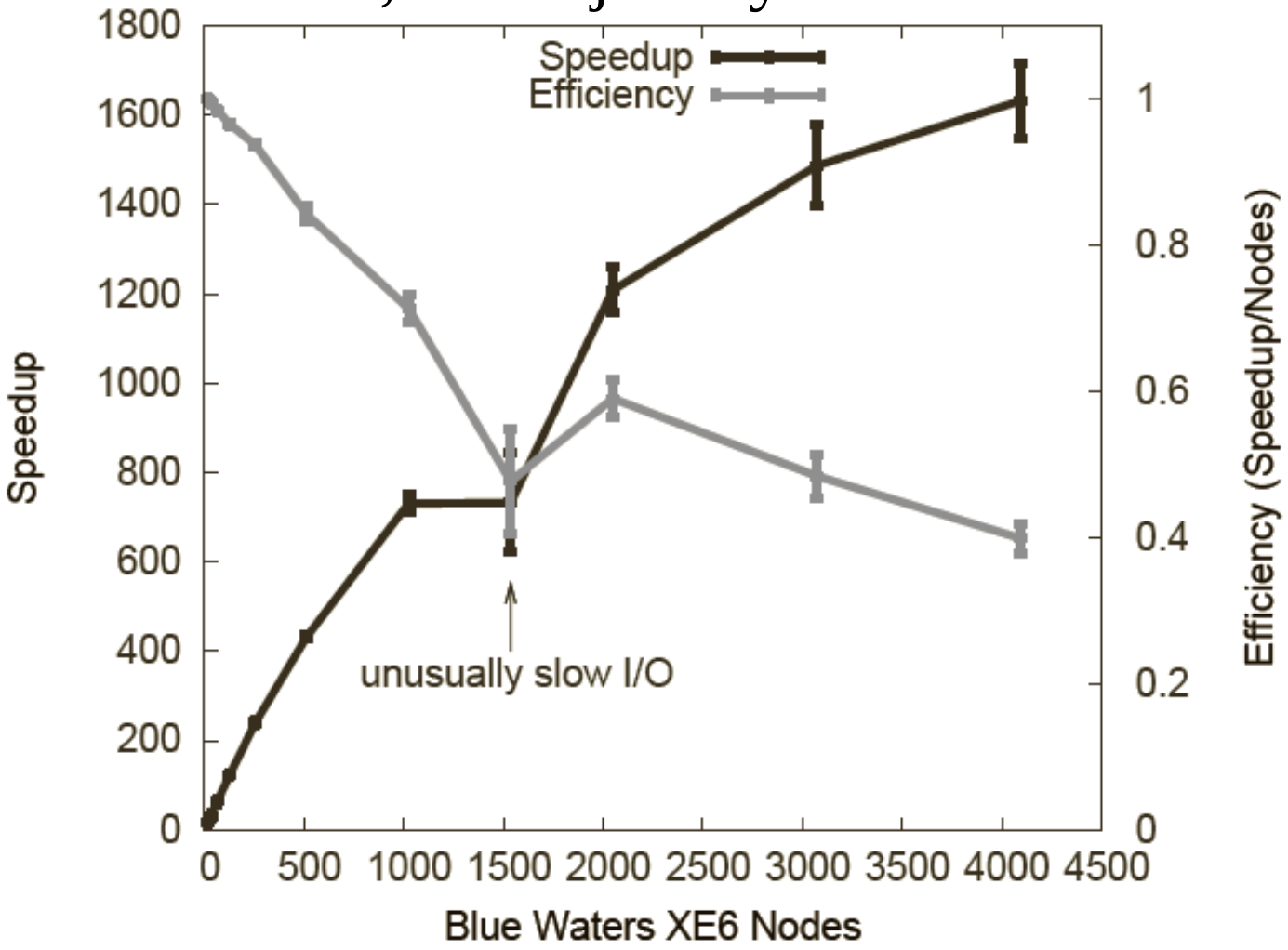


Timeline XE6 Runtimes for Single-Threaded vs. Multithreaded-by-Structure-Component cases



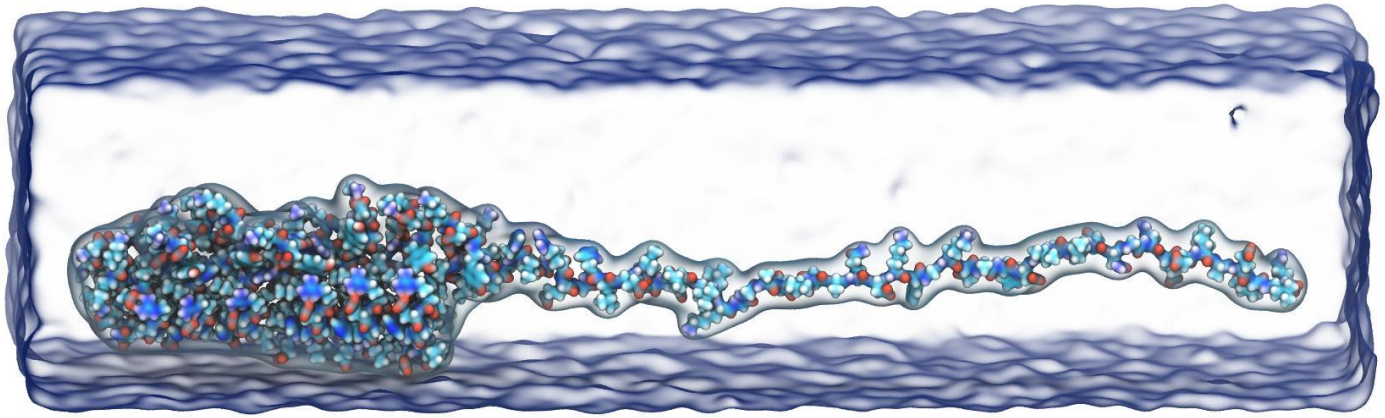
Timeline XE6 Scaling Efficiency:

Multithreaded-by-Structure-Component Case, 16,000 trajectory frames



VMD "QuickSurf" Representation

- Displays continuum of structural detail:
  - All-atom models
  - Coarse-grained models
  - Cellular scale models
  - Multi-scale models: All-atom + CG, Brownian + Whole Cell
  - Smoothly variable between full detail, and reduced resolution representations of very large complexes

