GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms

John E. Stone, Kirby L. Vandivort, 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/

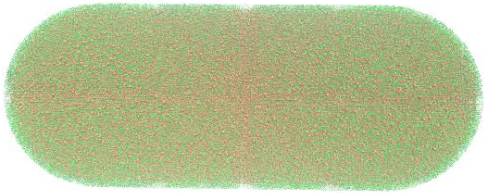
http://doi.acm.org/10.1145/2535571.2535595

UltraVis'13: Eighth Ultrascale Visualization Workshop

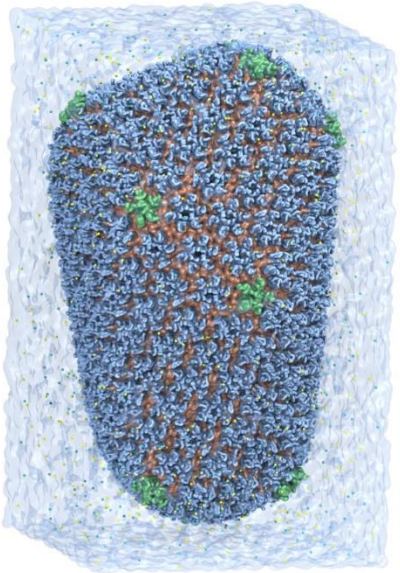
Denver, CO, November 17, 2013

VMD – “Visual Molecular Dynamics”

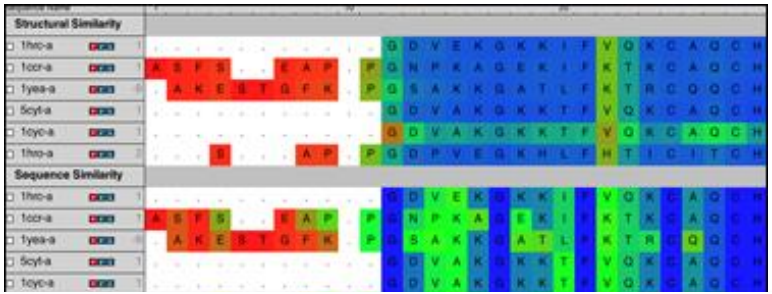
- Visualization and analysis of:
  - molecular dynamics simulations
  - particle systems and whole cells
  - cryoEM densities, volumetric data
  - quantum chemistry calculations
  - sequence information
- User extensible w/ scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/



