# GPU-Accelerated Visualization and Analysis of Petascale Molecular Dynamics Simulations

John E. 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/vmd/**

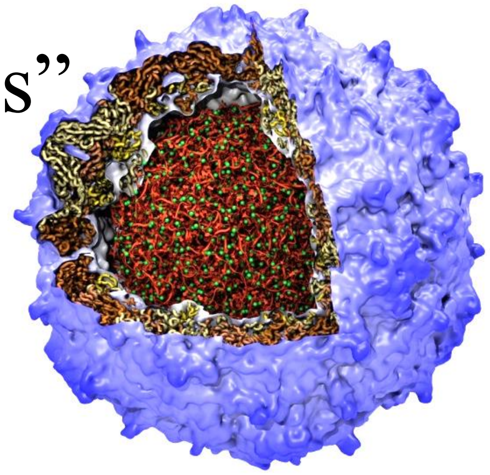
**http://www.ks.uiuc.edu/Research/gpu/**

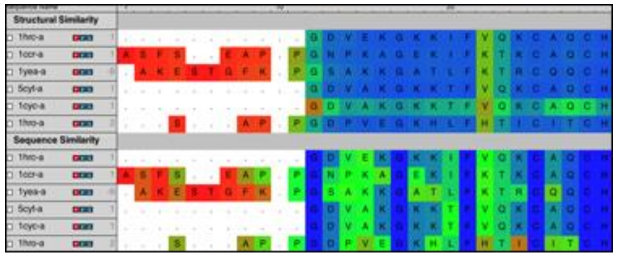
Missouri S&T, Research and Technology Day,

September 16, 2014

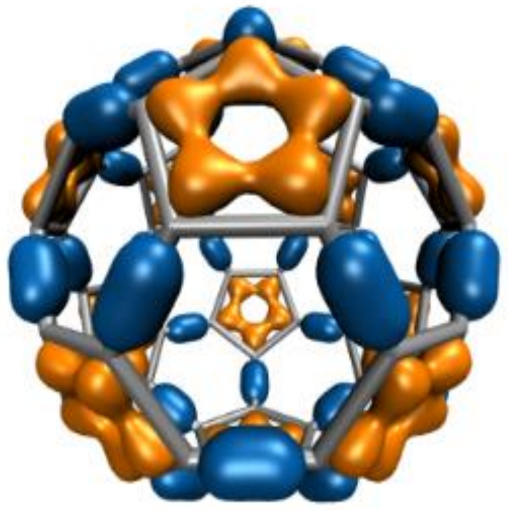
## VMD – “Visual Molecular Dynamics”

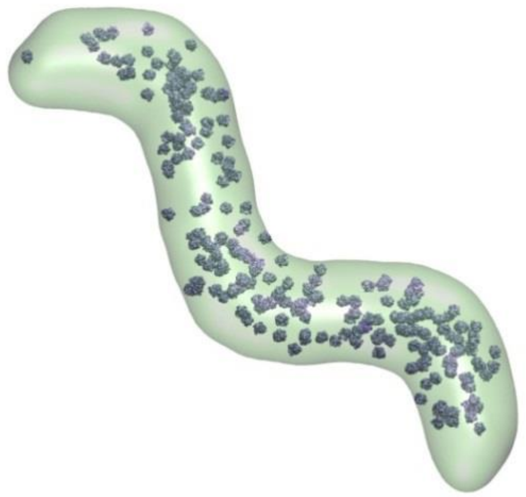
- Visualization and analysis of:
  - molecular dynamics simulations
  - quantum chemistry calculations
  - particle systems and whole cells
  - sequence data
- User extensible w/ scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/