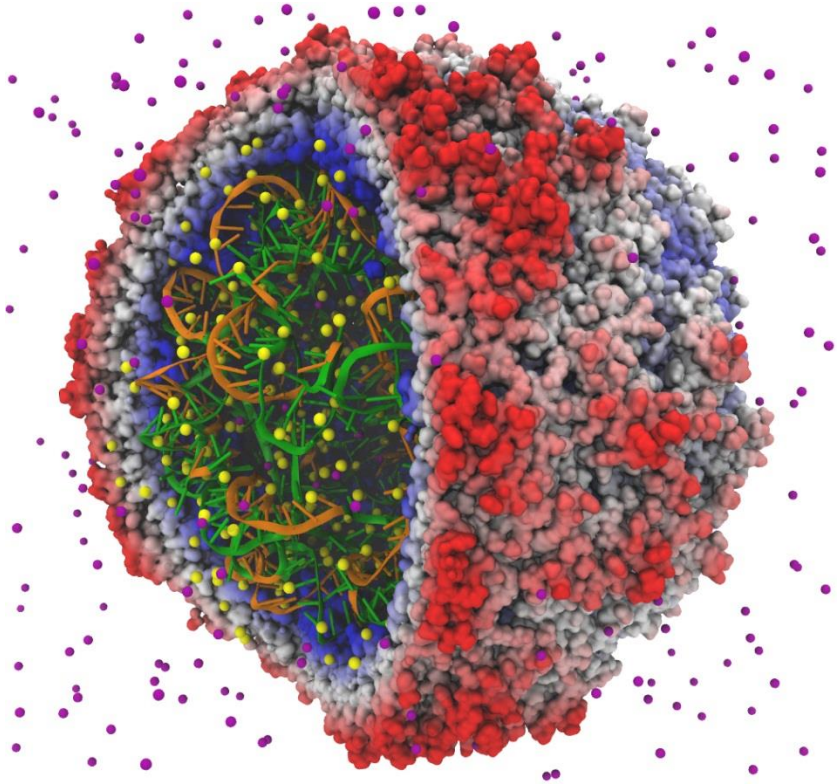


Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.

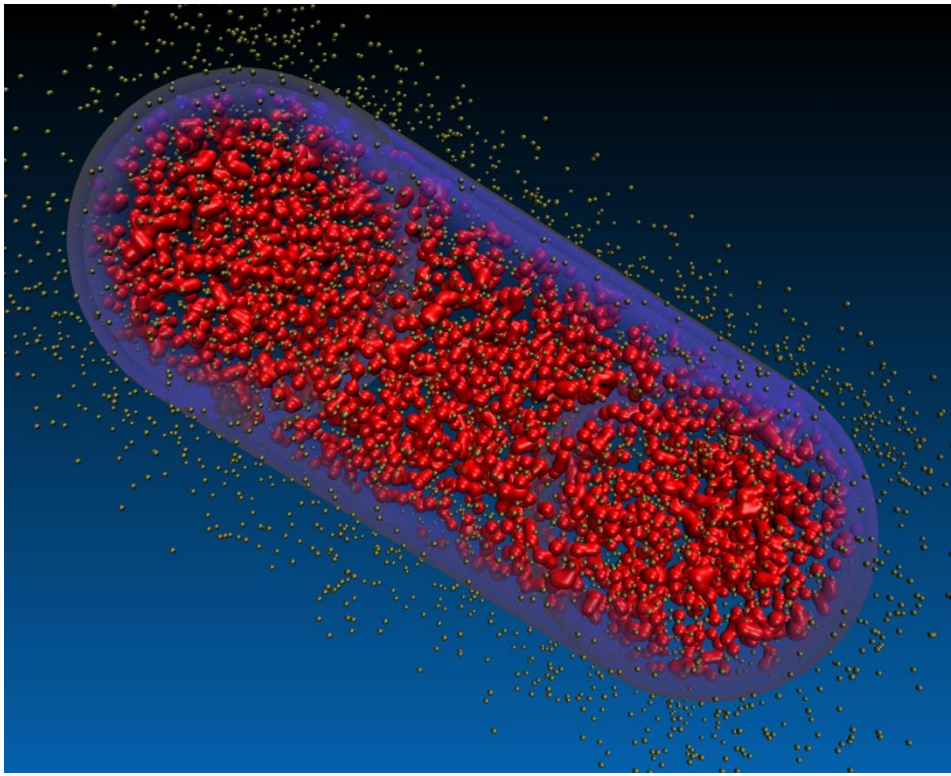
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

VMD “QuickSurf” Representation

- Uses multi-core CPUs and GPU acceleration to enable **smooth real-time animation** of MD trajectories
- Linear-time algorithm, scales to millions of particles, as limited by memory capacity