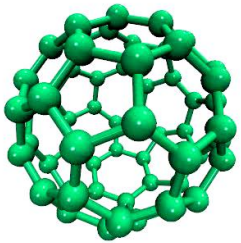
Whole Cell Simulation



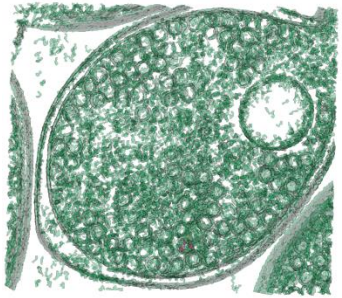
MD Simulations



Sequence Data



Quantum Chemistry

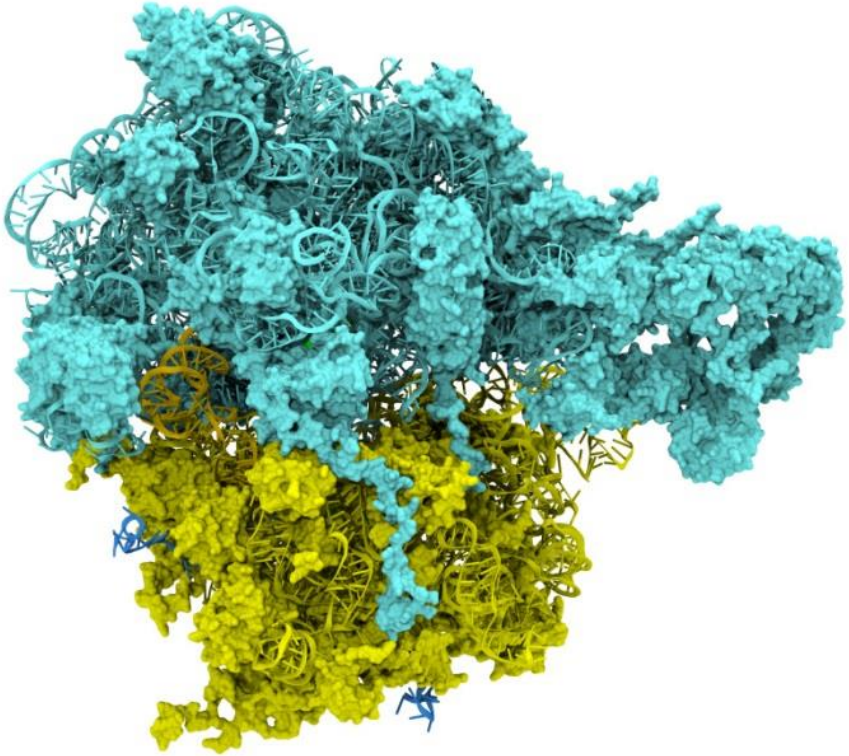


CryoEM, Cellular Tomography

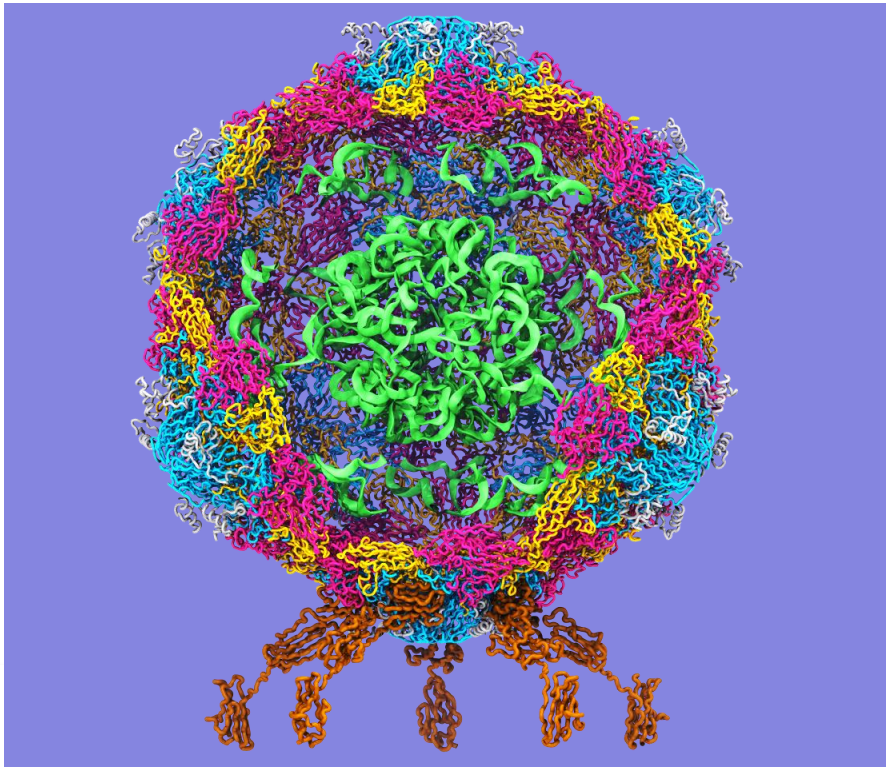
Goal: A Computational Microscope

Study the molecular machines in living cells

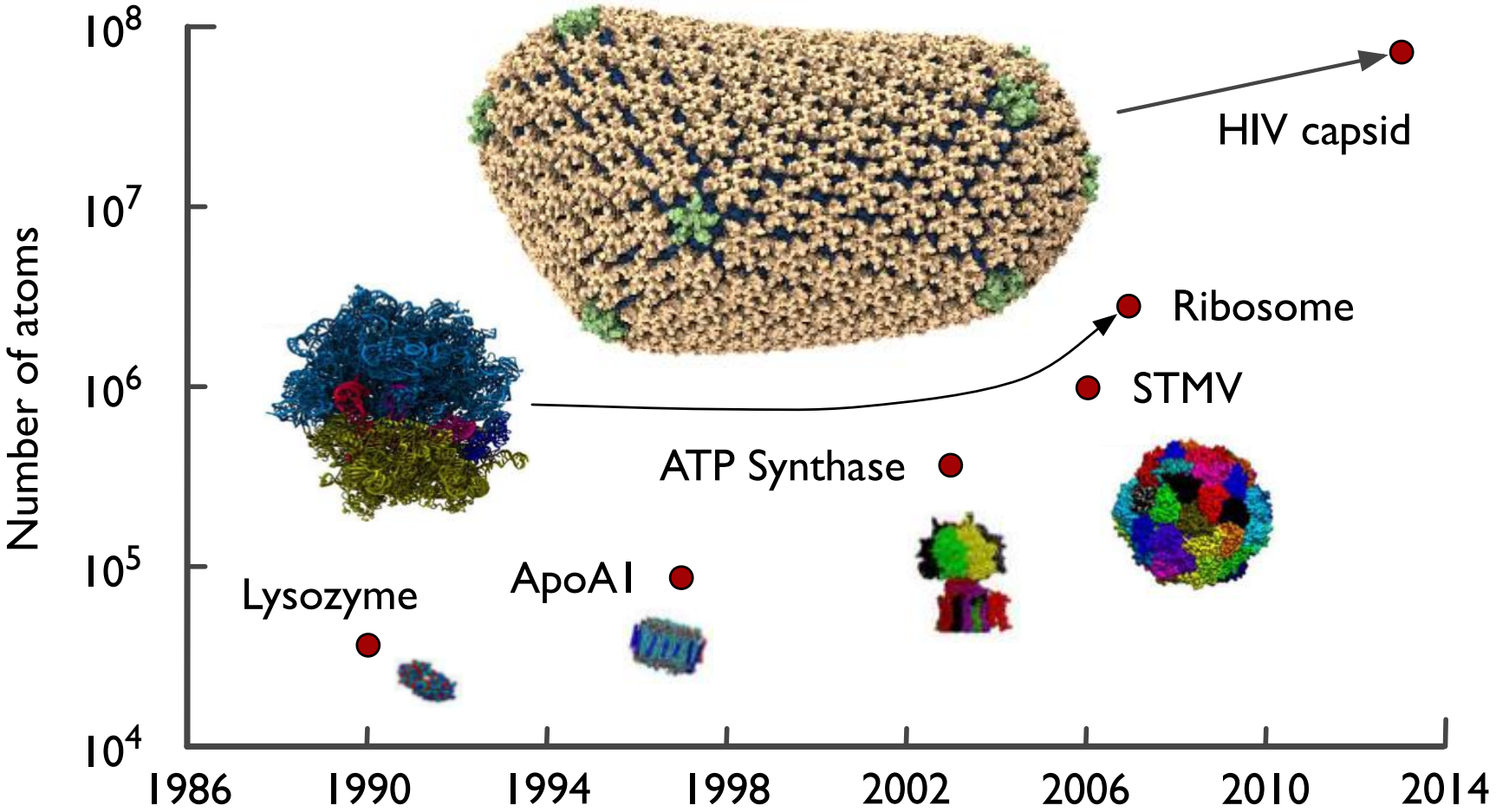
Ribosome: target for antibiotics



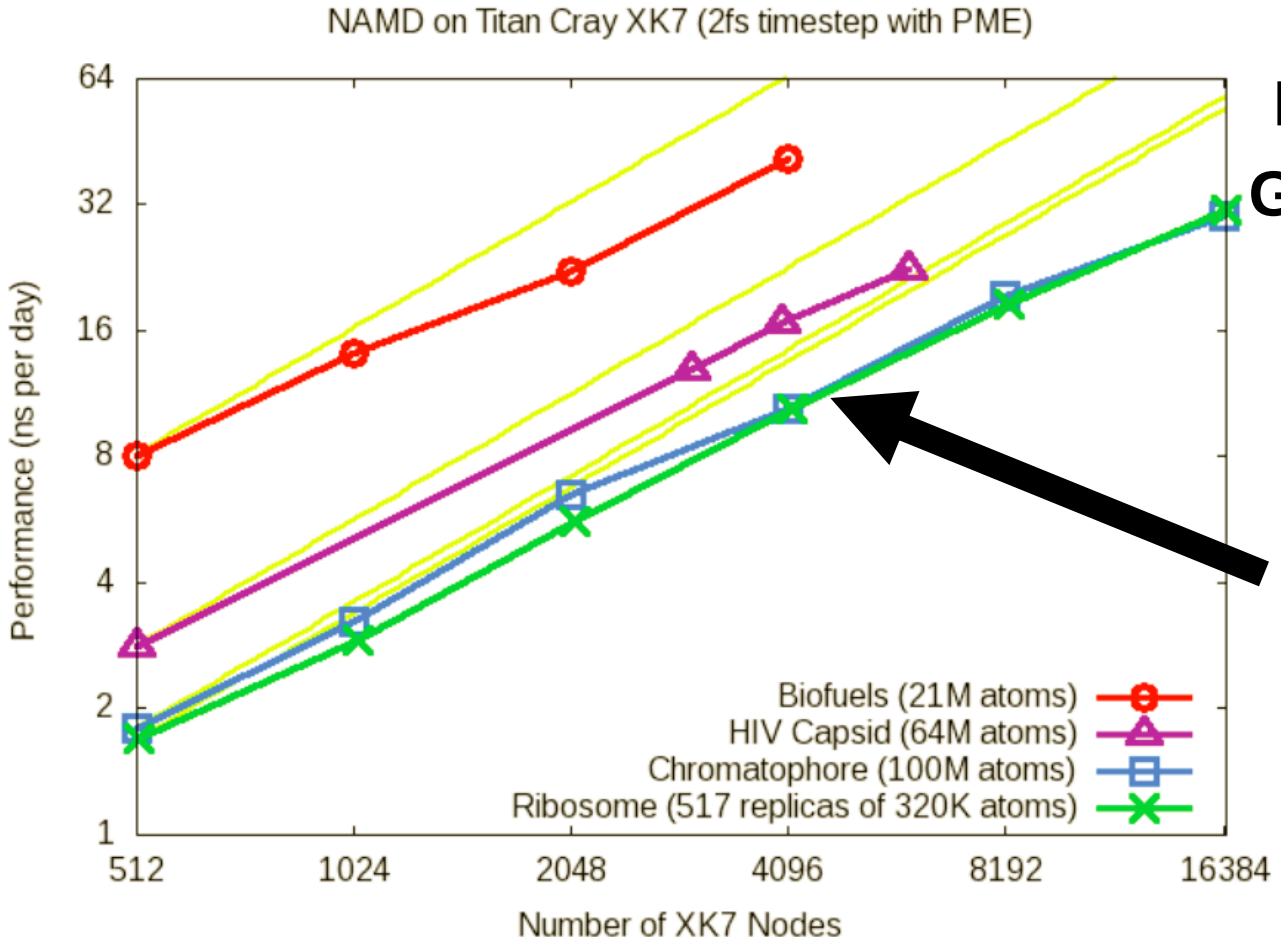
Poliovirus



Computational Biology's Insatiable Demand for Processing Power



NAMD Titan XK7 Performance August 2013



NAMD XK7 vs. XE6

GPU Speedup: 3x-4x

HIV-1 Trajectory: ~1.2 TB/day @ 4096 XK7 nodes

VMD Petascale Visualization and Analysis

- Analyze/visualize large trajectories too large to transfer off-site:
  - 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 8192 XE6 nodes (262,144 CPU cores), viz. runs w/ up to 512 XK7 nodes (K20X GPUs)
- **Supports GPU-accelerated Cray XK7 nodes for both visualization and analysis:**
  - GPU accelerated trajectory analysis w/ CUDA
  - OpenGL and OptiX ray tracing for visualization and movie rendering

NCSA Blue Waters Hybrid Cray XE6 / XK7