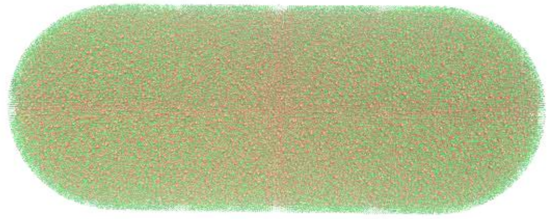

Poliovirus



Ribosome Sequences


Electrons in
Vibrating Buckyball

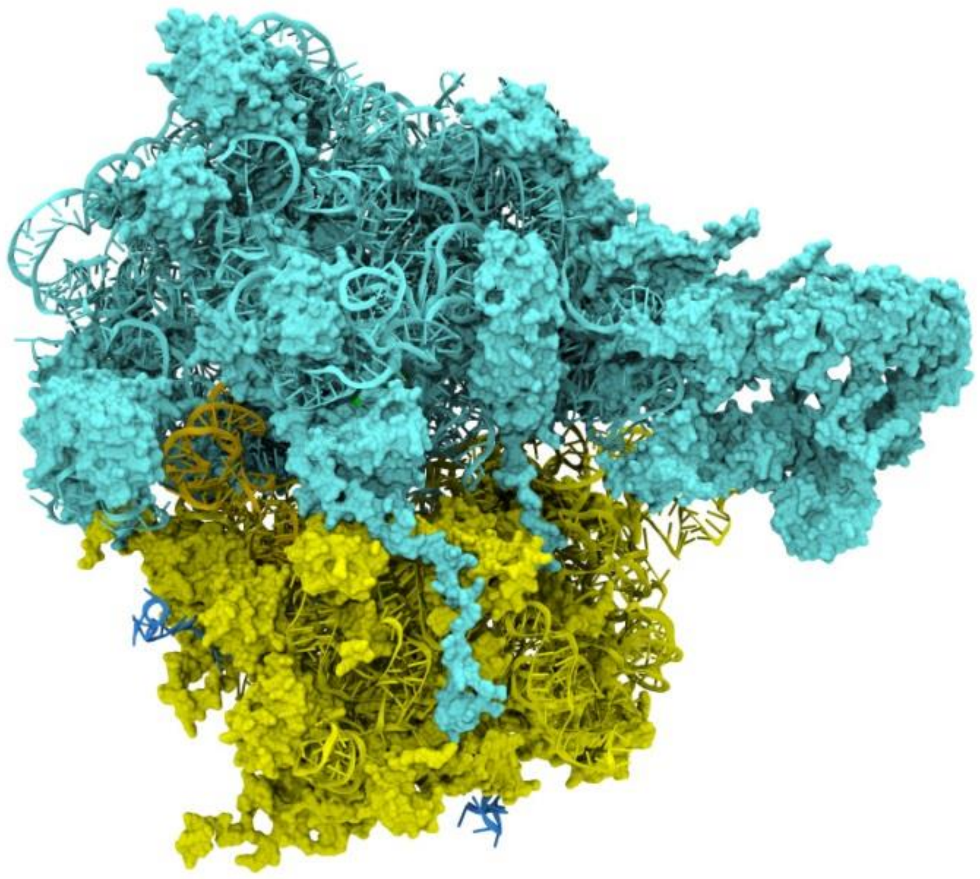

Cellular Tomography
Cryo-electron Microscopy


Whole Cell Simulations

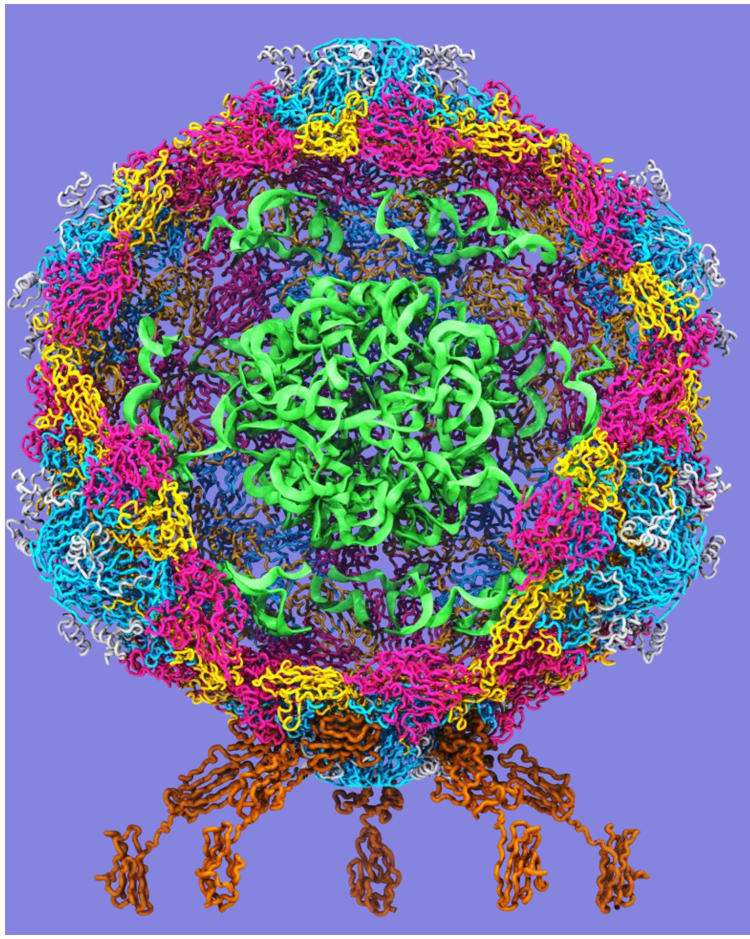
# Goal: A Computational Microscope

Study the molecular machines in living cells

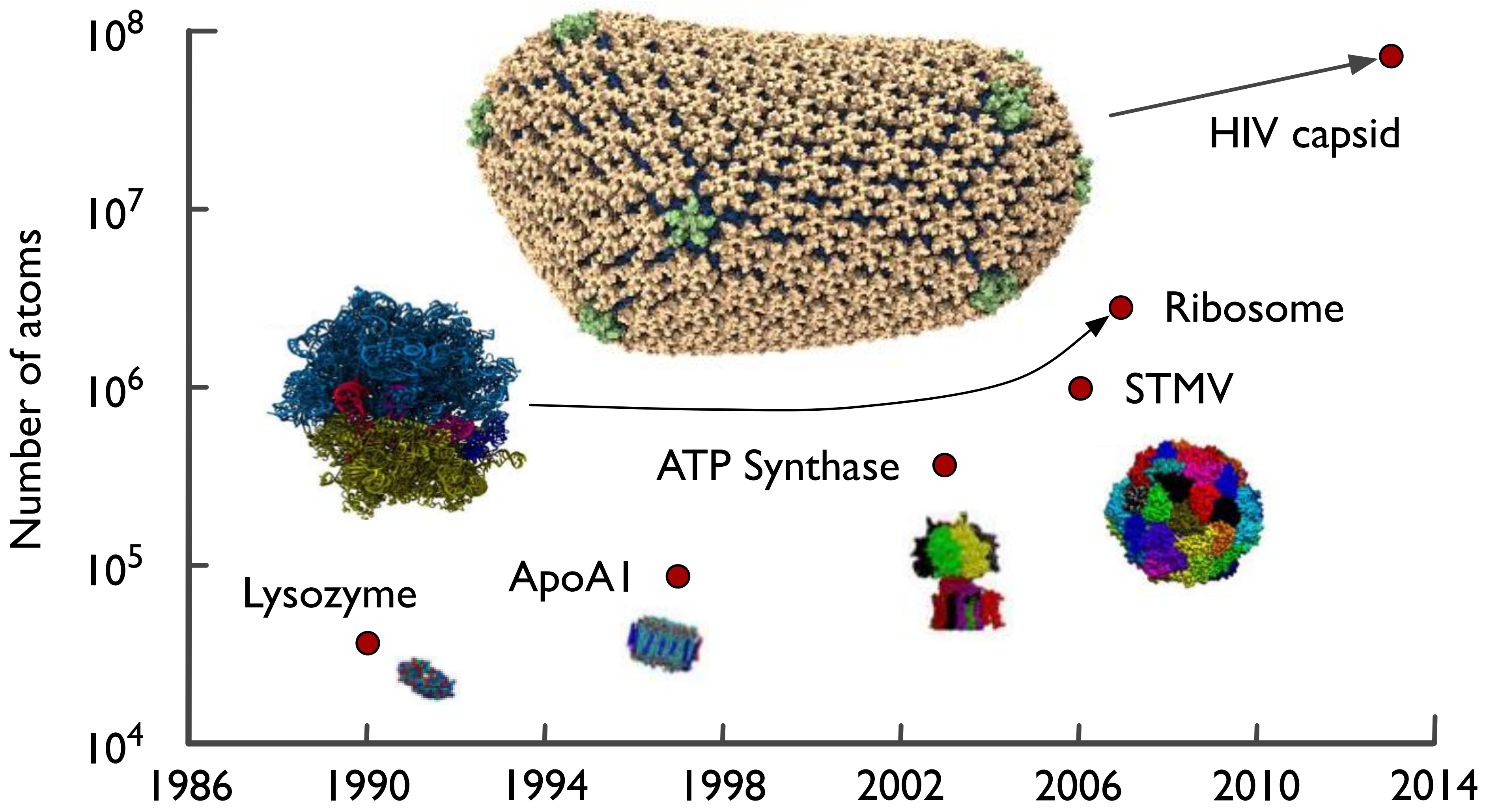
Ribosome: target for antibiotics



Poliovirus

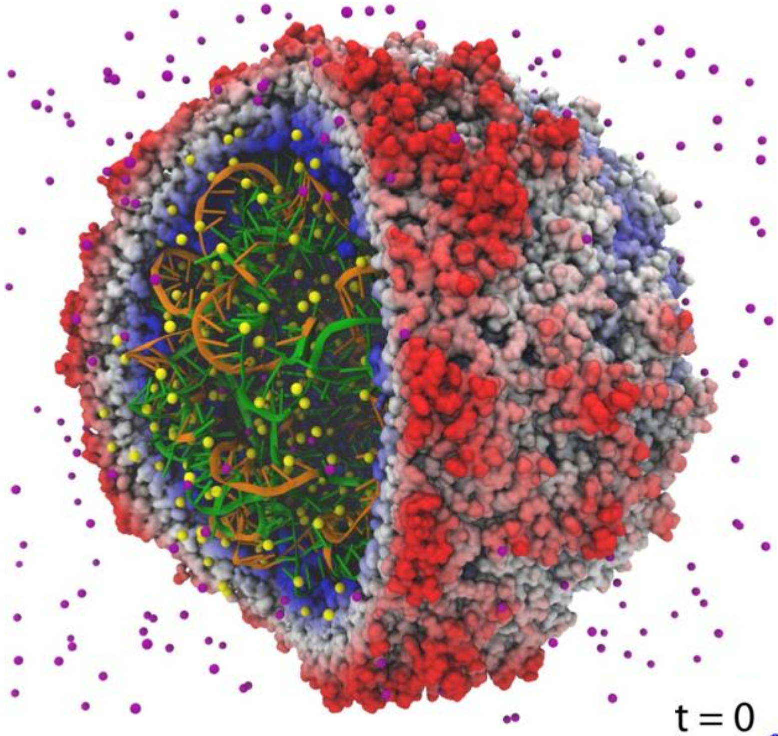


# NAMD and VMD Use GPUs & Petascale Computing to Meet Computational Biology's Insatiable Demand for Processing Power



# First Simulation of a Virus Capsid (2006)

Satellite Tobacco Mosaic Virus (STMV)

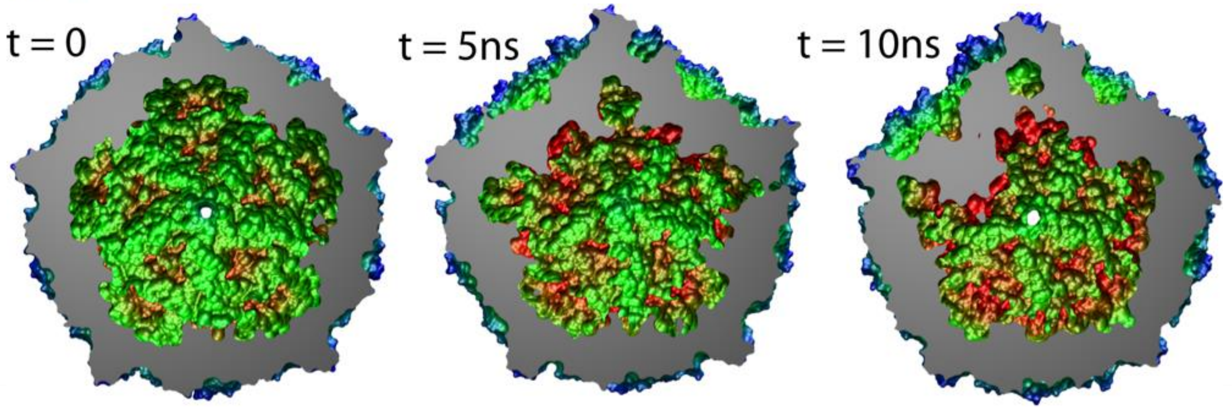


First MD simulation of a complete virus capsid

STMV smallest available capsid structure

**STMV simulation, visualization, and analysis pushed us toward GPU computing!**

MD showed that STMV capsid collapses without its RNA core



**1 million atoms**
**A huge system for 2006**

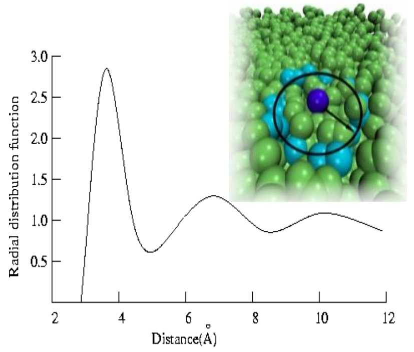
Freddolino et al., ***Structure***, **14**:437 (2006)