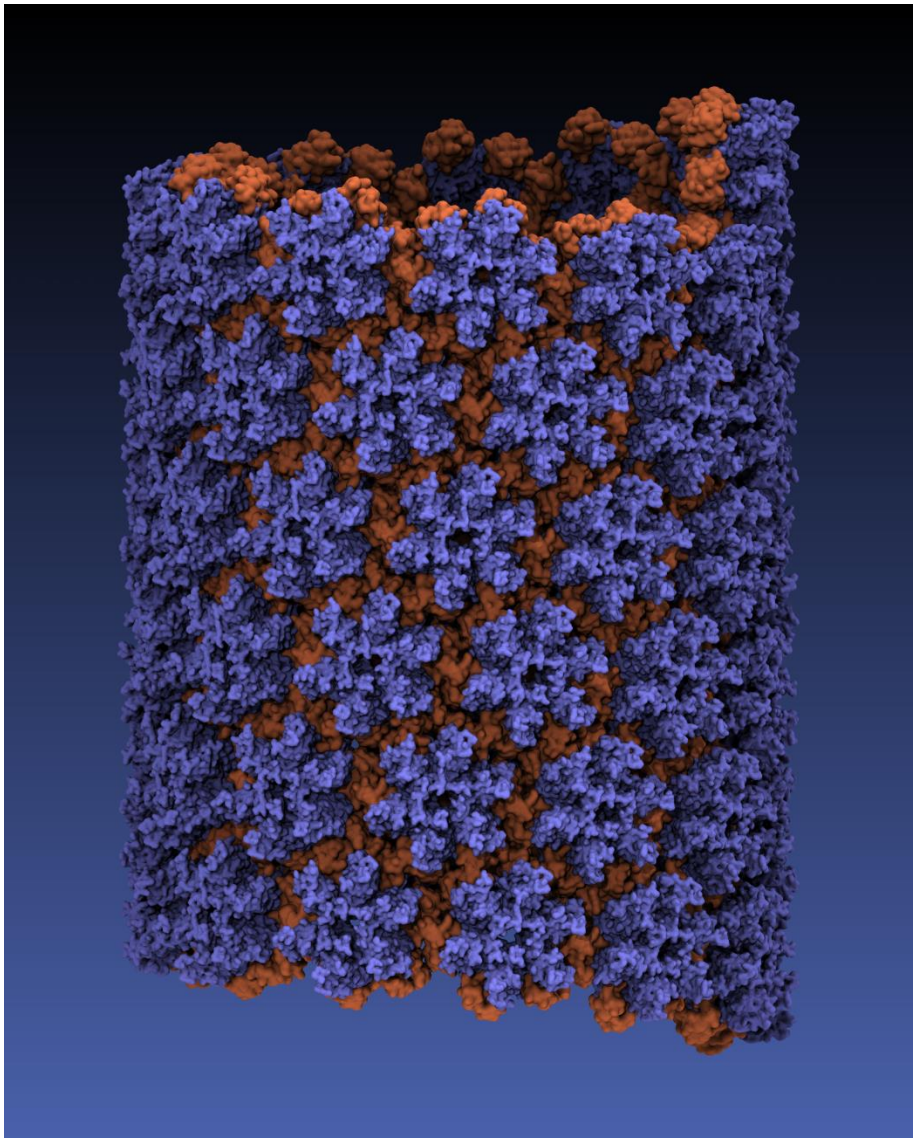


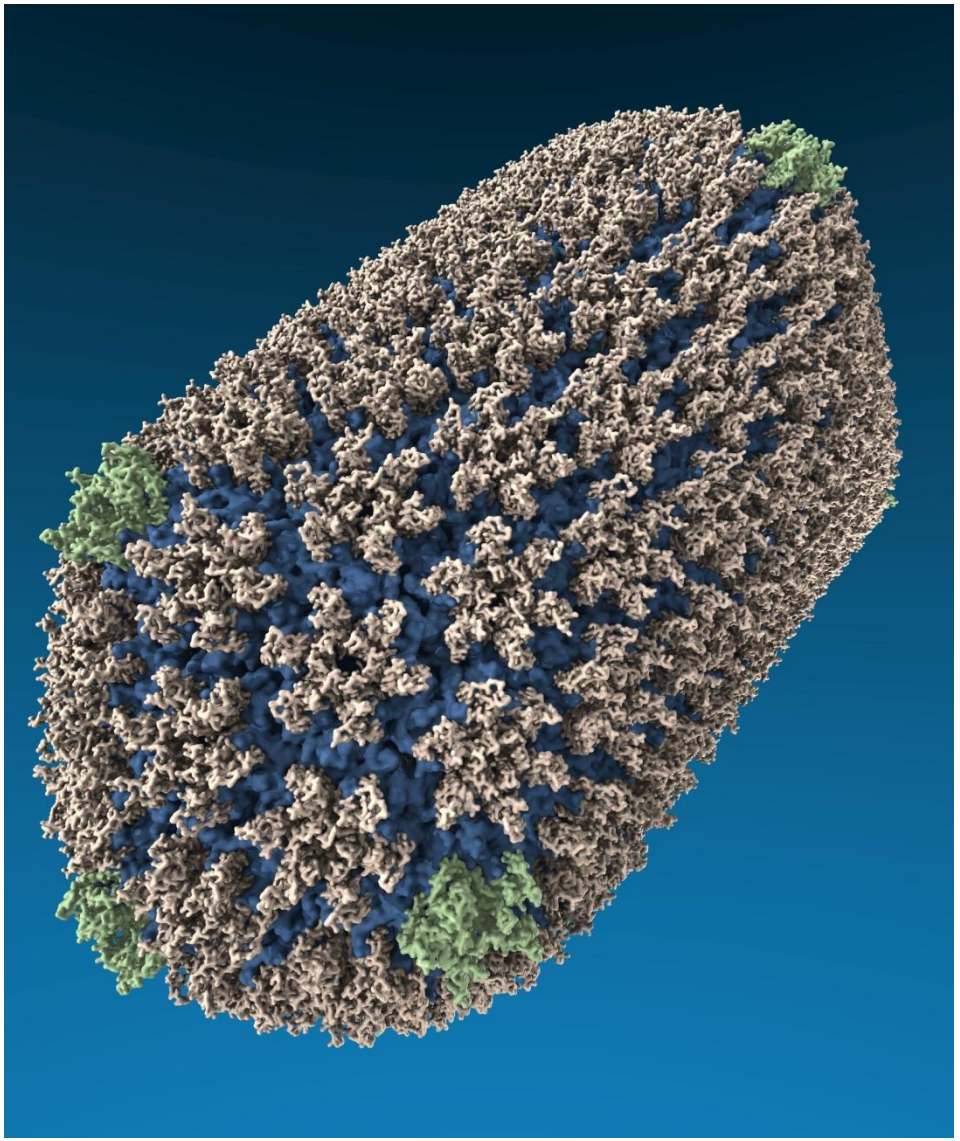
Satellite Tobacco Mosaic Virus



Lattice Cell Simulations

VMD “QuickSurf” Representation





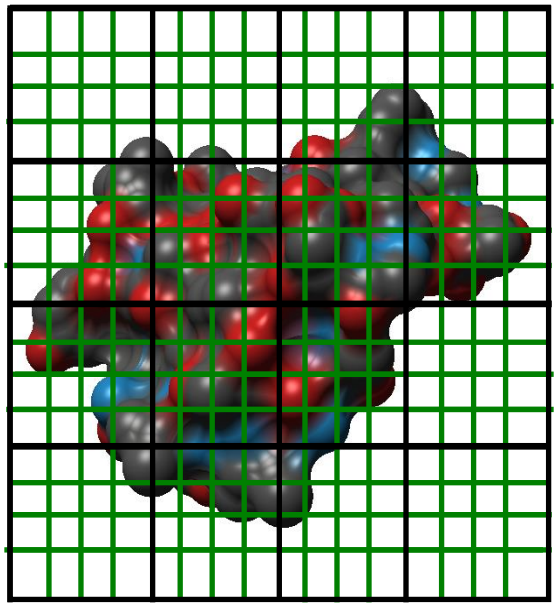
All-atom HIV capsid simulations rendered with VMD/Tachyon

QuickSurf Algorithm Overview

- Build spatial acceleration data structures, optimize data for GPU
- Compute 3-D density map, 3-D volumetric texture map:

$$\rho(\vec{r}; \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \sum_{i=1}^{N} e^{\frac{-|\vec{r}-\vec{r}_i|^2}{2\alpha^2}}$$