22,640 XE6 dual-Opteron CPU nodes

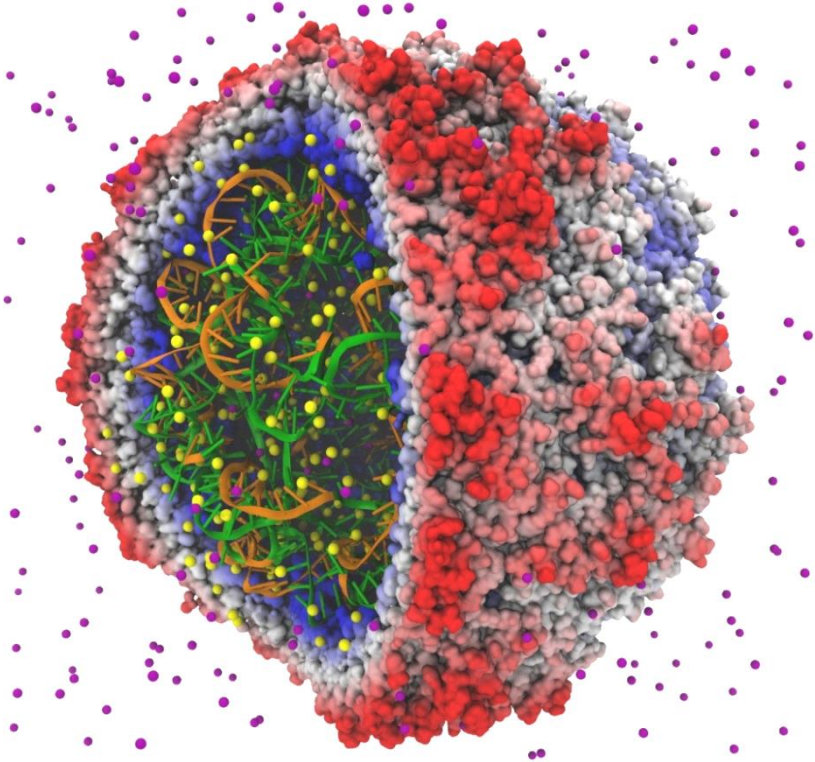
4,224 XK7 nodes w/ Telsa K20X GPUs

Visualization Goals, Challenges

- Increased GPU acceleration for visualization of **petascale molecular dynamics trajectories**
- **Overcome GPU memory capacity limits**, enable high quality visualization of >100M atom systems
- Use GPU to accelerate not only interactive-rate visualizations, but also photorealistic ray tracing with **artifact-free ambient occlusion lighting**, etc.
- Maintain **ease-of-use**, intimate link to VMD analytical features, atom selection language, etc.

VMD “QuickSurf” Representation

- Displays continuum of structural detail:
  - All-atom, coarse-grained, cellular models
  - Smoothly variable detail controls
- Linear-time algorithm, scales to millions of particles, as **limited by memory capacity**
- Uses multi-core CPUs and GPU acceleration to enable **smooth interactive animation** of molecular dynamics trajectories w/ ~1-2 million atoms
- GPU acceleration yields 10x-15x speedup vs. multi-core CPUs



Satellite Tobacco Mosaic Virus

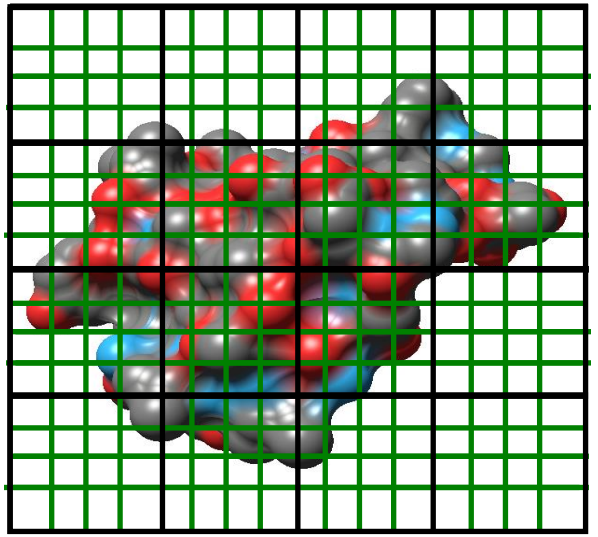
Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