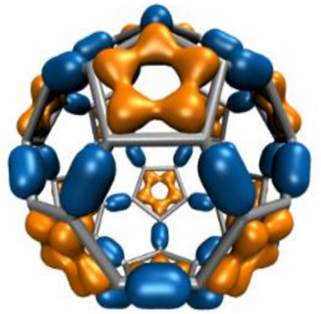
# GPU Computing

- Commodity devices, omnipresent in modern computers (over a **million** sold per **week**)
- Massively parallel hardware, hundreds of processing units, **throughput oriented architecture**
- Standard integer and floating point types supported
- Programming tools allow software to be written in dialects of familiar C/C++ and integrated into legacy software
- GPU algorithms are often multicore friendly due to attention paid to **data locality** and **data-parallel** work decomposition

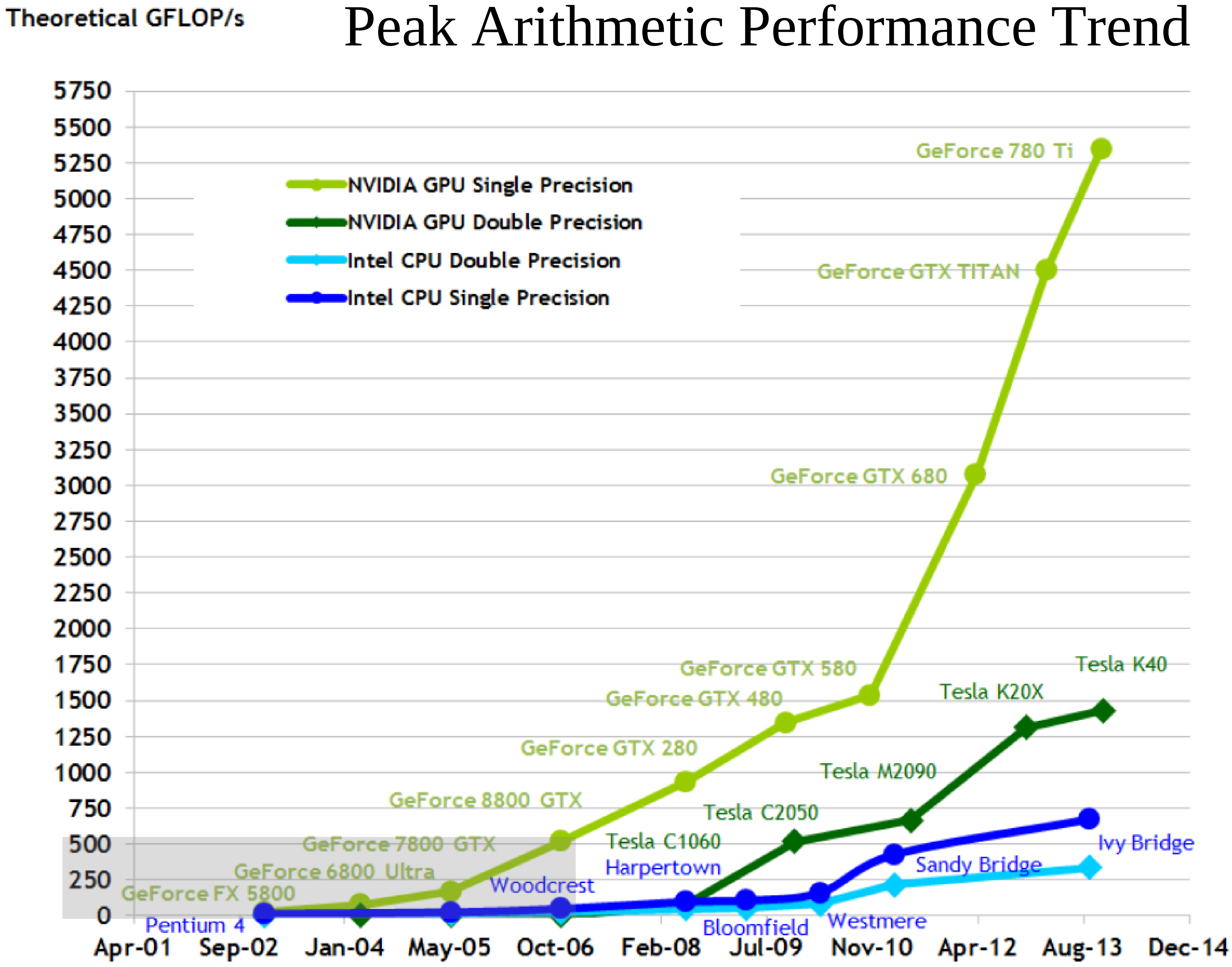
# CUDA GPU-Accelerated Trajectory Analysis and Visualization in VMD

| **VMD GPU-Accelerated Feature or Kernel** | **Typical speedup vs. multi-core CPU (e.g. 4-core CPU)** |
|---|---|
| **Molecular orbital display** | **30x** |
| **Radial distribution function** | **23x** |
| **Molecular surface display** | **15x** |
| **Electrostatic field calculation** | **11x** |
| **Ray tracing w/ shadows, AO lighting** | **7x** |
| Ion placement | 6x |
| MDFF density map synthesis | 6x |
| Implicit ligand sampling | 6x |
| Root mean squared fluctuation | 6x |
| Radius of gyration | 5x |
| Close contact determination | 5x |
| Dipole moment calculation | 4x |