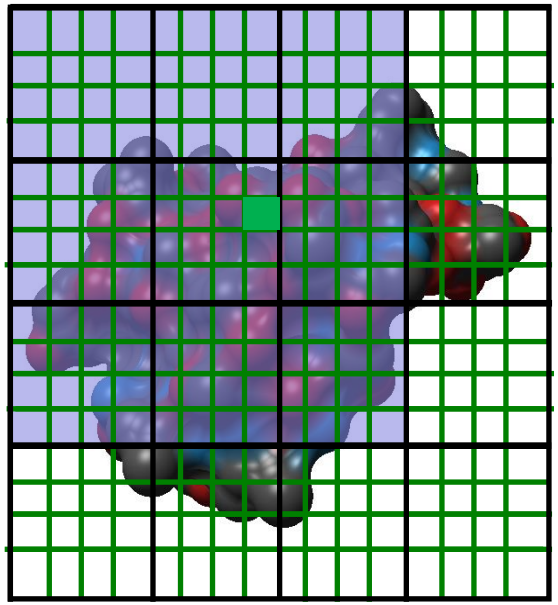
- Extract isosurface for a user-defined density value



3-D density map lattice, spatial acceleration grid, and extracted surface

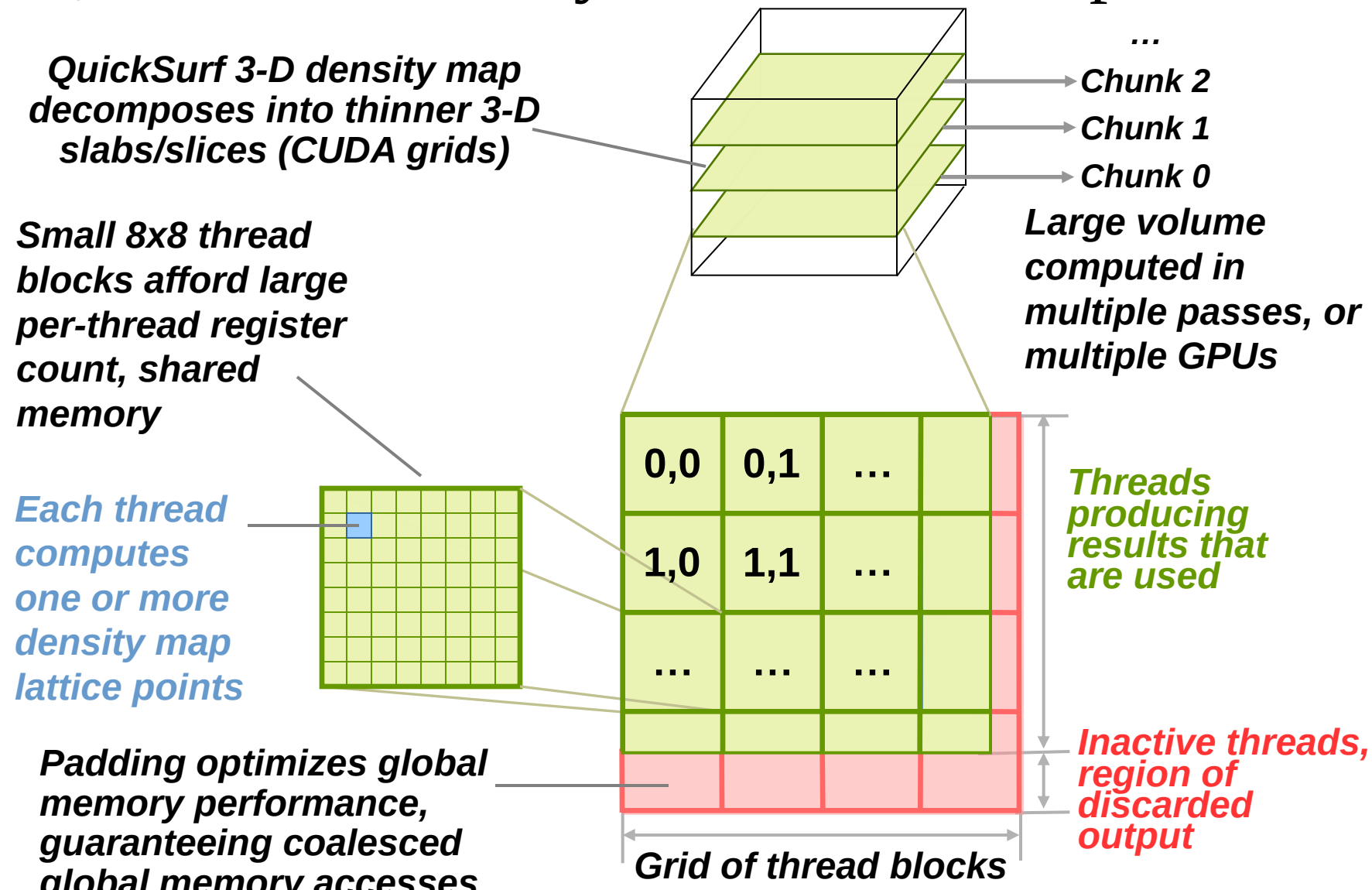
QuickSurf Density Map Algorithm

- Spatial acceleration grid cells are sized to match the cutoff radius for the exponential, beyond which density contributions are negligible
- Density map lattice points computed by summing density contributions from particles in 3x3x3 grid of neighboring spatial acceleration cells
- Volumetric texture map is computed by summing particle colors normalized by their individual density contribution



3-D density map lattice point and the neighboring spatial acceleration cells it references