QuickSurf Algorithm Improvements

- **50%-66% memory use, 1.5x-2x speedup**
- Build spatial acceleration data structures, optimize data for GPU
- Compute 3-D density map, 3-D color texture map with **data-parallel *"gather"* algorithm**:

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

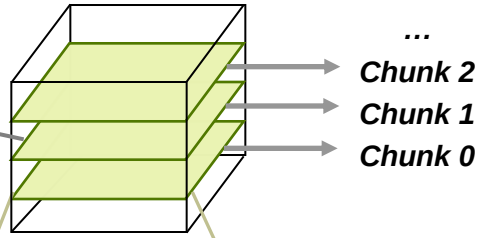
- Normalize, quantize, and compress density, color, surface normal data **while in registers**, before writing out to GPU global memory
- Extract isosurface, maintaining quantized/compressed data representation



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

QuickSurf Density Calc. Parallel Decomposition

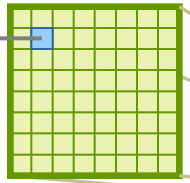
QuickSurf 3-D density map decomposes into thinner 3-D slabs/slices (CUDA grids)

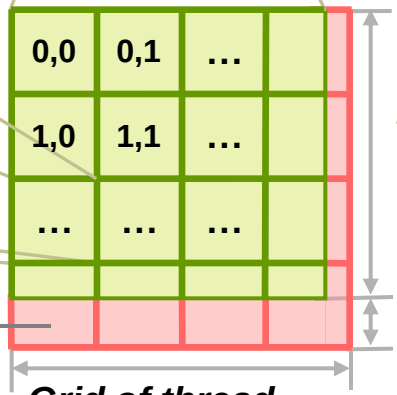


Large volume computed in multiple passes

Small 8x8 thread blocks afford large per-thread register count, shared memory

Each thread computes 1, 4, or 8 density map lattice points; register tiling increases operand bandwidth





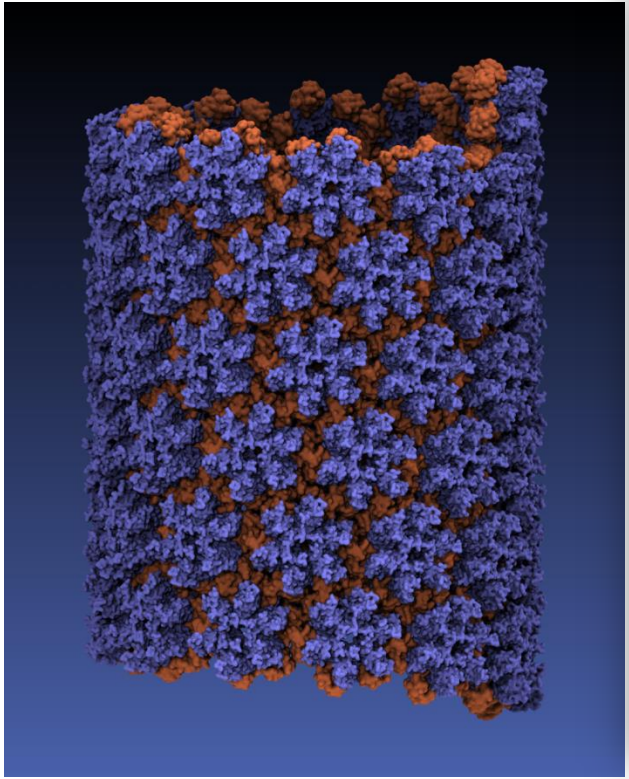
Threads producing results that are used

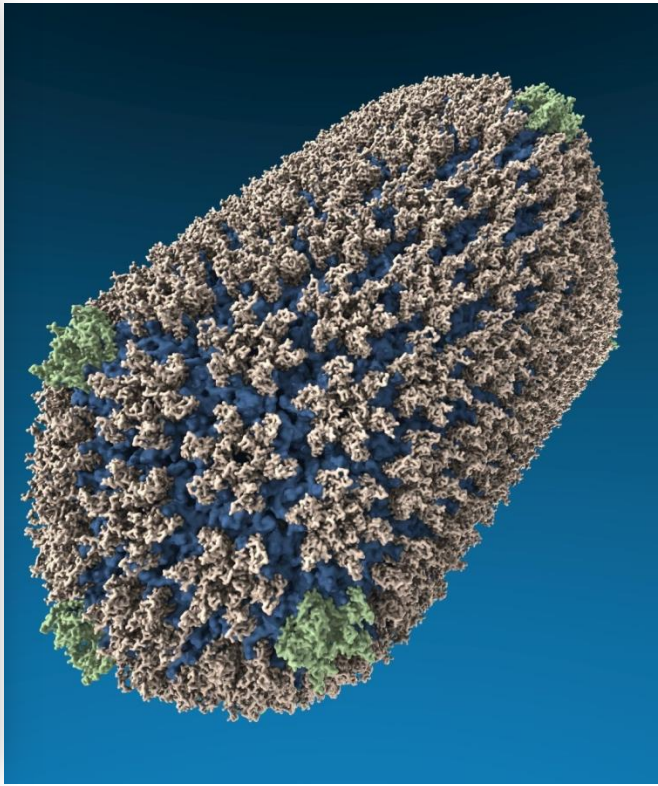
Inactive threads, region of discarded output

Padding optimizes global memory performance, guaranteeing coalesced global memory accesses

Grid of thread blocks

VMD “QuickSurf” Representation





All-atom HIV capsid simulations w/ up to 64M atoms

Net Result of QuickSurf Memory Efficiency Optimizations

- **Roughly halved** overall GPU memory use
- Achieved **1.5x to 2x performance gain**:
  - The ***"gather"*** density map algorithm keeps type conversions out of the innermost loops
  - Density map global memory writes **reduced to about half**
  - Marching cubes and later rendering steps all operate on smaller input and output data types
  - **Same code supports multiple precisions, multiple memory formats using CUDA support for C++ templates**
- Users now get full GPU-accelerated QuickSurf in many cases that previously triggered CPU-fallback, all platforms (laptop/desk/super) benefit!