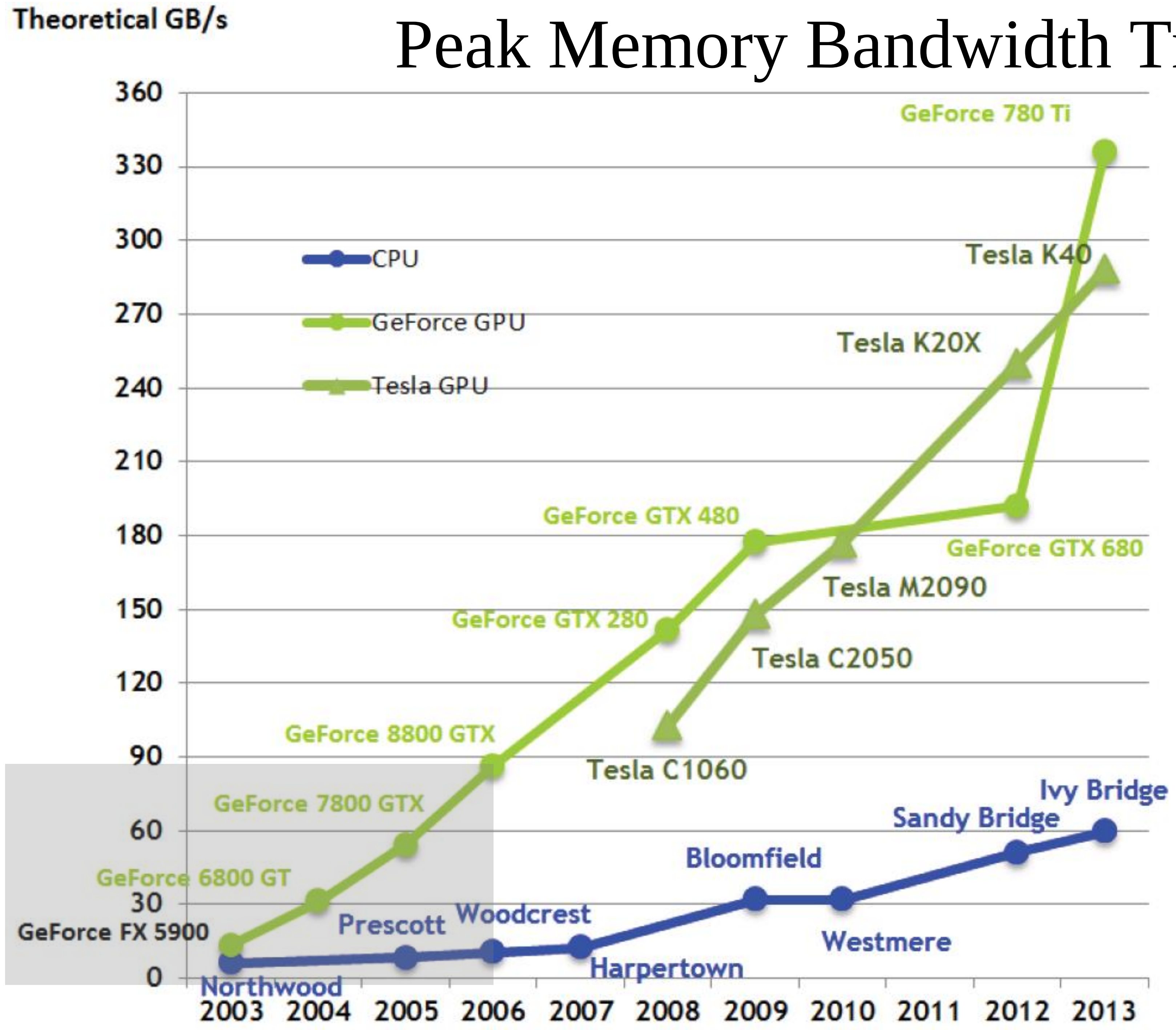




# Peak Arithmetic Performance Trend



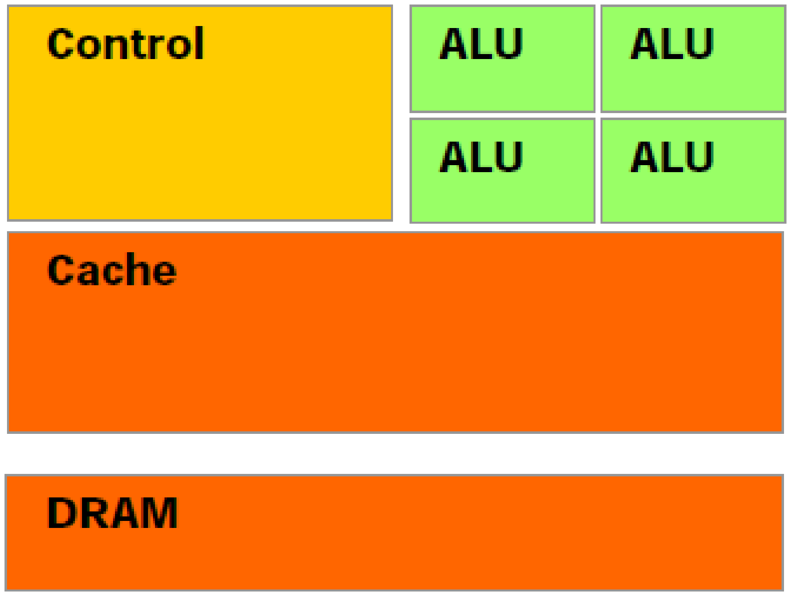
# Peak Memory Bandwidth Trend

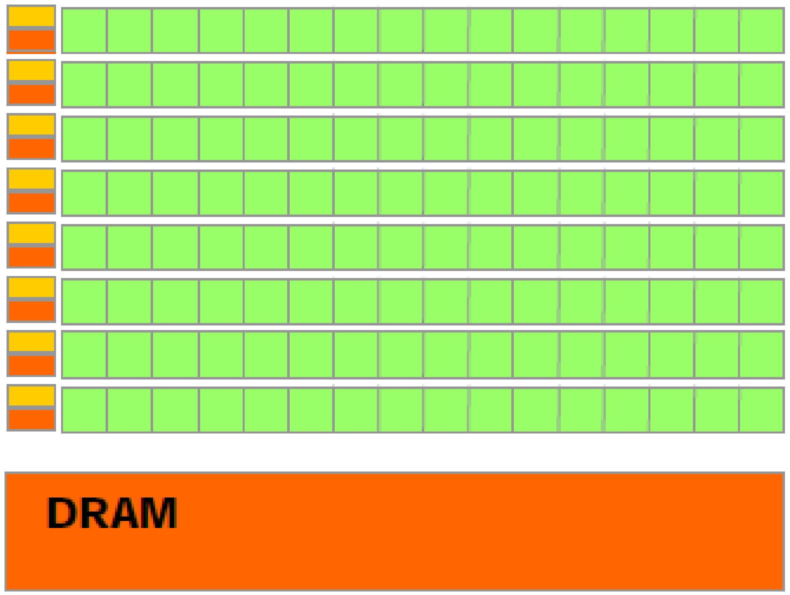


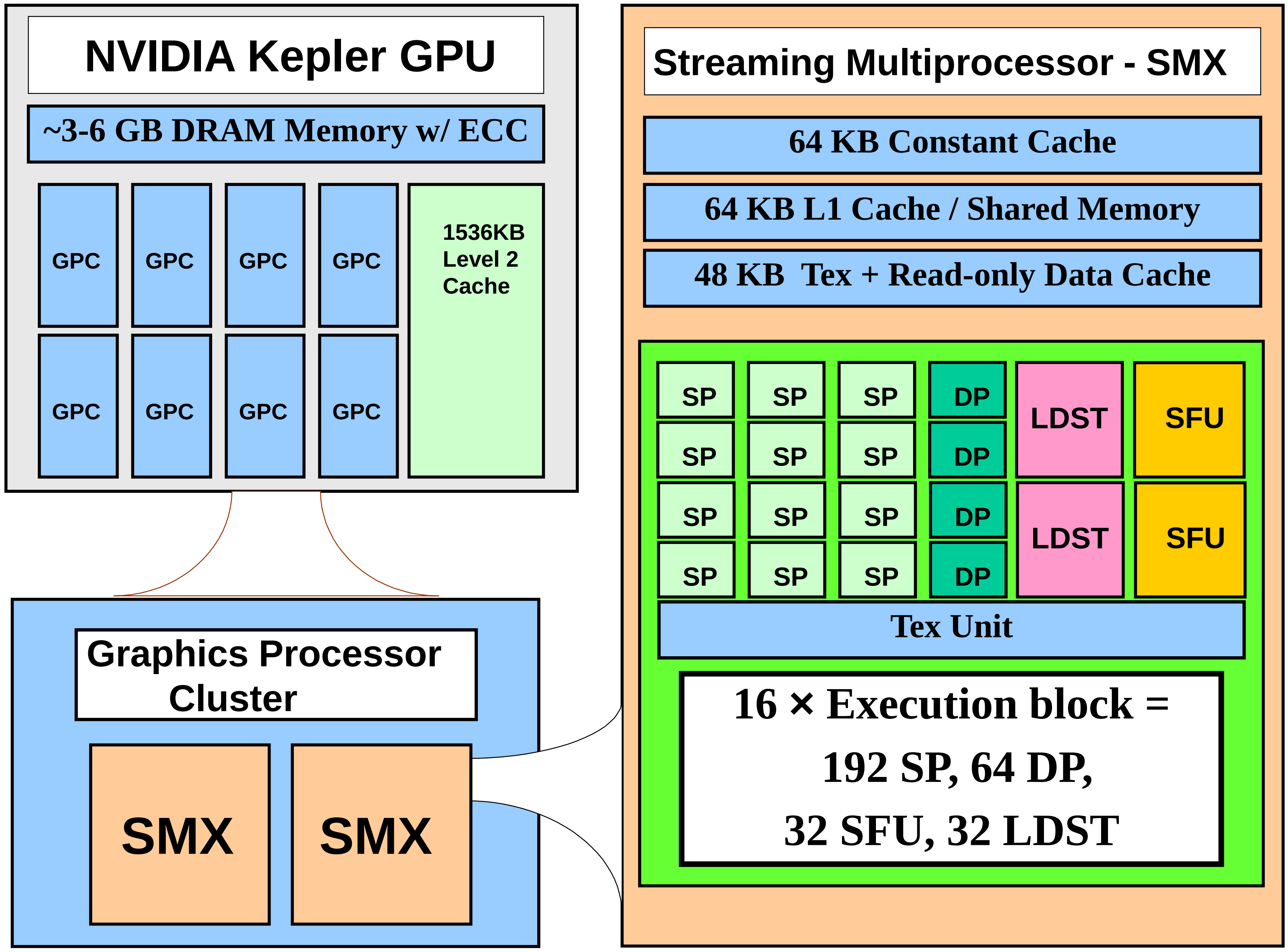
# Comparison of CPU and GPU Hardware Architecture