QuickSurf Density Parallel Decomposition

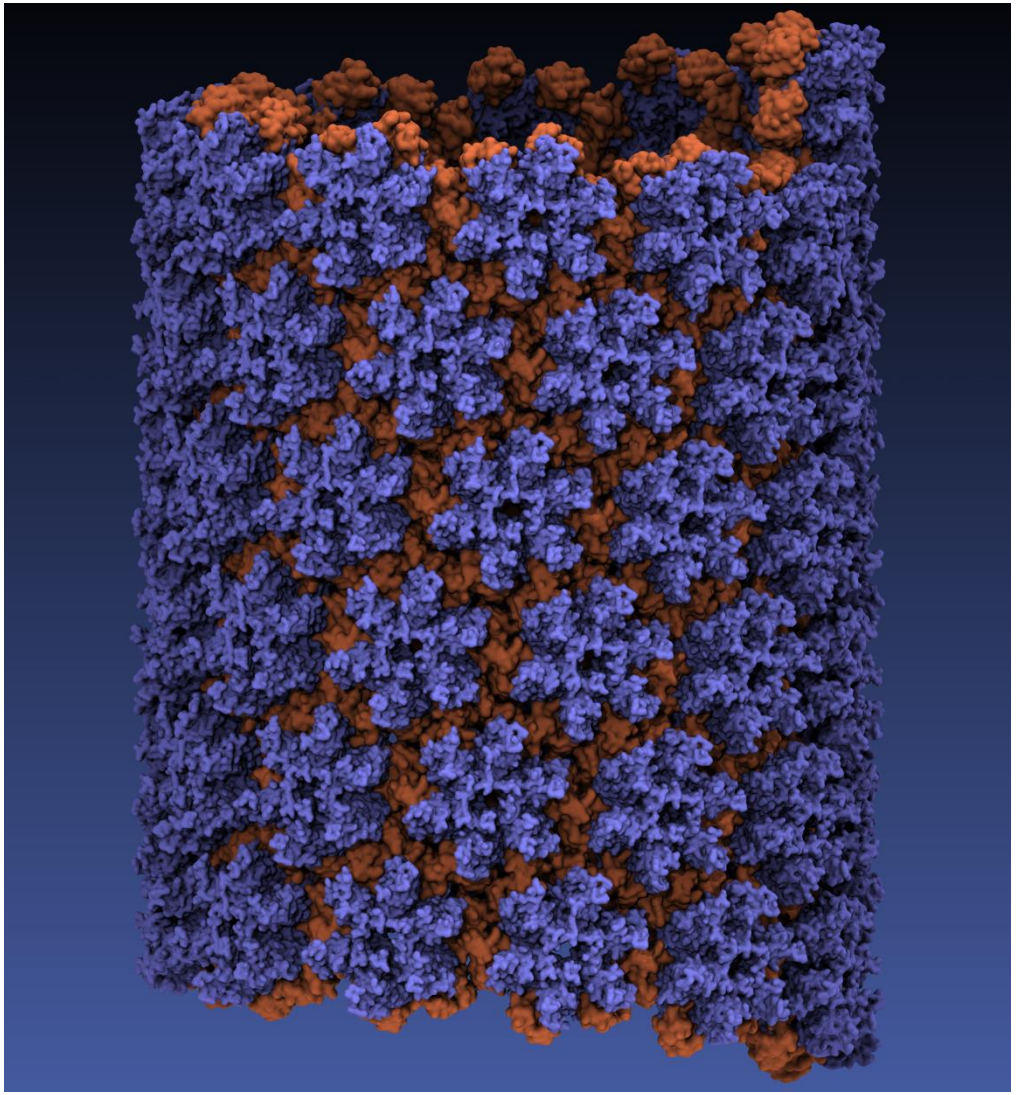


QuickSurf Density Map Kernel Snippet…

```
for (zab=zabmin; zab<=zabmax; zab++) {
  for (yab=yabmin; yab<=yabmax; yab++) {
    for (xab=xabmin; xab<=xabmax; xab++) {
      int abcellidx = zab * acplanesz + yab * acncells.x + xab;
      uint2 atomstartend = cellStartEnd[abcellidx];
      if (atomstartend.x != GRID_CELL_EMPTY) {
        for (unsigned int atomid=atomstartend.x; atomid<atomstartend.y; atomid++) {
          float4 atom = sorted_xyzr[atomid];
          float dx = coorx - atom.x;          float dy = coory - atom.y;        float dz = coorz - atom.z;
          float dxy2 = dx*dx + dy*dy;
          float r21 = (dxy2 + dz*dz) * atom.w;
          densityval1 += exp2f(r21);
          /// Loop unrolling and register tiling benefits begin here……
          float dz2 = dz + gridspacing;
          float r22 = (dxy2 + dz2*dz2) * atom.w;
          densityval2 += exp2f(r22);
          /// More loop unrolling ….
```

Challenge: Support GPU-accelerated QuickSurf for **Large** Biomolecular Complexes

- Structures such as HIV initially needed all XK7 GPU memory to generate detailed surface renderings
- Goals and approach:
  - **Avoid slow CPU-fallback!**
  - Incrementally change algorithm phases to use more compact data types, while maintaining performance
  - Specialize code for different precision/performance/memory capacity cases



Supporting Multiple Data Types for QuickSurf Density Maps and Marching Cubes Vertex Arrays

- The major algorithm components of QuickSurf are now used for many other purposes:
  - Gaussian density map algorithm now used for MDFF Cryo EM density map fitting methods in addition to QuickSurf
  - Marching Cubes routines also used for Quantum Chemistry visualizations of molecular orbitals
- Rather than simply changing QuickSurf to use a particular internal numerical representation, it is desirable to instead use CUDA C++ templates to make type-generic versions of the key objects, kernels, and output vertex arrays
- Accuracy-sensitive algorithms use high-precision data types, performance and memory capacity sensitive cases use quantized or reduced precision approaches

Improving QuickSurf Memory Efficiency

- Both host and GPU memory capacity limitations are a significant concern when rendering surfaces for virus structures such as HIV or for large cellular models which can contain hundreds of millions of particles
- The original QuickSurf implementation used single-precision floating point for output vertex arrays and textures
- Judicious use of reduced-precision numerical representations, cut the overall memory footprint of the entire QuickSurf algorithm to half of the original
  - Data type changes made throughout the entire chain from density map computation through all stages of Marching Cubes

Minimizing the Impact of Generality on QuickSurf Code Complexity

- A critical factor in the simplicity of supporting multiple QuickSurf data types arises from the so-called “gather” oriented algorithm we employ
  - Internally, all in-register arithmetic is single-precision
  - Data conversions to/from compressed or reduced precision data types are performed on-the-fly as needed
- Small inlined type conversion routines are defined for each of the cases we want to support
- Key QuickSurf kernels are genericized using C++ template syntax, and the compiler “connects the dots” to automatically generate type-specific kernels as needed

Example Templated Density Map Kernel

```
template<class DENSITY, class VOLTEX>
__global__ static void
gaussdensity_fast_tex_norm(int natoms,
                           const float4 * RESTRICT sorted_xyzr,
                           const float4 * RESTRICT sorted_color,
                           int3 numvoxels,
                           int3 acncells,
                          float acgridspacing,
                          float invacgridspacing,
                          const uint2 * RESTRICT cellStartEnd,
                          float gridspacing, unsigned int z,
                          DENSITY * RESTRICT densitygrid,
                          VOLTEX * RESTRICT voltexmap,
                         float invisovalue) {
```