VMD GPU-Accelerated Ray Tracing Engine: "TachyonL-OptiX"

- Complementary to VMD OpenGL GLSL renderer that uses fast, interactivity-oriented rendering techniques
- Key ray tracing benefits: ambient occlusion lighting, shadows, high quality transparent surfaces, …
  - Subset of Tachyon parallel ray tracing engine in VMD
  - GPU acceleration w/ CUDA+OptiX ameliorates long rendering times associated with advanced lighting and shading algorithms
    - **Ambient occlusion generates large secondary ray workload**
    - **Transparent surfaces and transmission rays can increase secondary ray counts by another order of magnitude**
  - Adaptation of Tachyon to the GPU required careful avoidance of GPU branch divergence, use of GPU memory layouts, etc.

VMD w/ OpenGL GLSL vs. GPU Ray Tracing

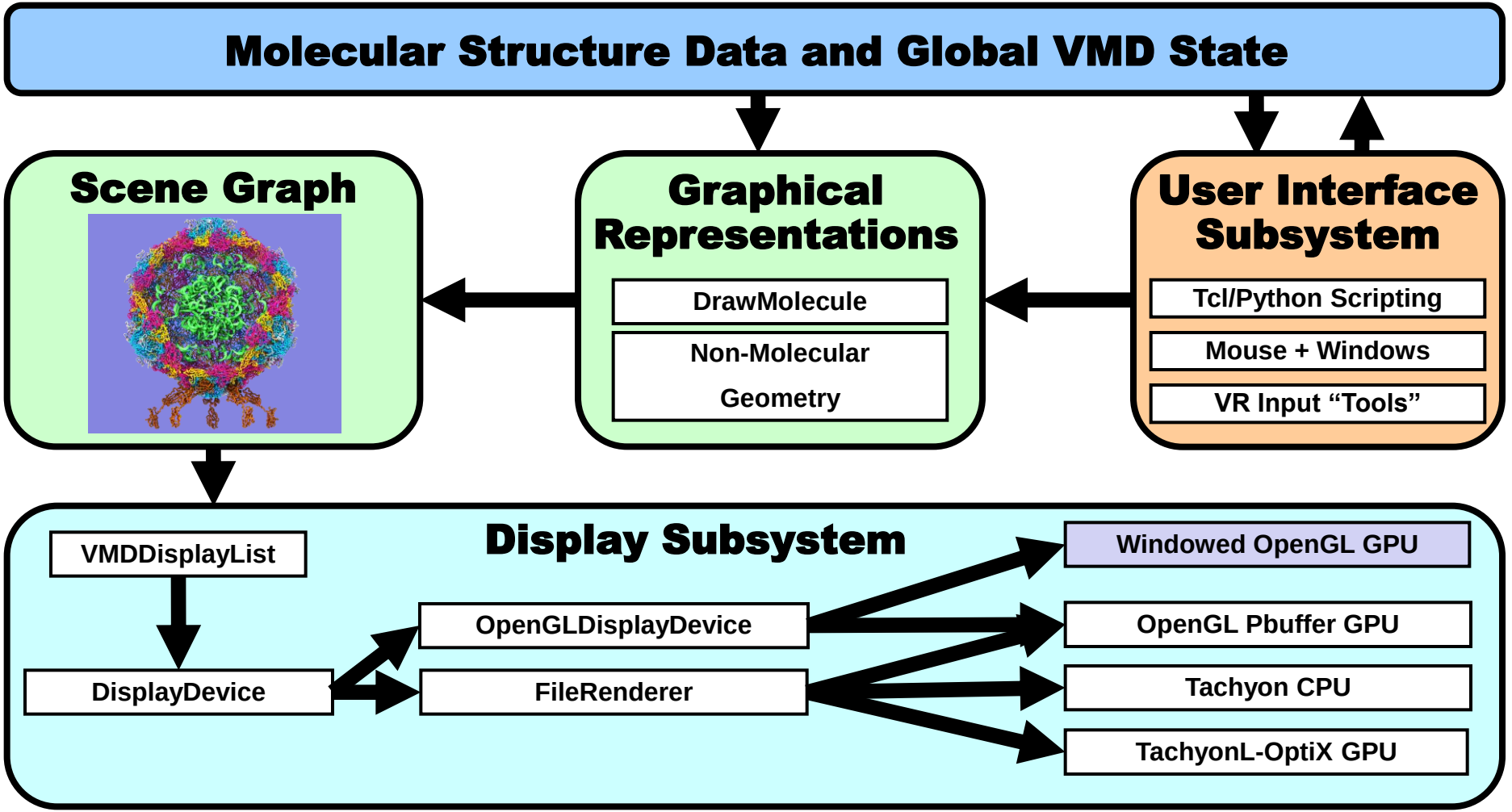
- GPU Ray Tracing:
  - Entire scene resident in GPU on-board memory for speed
  - RT performance is **heavily dependent on BVH** acceleration, particularly for scenes with large secondary ray workloads – shadow rays, ambient occlusion shadow feelers, transmission rays
  - RT **BVH structure regenerated / updated each trajectory timestep**, for some petascale visualizations BVH gen. can take up to ~25 sec!
- OpenGL GLSL:
  - No significant per-frame preprocessing required
  - Minimal persistent GPU memory footprint
  - Implements point sprites, ray cast spheres, pixel-rate lighting, …

TachyonL-Optix GPU Ray Tracing w/ OptiX+CUDA

- OptiX/CUDA kernels can only run for about 2 seconds uninterrupted
- GPU RT therefore cannot go wild with uninterrupted recursion, internal looping within shading code, or **GPU timeout will occur and kernel will be terminated** by OS/driver
- Complex ray tracing algorithms broken out into **multi-pass algorithms**:
  - Many GPU kernel launches (up to hundreds in some cases)
  - Intermediate rendering state written to GPU memory at end of each pass
  - Intermediate rendering state is reloaded at the start of the next pass
  - **Examples: state of multiple random number generators, color accumulation buffers, are stored and reloaded in our current implementation**

Why Built-In VMD Ray Tracing Engines?