**CPU**: Cache heavy, focused on individual thread performance

**GPU**: ALU heavy, massively parallel, throughput oriented



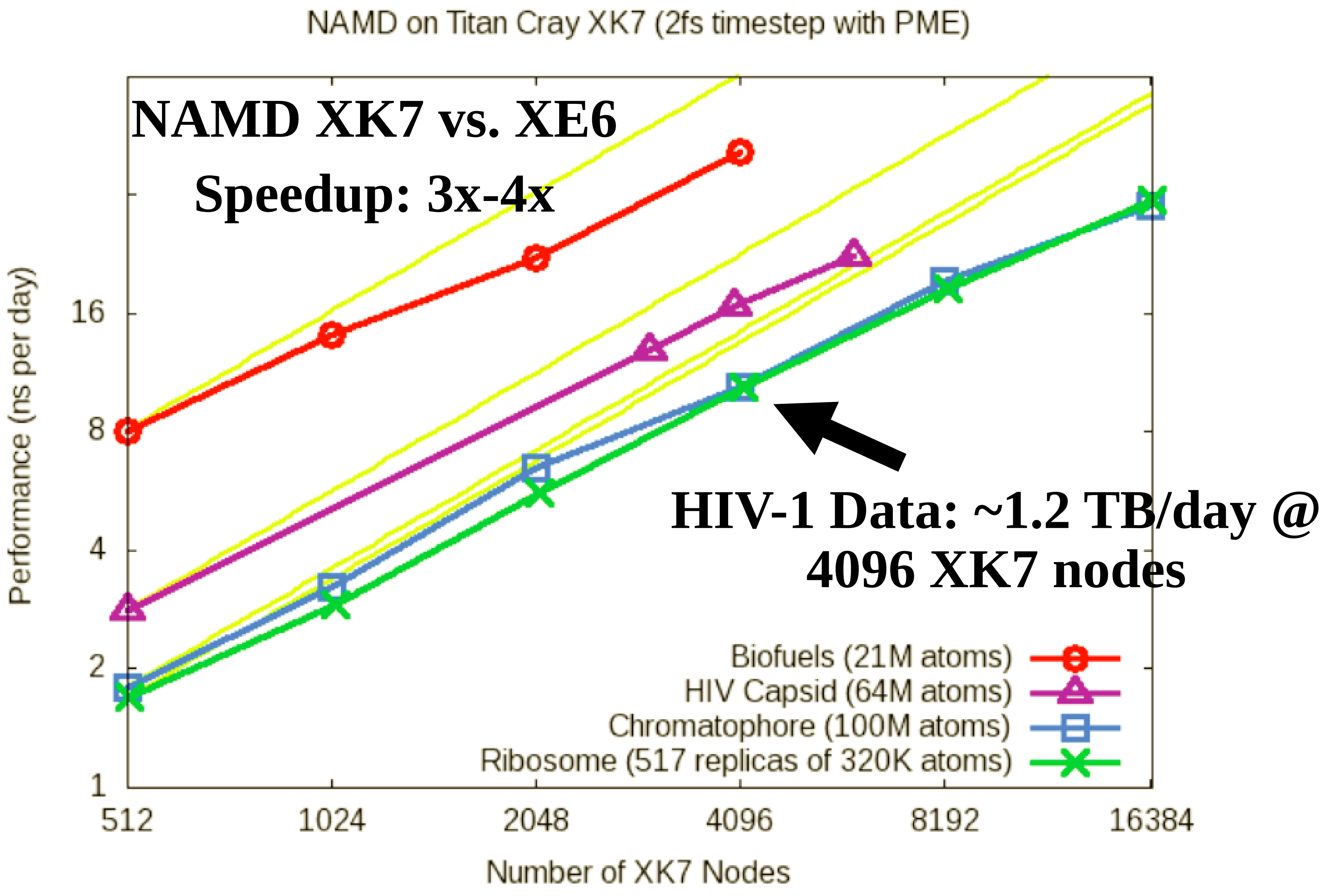



NVIDIA Kepler GPU
~3-6 GB DRAM Memory w/ ECC
GPC
GPC
GPC
GPC
GPC
GPC
GPC
GPC
1536KB Level 2 Cache
Graphics Processor Cluster
SMX
SMX
Streaming Multiprocessor - SMX
64 KB Constant Cache
64 KB L1 Cache / Shared Memory
48 KB Tex + Read-only Data Cache
SP
DP
LDST
SFU
Tex Unit
16 × Execution block = 192 SP, 64 DP, 32 SFU, 32 LDST

# GPU On-Board Global Memory

- GPU arithmetic rates dwarf memory bandwidth
- For Kepler K40 hardware:
  - ~4.3 SP TFLOPS vs. ~288 GB/sec
  - The ratio is roughly **60 FLOPS per memory reference** for single-precision floating point
- GPUs include multiple fast on-chip memories to help **narrow the gap**:
  - **Registers**
  - Constant memory (64KB)
  - **Shared memory (48KB / 16KB)**
  - Read-only data cache / Texture cache (48KB)

# NAMD Titan XK7 Performance August 2013



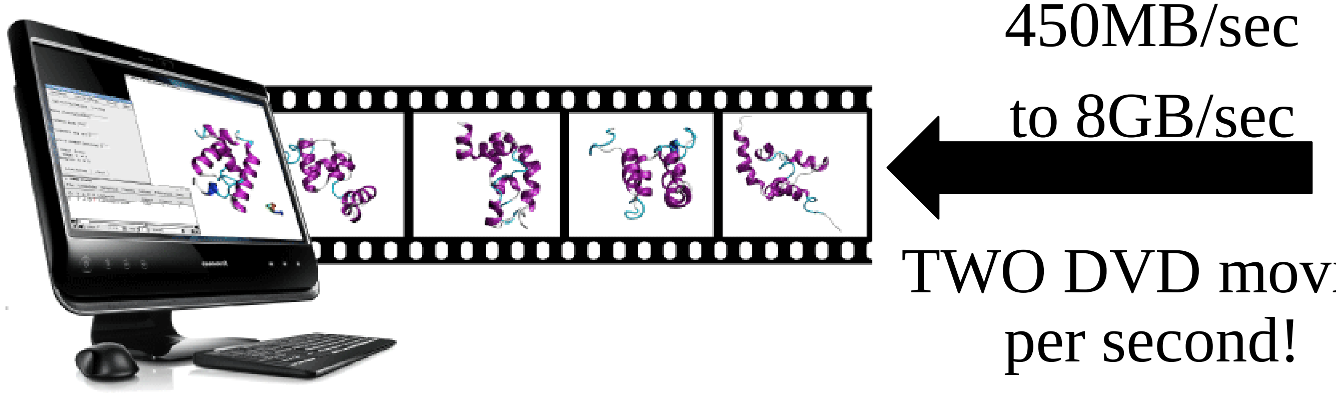
**NAMD XK7 vs. XE6**

**Speedup: 3x-4x**

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

# Interactive Display & Analysis of Terabytes of Data:

## Out-of-Core Trajectory I/O w/ Solid State Disks **and GPUs**



TWO DVD movies per second!