Example Templated Density Map Kernel

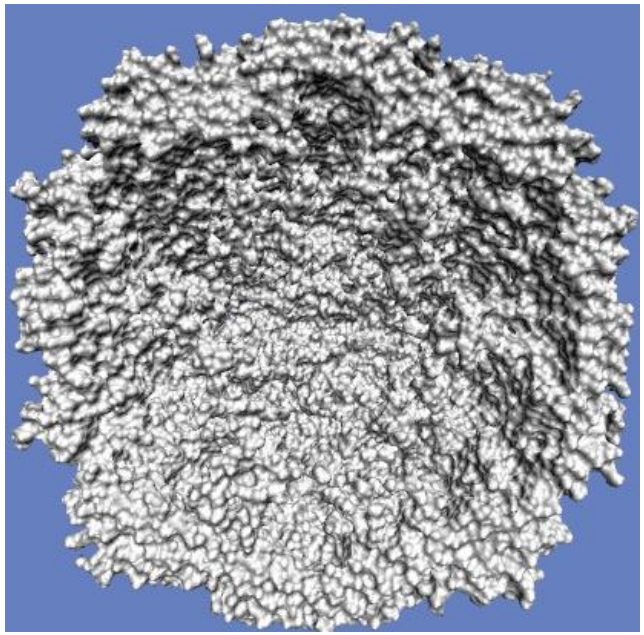
```
template<class DENSITY, class VOLTEX>
__global__ static void
gaussdensity_fast_tex_norm( …  ) {

  … Triple-nested and unrolled inner loops here …

  DENSITY densityout;
  VOLTEX texout;
  convert_density(densityout, densityval1);
  densitygrid[outaddr         ] = densityout;
  convert_color(texout, densitycol1);
  voltexmap[outaddr         ] = texout;
```

Net Result of QuickSurf Memory Efficiency Optimizations

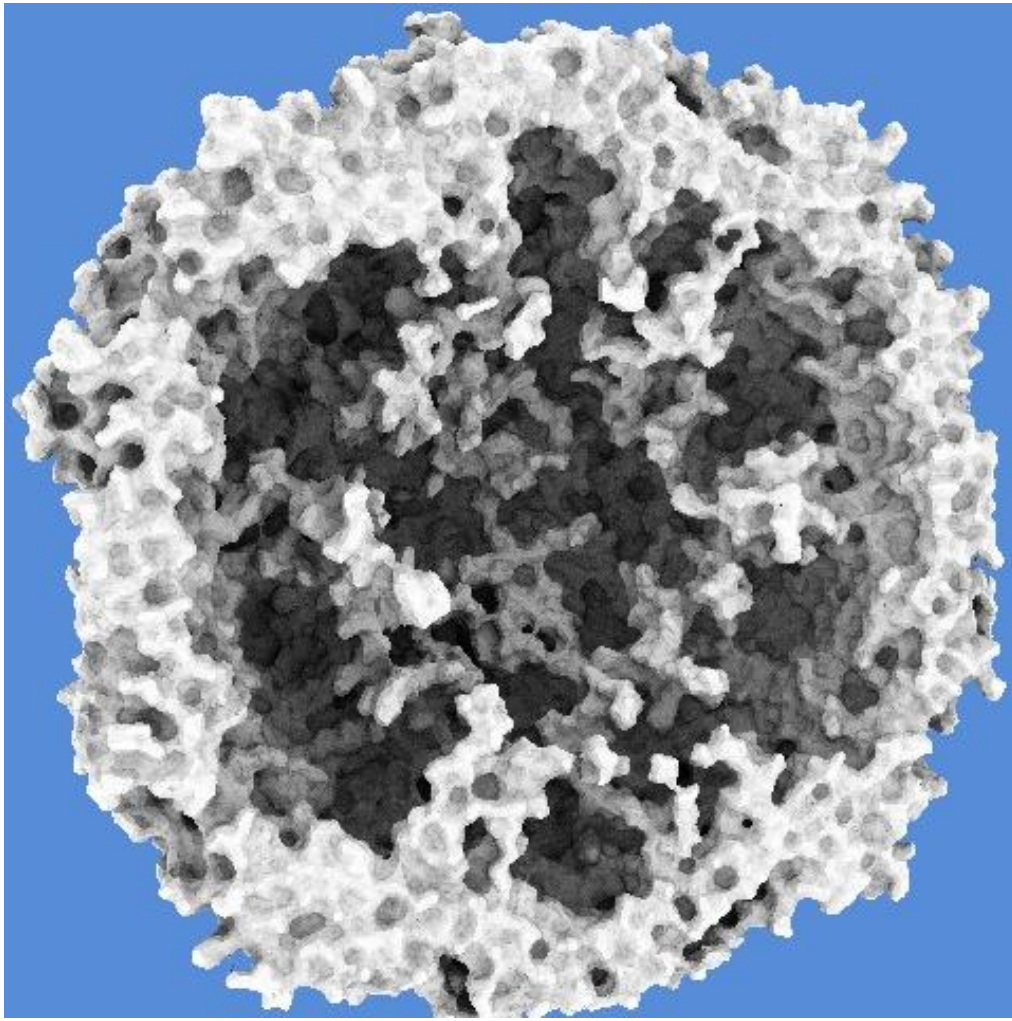
- **Halved** overall GPU memory use
- Achieved **1.5x to 2x performance gain**:
  - The "gather" density map algorithm keeps type conversion operations out of the innermost loop
  - Density map global memory writes reduced to half
  - Multiple stages of Marching Cubes operate on smaller input and output data types
  - Same code path supports multiple precisions
- Users now get full GPU-accelerated QuickSurf in many cases that previously triggered CPU-fallback, all platforms (laptop/desk/super) benefit!

Ray Tracing Molecular Graphics (& w/ OptiX)

- Ambient occlusion lighting, shadows, reflections, transparency, and more…
- Satellite tobacco mosaic virus capsid w/ ~75K atoms