- **No disk I/O** or communication to outboard renderers
- **Eliminate unnecessary data replication and host-GPU memory transfers**
- Directly operate on VMD internal molecular scene, **quantized/compressed data formats**
- Implement all **curved surface primitives**, volume rendering, texturing, shading features required by VMD
- **Same scripting, analysis, atom selection**, and rendering features are available on all platforms, **graceful CPU fallback**

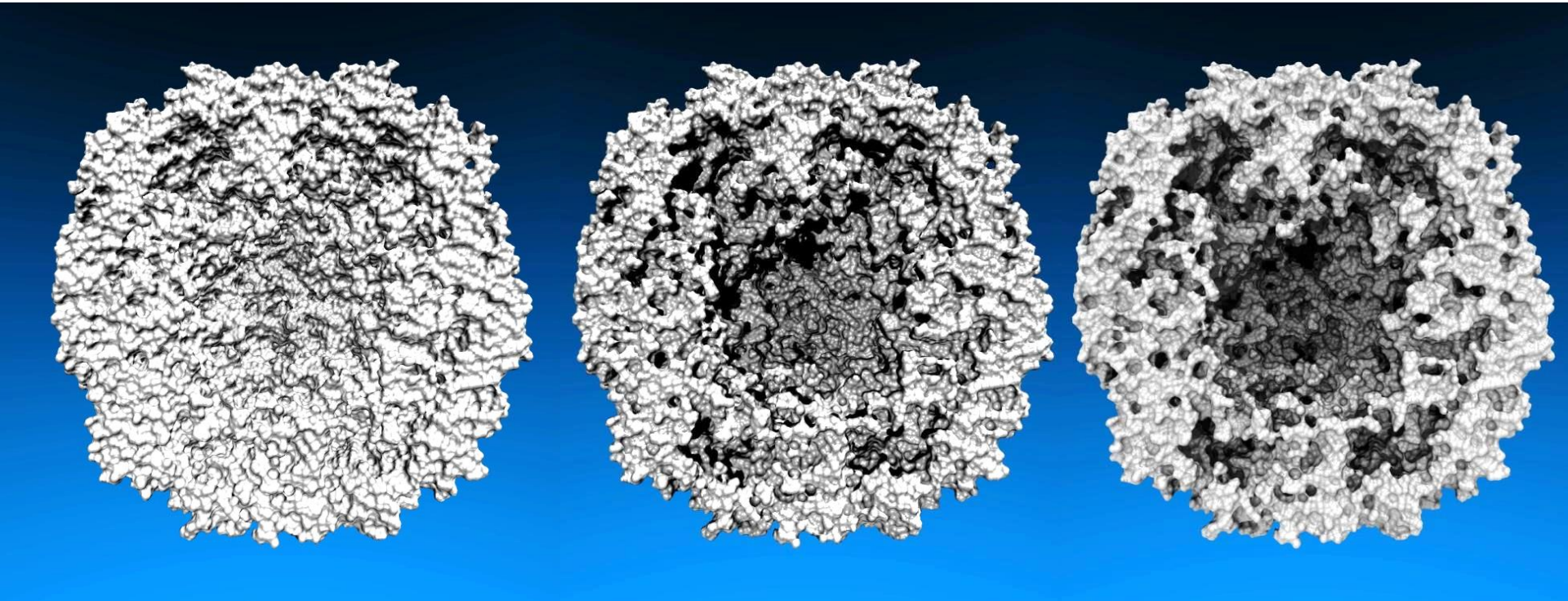

Molecular Structure Data and Global VMD State
Scene Graph
Graphical Representations
DrawMolecule
Non-Molecular Geometry
User Interface Subsystem
Tcl/Python Scripting
Mouse + Windows
VR Input “Tools”
Display Subsystem
VMDDisplayList
DisplayDevice
OpenGLDisplayDevice
FileRenderer
Windowed OpenGL GPU
OpenGL Pbuffer GPU
Tachyon CPU
TachyonL-OptiX GPU

Lighting Comparison

Two lights, no shadows

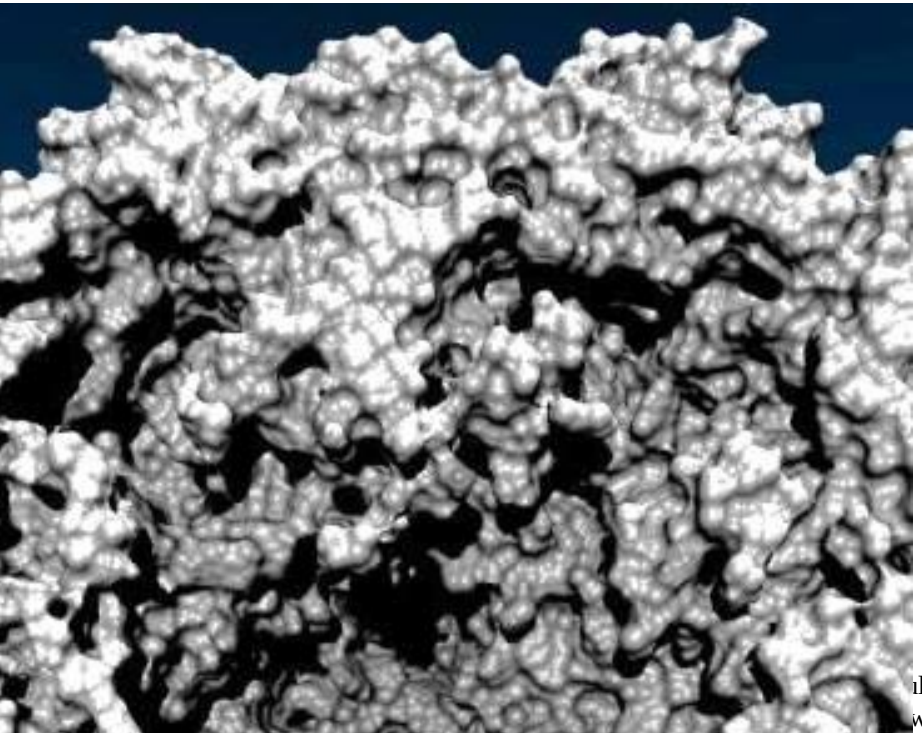
Two lights, hard shadows, 1 shadow ray per light