Commodity SSD, SSD RAID

- Timesteps loaded on-the-fly (out-of-core)
  - Eliminates memory capacity limitations, even for multi-terabyte trajectory files
  - High performance achieved by new trajectory file formats, optimized data structures, and efficient I/O
- **GPUs accelerate per-timestep calculations**
- Analyze long trajectories significantly faster using just a personal computer

**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.

# 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 ray tracing, 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

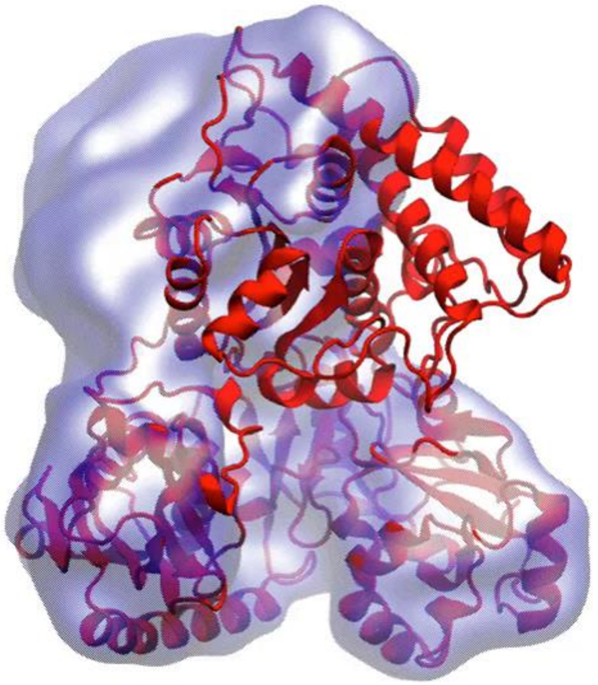
4,224 XK7 nodes w/ GPUs support fast VMD OpenGL movie rendering and visualization

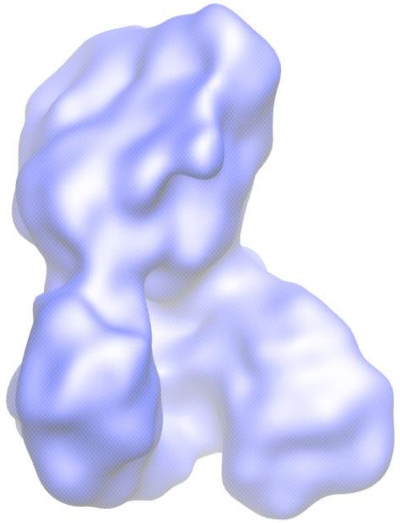
# Molecular Dynamics Flexible Fitting (MDFF)

$$\begin{tikzcd}
\text{X-ray crystallography} \arrow[r] & \text{MDFF} & \text{Electron microscopy} \arrow[l]
\end{tikzcd}$$

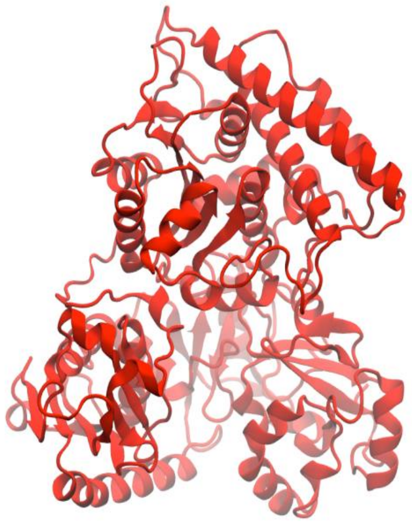
APS at Argonne

FEI microscope





ORNL Titan


Acetyl - CoA Synthase

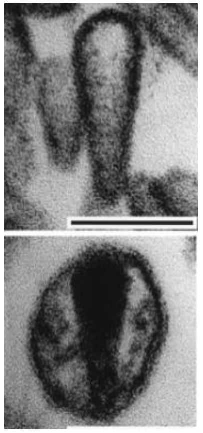
**Flexible fitting of atomic structures into electron microscopy maps using molecular dynamics.**
**L. Trabuco, E. Villa, K. Mitra, J. Frank, and K. Schulten. Structure, 16:673-683, 2008.**

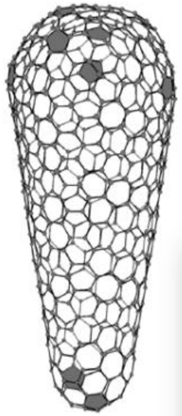
# Structural Route to the all-atom HIV-1 Capsid

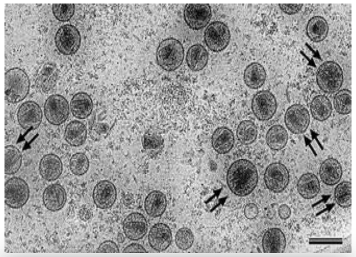
1st TEM (1999)

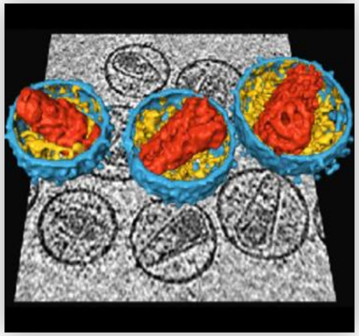
1st tomography (2003)

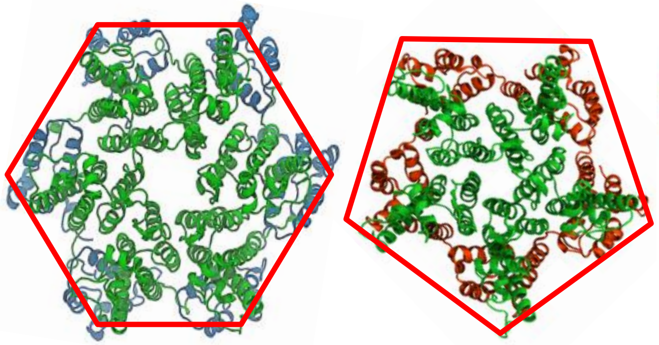
Crystal structures of separated hexamer and pentamer

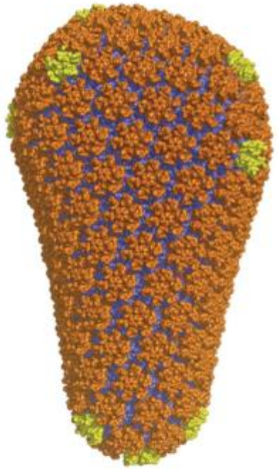












**Ganser et al. *Science*, 1999**

**Briggs et al. *EMBO J*, 2003**

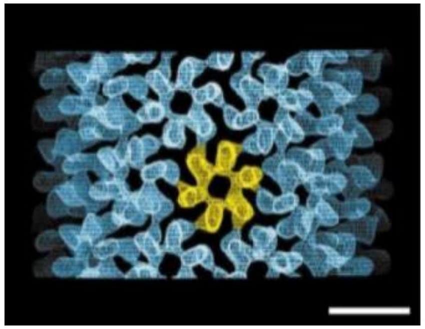
**Briggs et al. *Structure*, 2006**

cryo-ET (2006)

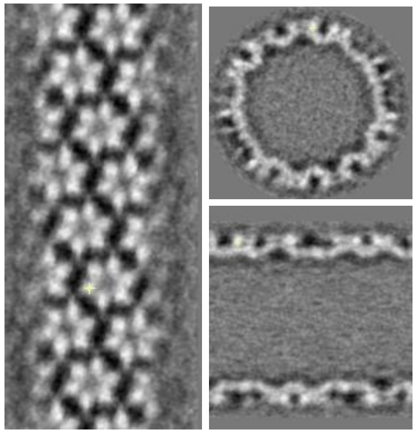
**Pornillos et al. , *Cell* 2009, *Nature* 2011**

High res. EM of hexameric tubule, tomography of capsid,
**all-atom model of capsid by MDFF w/ NAMD & VMD,**
**NSF/NCSA Blue Waters computer at Illinois**