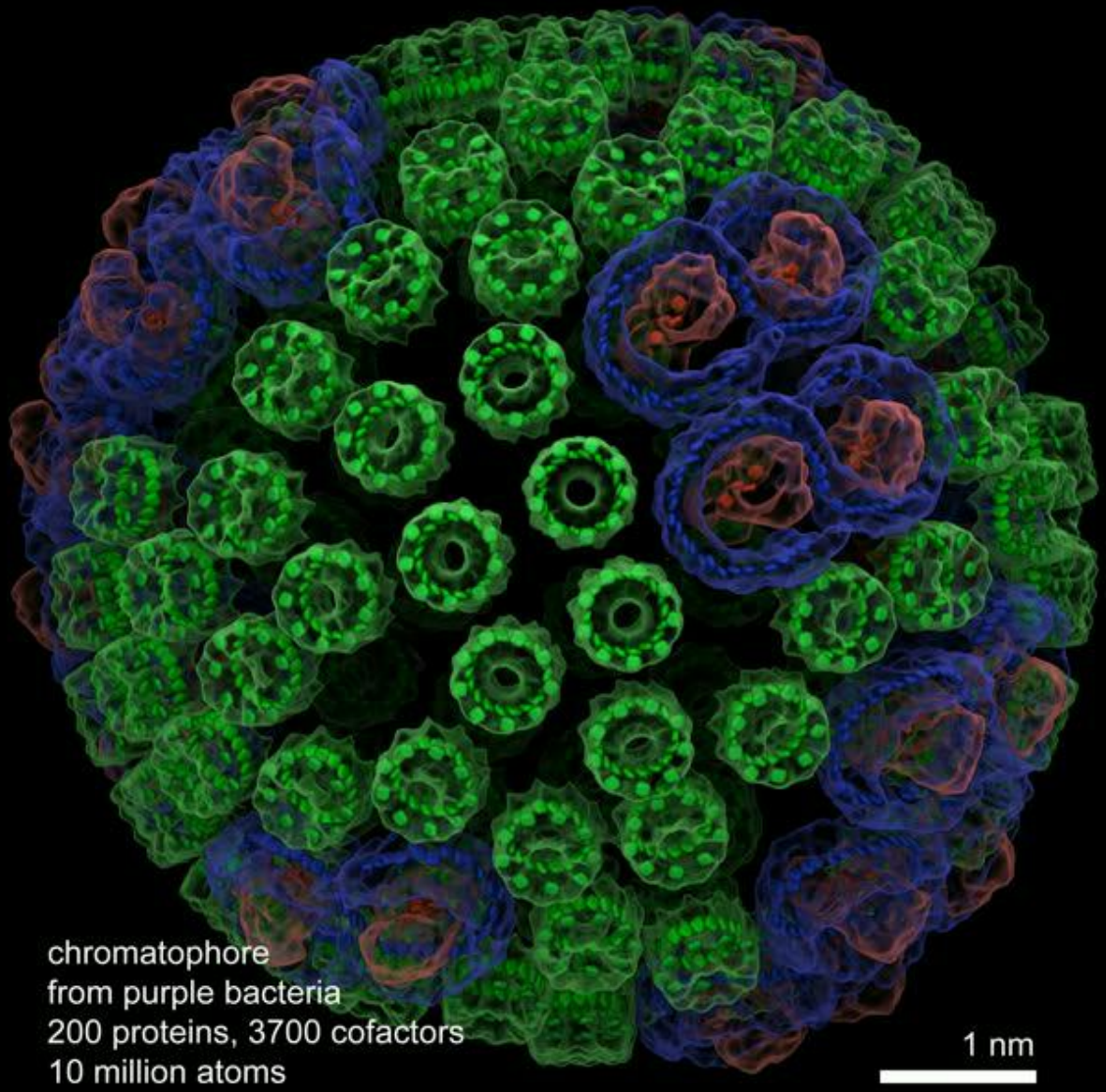


Standard OpenGL rasterization



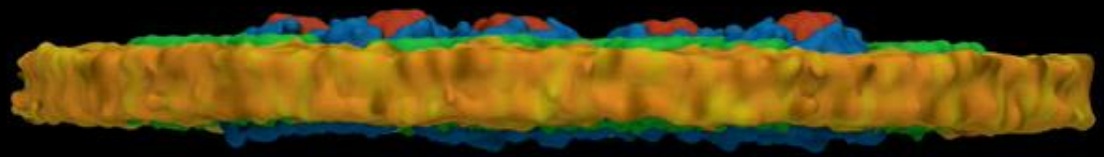
Prototype VMD/OptiX GPU ray tracing w/ ambient occlusion lighting

BW-ESS VMD/Tachyon Movie Generation



480 XE6 nodes for 85m @ 4096x2400

BW-ESS VMD/Tachyon Movie Generation



20 M atom chromatophore patch

360 XE6 nodes for 3h50m @ 4096x2400

Parallel Movie Rendering Results

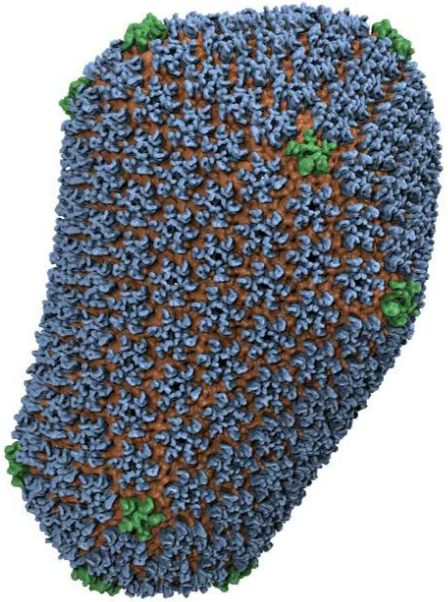
- Unexpected I/O overhead from sourcing scripts!
- XK7 CUDA algorithms reduce per-frame surface and other geometry calculation times by a factor of ~15 vs. multithreaded SSE CPU code on XE6 nodes
- OpenGL rasterization is so fast it is essentially “free” – I/O time dominates OpenGL test cases currently... (XK7 partition had no I/O nodes)
- For Tachyon (CPU-only) XE6 nodes render almost exactly 2x faster than XK7 nodes
- All test cases start to be penalized at >= 512 nodes due to increased I/O contention for common input files, reading of scripts, etc – need broadcast scheme for this data

Parallel Movie Generation

Movie Resolution	Rendering Mode	Node Type	Nodes	Wall Clock Execution Time			
				Script Loading	State Loading	Geometry and Rendering	Total
"PowerPoint" 1057 × 652 689,164 pixels	OpenGL rasterization	XK7	16	2 s	152 s	99 s	253 s
		XK7	32	2 s	158 s	45 s	205 s
		XK7	64	2 s	167 s	20 s	189 s
		XK7	128	2 s	191 s	11 s	205 s
		XK7	256	6 s	244 s	5.4 s	255 s
		XK7	512	7 s	302 s	2.5 s	312 s
	In-place Tachyon ray tracing w/ ambient occlusion (AO) lighting	XK7	256	4 s	225 s	918 s	1,147 s
		XK7	512	9 s	292 s	532 s	834 s
		XE6	128	2 s	83 s	943 s	1,029 s
		XE6	256	4 s	125 s	560 s	692 s
		XE6	512	7 s	221 s	330 s	560 s
	Combined OpenGL rasterization and Tachyon ray tracing w/ AO	XK7	256	4 s	214 s	913 s	1,170 s
		XK7	512	9 s	300 s	531 s	848 s
4K UltraHD 3840 × 2160 8,294,400 pixels	OpenGL rasterization	XK7	512	9 s	300 s	3.1 s	314 s
	Combined OpenGL rasterization and Tachyon ray tracing w/ AO	XK7	512	9 s	295 s	5,828 s	6,133 s
No Image Output	Tesla K20X CUDA Geometry Calc.	XK7	512	7 s	188 s	1.5 s	197 s
	CPU Geometry Calc.	XE6	512	7 s	214 s	23 s	244 s