Ambient occlusion + two lights, 144 AO rays/hit

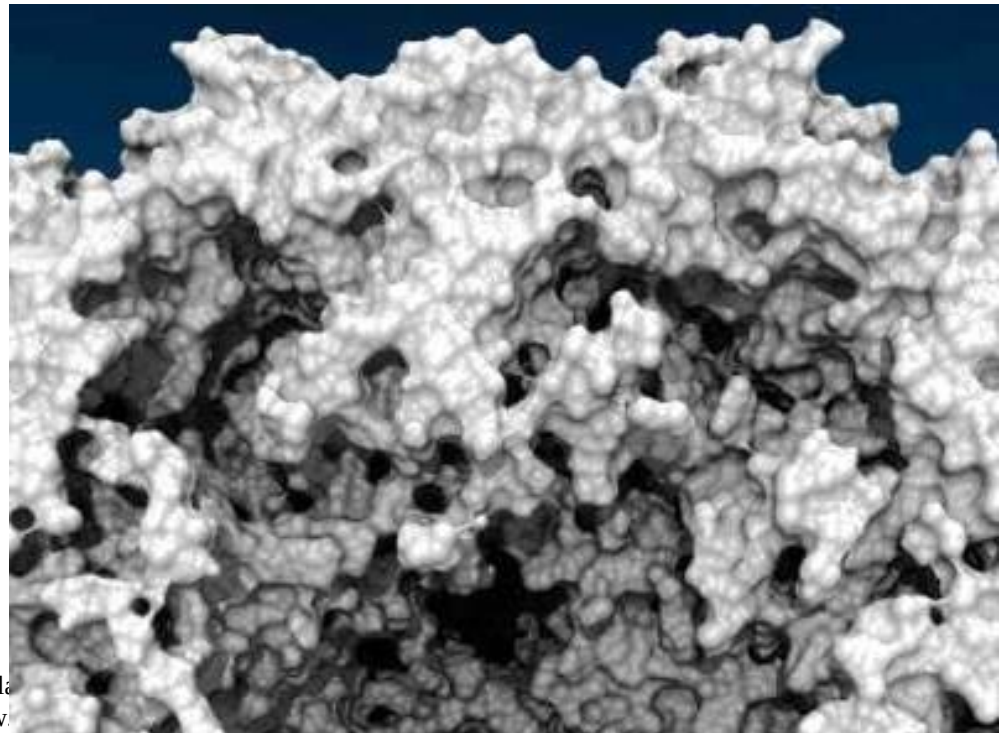


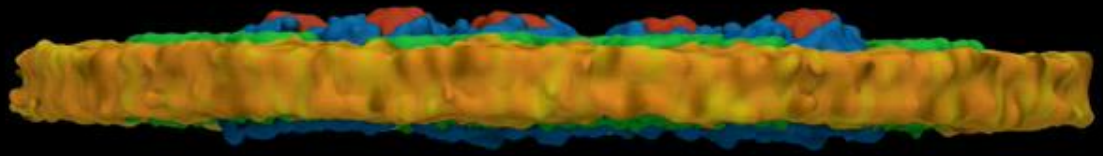
“My Lights are Always in the Wrong Place…”