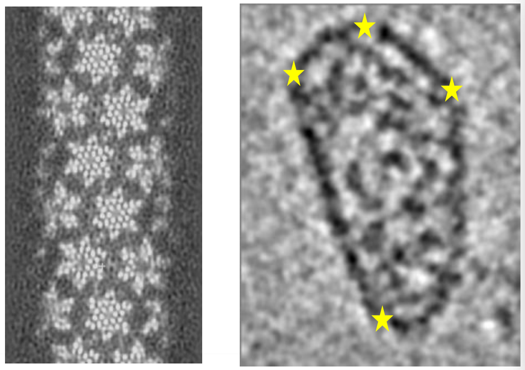
hexameric tubule



**Li et al., *Nature*, 2000**



**Byeon et al., *Cell* 2009**

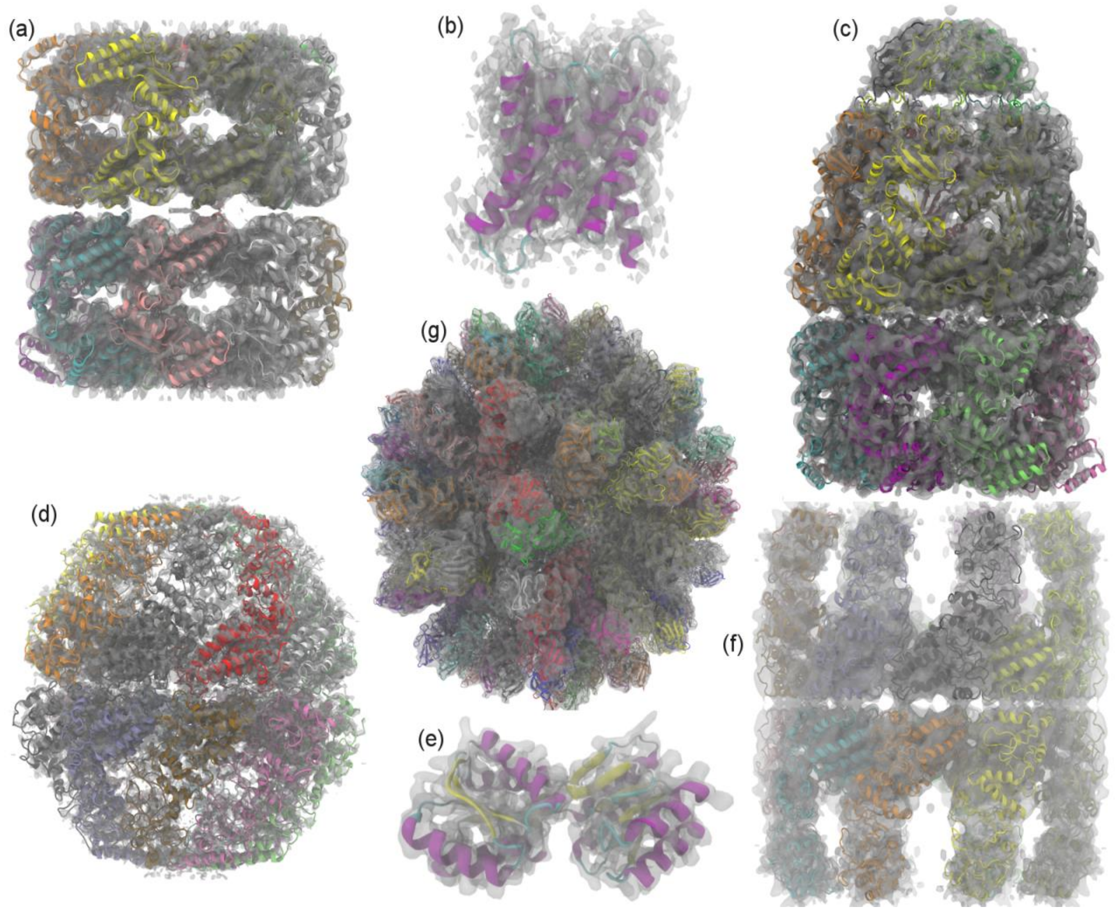


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
THE HIV-1 CAPSID
Atomic structure of the AIDS pathogen's protein coat

**Zhao et al. , *Nature* 497: 643-646 (2013)**

# Evaluating Quality-of-Fit for Structures Solved by Hybrid Fitting Methods

Compute Pearson correlation to evaluate the fit of a reference cryo-EM density map with a **simulated density map** produced from an **all-atom structure**.

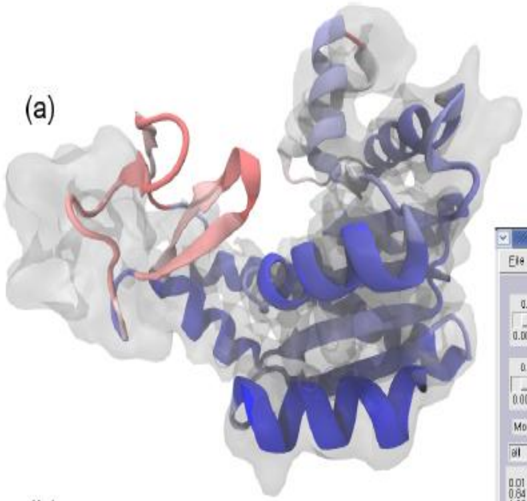


# GPUs Can Reduce Trajectory Analysis Runtimes from Hours to Minutes

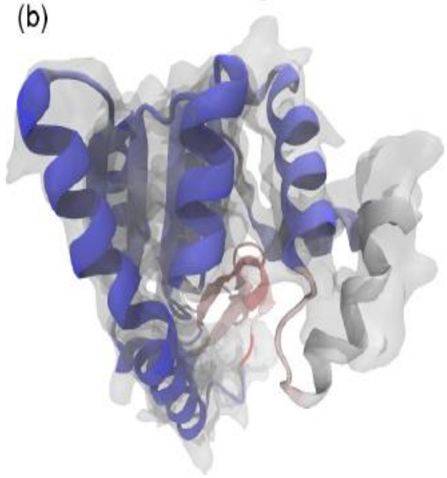
GPUs enable laptops and desktop workstations to handle tasks that would have previously required a cluster, or a very long wait…

GPU-accelerated petascale supercomputers enable analyses were previously impractical, allowing detailed study of very large structures such as viruses

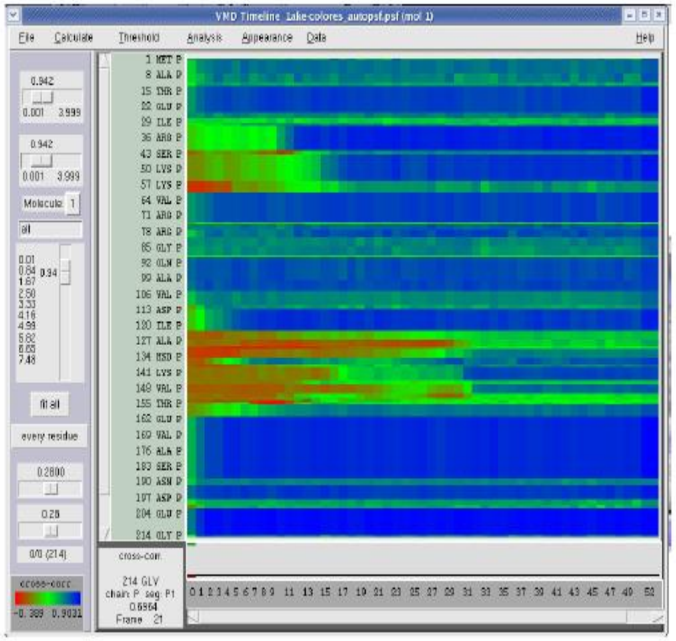
(a)



(b)



(c)