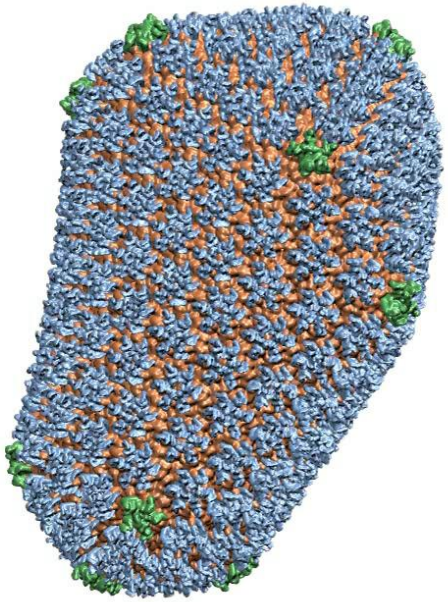
TABLE II. VMD PARALLEL MOVIE RENDERING PERFORMANCE TESTS.

Tachyon w/
Ambient
Occlusion

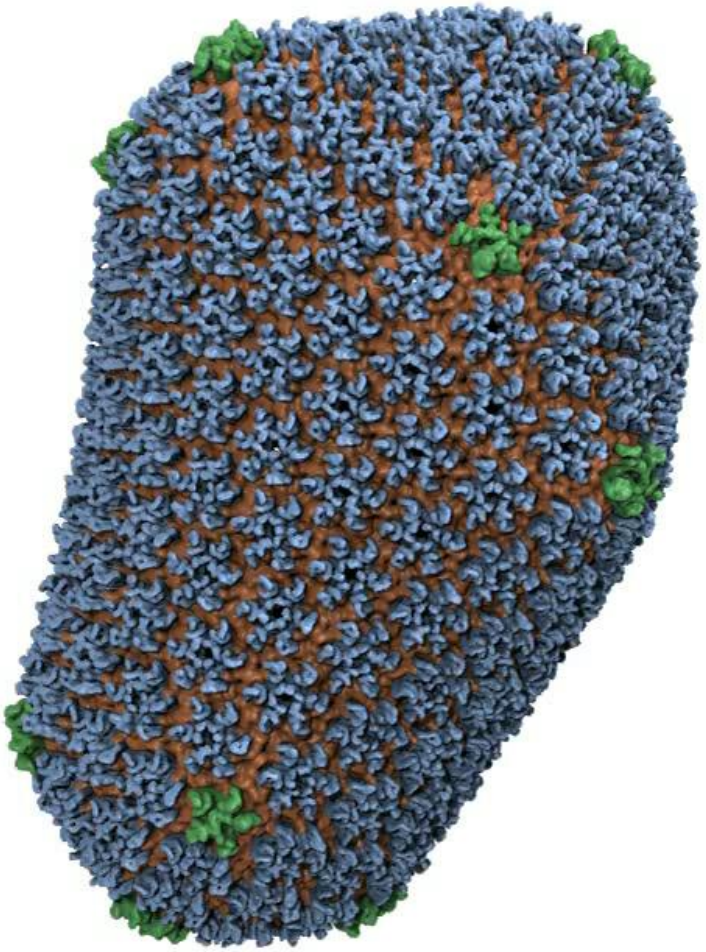


64M atom
HIV-1 capsid

OpenGL
GLSL



OpenGL as a “Failsafe” for Movie Segments with Shadowing Problems



Ongoing VMD Development Work

- Updated GPU kernels for XK7
- Adjust Timeline work granularity
- Eliminate remaining cases of duplicated I/O during startup phases by using internal broadcasts
- XK7 GPU ray tracing with OptiX toolkit
- Re-test I/O and other results on BW after hardware upgrades…

Upgraded Blue Waters
Back Online
Beginning Today 8/15!

Acknowledgements

- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- NCSA Blue Waters Team
- NVIDIA CUDA Center of Excellence, University of Illinois at Urbana-Champaign
- Many of the staff at NVIDIA and Cray
- Funding:
  - NSF OCI 07-25070
  - NSF PRAC “The Computational Microscope”
  - NIH support: 9P41GM104601, 5R01GM098243-02

GPU Computing Publications

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

- **Lattice Microbes: High-performance stochastic simulation method for the reaction-diffusion master equation.**
  E. Roberts, J. E. Stone, and Z. Luthey-Schulten.
  J. Computational Chemistry 34 (3), 245-255, 2013**.**
- **Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.** M. Krone, J. E. Stone, T. Ertl, and K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012.
- **Immersive Out-of-Core Visualization of Large-Size and Long-Timescale Molecular Dynamics Trajectories.** J. Stone, K. Vandivort, and K. Schulten. G. Bebis et al. (Eds.): *7th International Symposium on Visual Computing (ISVC 2011)*, LNCS 6939, pp. 1-12, 2011.
- **Fast Analysis of Molecular Dynamics Trajectories with Graphics Processing Units – Radial Distribution Functions.** B. Levine, J. Stone, and A. Kohlmeyer. *J. Comp. Physics*, 230(9):3556-3569, 2011.

GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/

- **Quantifying the Impact of GPUs on Performance and Energy Efficiency in HPC Clusters.** J. Enos, C. Steffen, J. Fullop, M. Showerman, G. Shi, K. Esler, V. Kindratenko, J. Stone, J Phillips. *International Conference on Green Computing*, pp. 317-324, 2010.
- **GPU-accelerated molecular modeling coming of age. J. Stone, D. Hardy, I. Ufimtsev, K. Schulten.** *J. Molecular Graphics and Modeling*, 29:116-125, 2010.
- **OpenCL: A Parallel Programming Standard for Heterogeneous Computing. J. Stone, D. Gohara, G. Shi.** *Computing in Science and Engineering*, 12(3):66-73, 2010.
- **An Asymmetric Distributed Shared Memory Model for Heterogeneous Computing Systems**. I. Gelado, J. Stone, J. Cabezas, S. Patel, N. Navarro, W. Hwu. *ASPLOS '10: Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 347-358, 2010.

GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/

- **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)*, In Proceedings IEEE Cluster 2009, pp. 1-8, Aug. 2009.
- **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.
- **Probing Biomolecular Machines with Graphics Processors**. J. Phillips, J. Stone. *Communications of the ACM*, 52(10):34-41, 2009.
- **Multilevel summation of electrostatic potentials using graphics processing units**. D. Hardy, J. Stone, K. Schulten. *J. Parallel Computing*, 35:164-177, 2009.

GPU Computing Publications
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.