**Two lights,
harsh shadows,
1 shadow ray per light per hit**

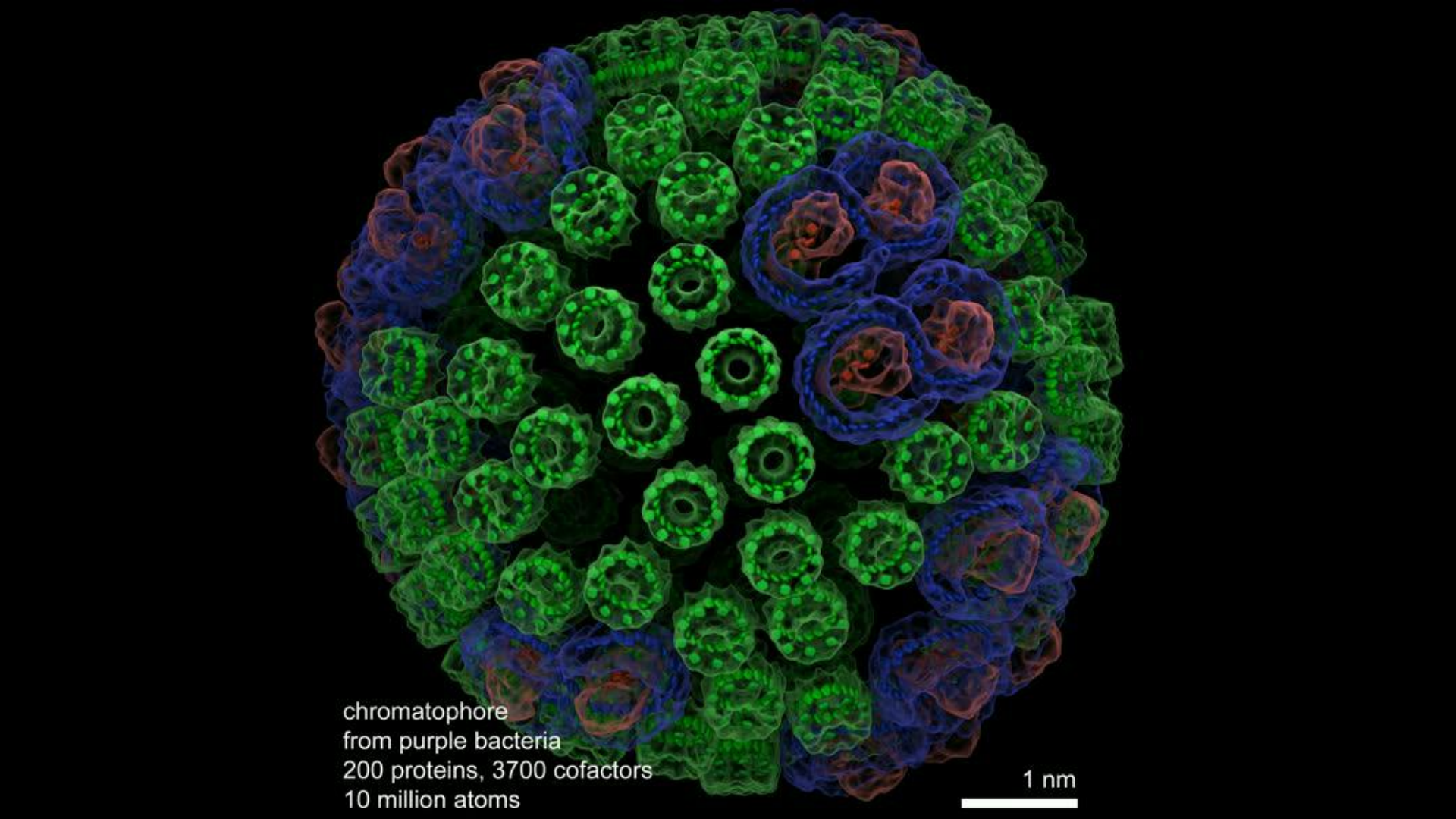


**Ambient occlusion (~80%)
+ two lights (~20%),
144 AO rays/hit**



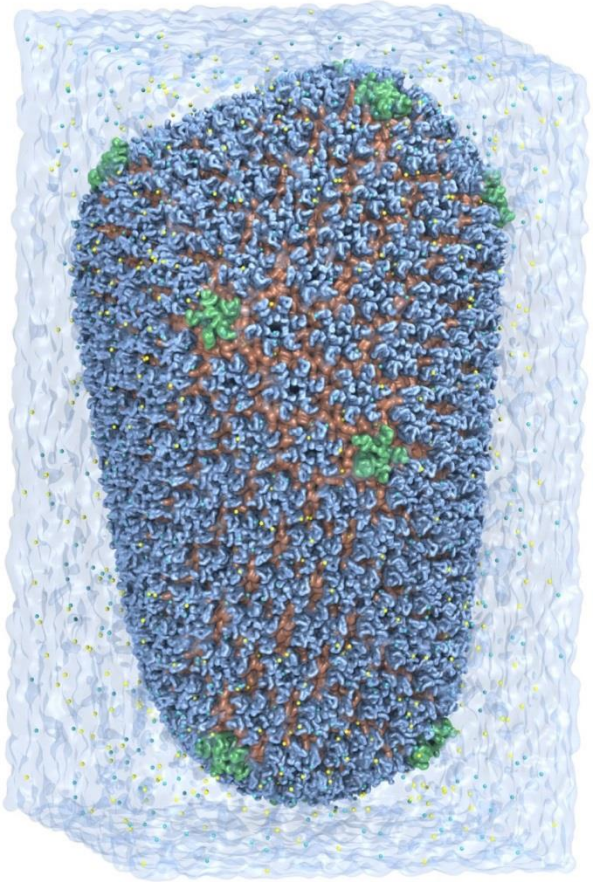


20 M atom chromatophore patch

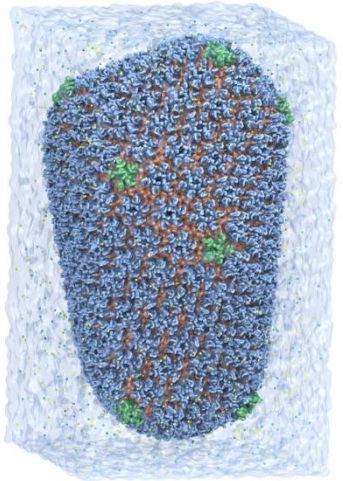

chromatophore
from purple bacteria
200 proteins, 3700 cofactors
10 million atoms
1 nm

GPU Ray Tracing of HIV-1 on Blue Waters

- 64M atom simulation, 1079 movie frames
- **Ambient occlusion lightin**g, shadows, transparency, antialiasing, depth cueing, **144 rays/pixel minimum**
- GPU memory capacity hurdles:
  - Surface calc. and ray tracing each use **over 75% of K20X 6GB on-board GPU memory** even with quantized/compressed colors, surface normals, …
  - Evict non-RT GPU data to host prior to ray tracing
  - Eviction was **still required** on a test machine with a **12GB Quadro K6000** GPU – the multi-pass surface algorithm grows the per-pass chunk size to reduce the number of passes

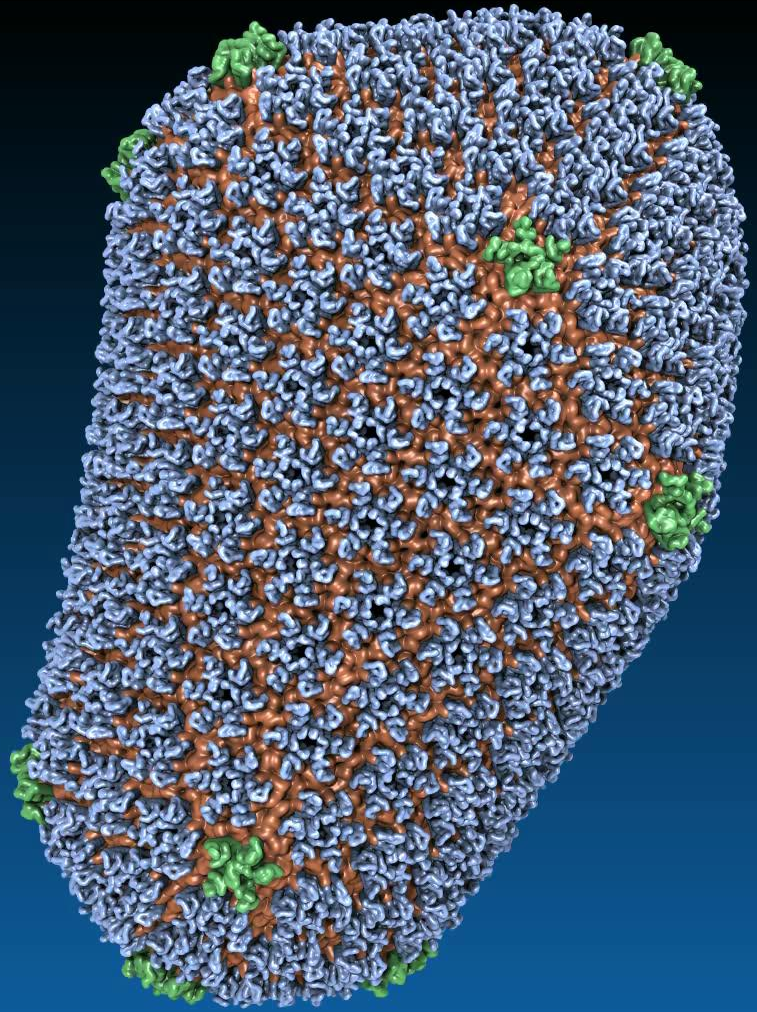


HIV-1 Parallel HD Movie Rendering on Blue Waters Cray XE6/XK7



New “TachyonL-OptiX” on XK7 vs. Tachyon on XE6:
K20X GPUs yield **up to eight times** geom+ray tracing speedup

Node Type and Count	Script Load Time	State Load Time	Geometry + Ray Tracing	Total Time
256 XE6 CPUs	7 s	160 s	**1,374 s**	**1,541 s**
512 XE6 CPUs	13 s	211 s	808 s	1,032 s
64 XK7 Tesla K20X GPUs	2 s	38 s	655 s	695 s
128 XK7 Tesla K20X GPUs	4 s	74 s	331 s	410 s
256 XK7 Tesla K20X GPUs	**7 s**	**110 s**	**171 s**	**288 s**



Future Work

- Improve multi-pass ray casting implementation
- Improve GPU BVH regen speed for time-varying geometry, MD trajectories
- Performance improvements for ambient occlusion sampling strategy, optional use of distance-limited shadow feelers per Laine et al.
- Continue tuning of GPU-specific RT intersection routines, memory layout
- Add GPU-accelerated movie encoder back-end

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
- NVIDIA OptiX team – especially James Bigler
- NVIDIA CUDA team
- Funding:
  - NSF OCI 07-25070
  - NSF PRAC “The Computational Microscope”
  - NIH support: 9P41GM104601, 5R01GM098243-02


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

GPU Computing Publications

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

- **Early Experiences Scaling VMD Molecular Visualization and Analysis Jobs on Blue Waters.** J. E. Stone, B. Isralewitz, and K. Schulten. In proceedings, Extreme Scaling Workshop, 2013. (In press)
- **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.