**GPU-accelerated MDFF Cross Correlation Timeline**

**Regions with poor fit** **Regions with good fit**

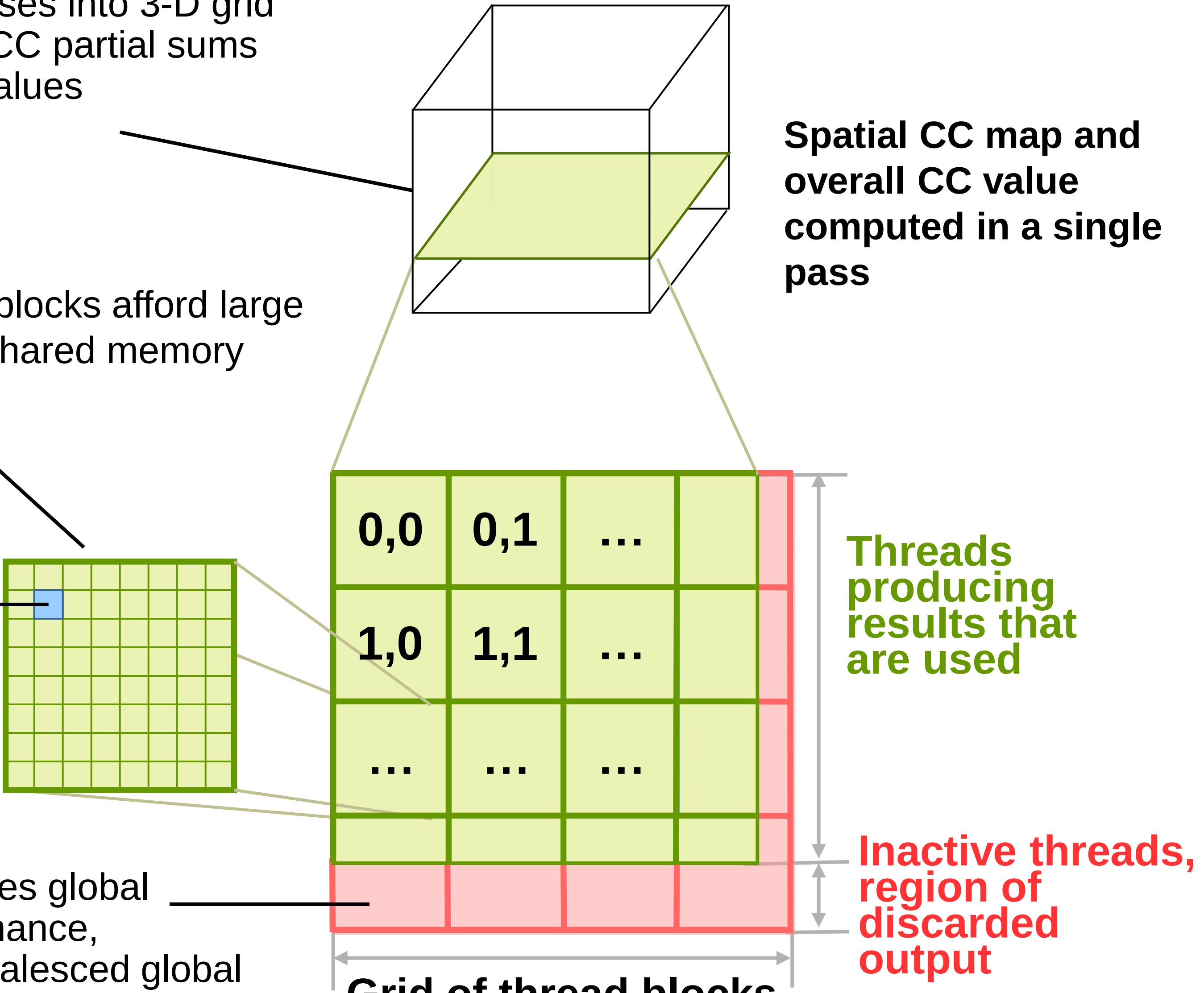
# Single-Pass MDFF GPU Cross-Correlation

3-D density map decomposes into 3-D grid of 8x8x8 tiles containing CC partial sums and local CC values

**Spatial CC map and overall CC value computed in a single pass**

Small 8x8x2 CUDA thread blocks afford large per-thread register count, shared memory

Each thread computes 4 z-axis density map lattice points and associated CC partial sums



0,0
0,1
…
1,0
1,1
…
…
…
…
Threads producing results that are used
Inactive threads, region of discarded output
Grid of thread blocks

Padding optimizes global memory performance, guaranteeing coalesced global memory accesses

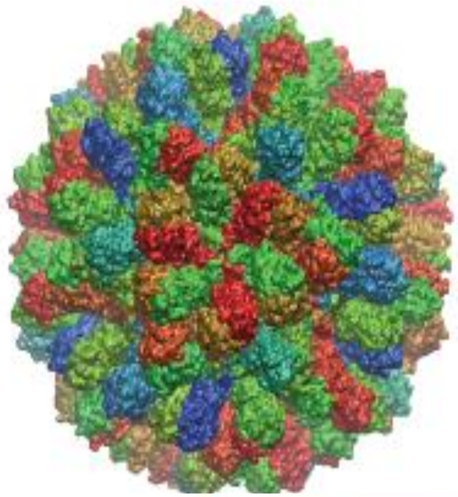
# VMD GPU Cross Correlation Performance

| | **RHDV** | **Mm-cpn open** | **GroEL** | **Aquaporin** |
|---|---|---|---|---|
| **Resolution (Å)** | **6.5** | **8** | **4** | **3** |
| **Atoms** | **702K** | **61K** | **54K** | **1.6K** |
| **VMD-CUDA**<br>**Quadro K6000** | **0.458s**<br>**34.6x** | **0.06s**<br>**25.7x** | **0.034s**<br>**36.8x** | **0.007s**<br>**55.7x** |
| VMD-CPU-SSE<br>32-threads, 2x Xeon E5-2687W | 0.779s<br>20.3x | 0.085s<br>18.1x | 0.159s<br>7.9x | 0.033s<br>11.8x |
| **Chimera**<br>**1-thread Xeon E5-2687W** | **15.86s**<br>**1.0x** | **1.54s**<br>**1.0x** | **1.25s**<br>**1.0x** | **0.39s**<br>**1.0x** |
| **VMD CPU-SEQ (plugin)**<br>1-thread Xeon E5-2687W | 62.89s<br>0.25x | 2.9s<br>0.53x | 1.57s<br>0.79x | 0.04s<br>9.7x |

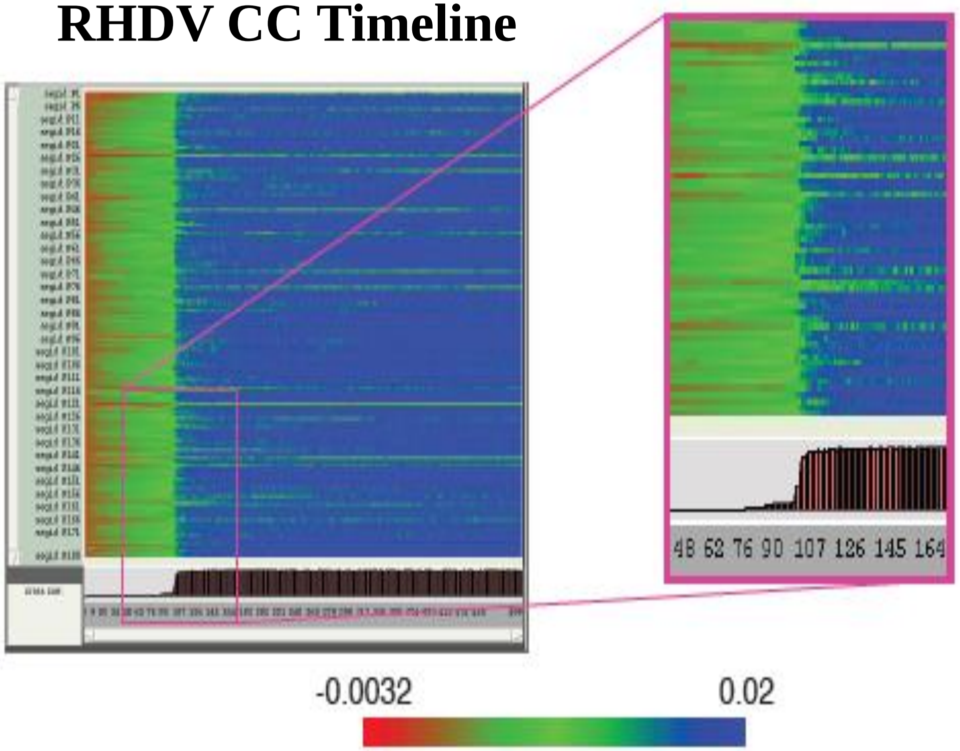
# VMD RHDV Cross Correlation Timeline on Cray XK7

| | **RHDV** |
|---|---|
| **Atoms** | **702K** |
| **Traj. Frames** | **10,000** |
| **Component Selections** | **720** |
| **Single-node XK7 (projected)** | **336 hours (14 days)** |
| **128-node XK7** | **3.2 hours<br>105x speedup** |
| **2048-node XK7** | **19.5 minutes<br>1035x speedup** |

Calculation would take **5 years** using original serial VMD CC plugin on a workstation!

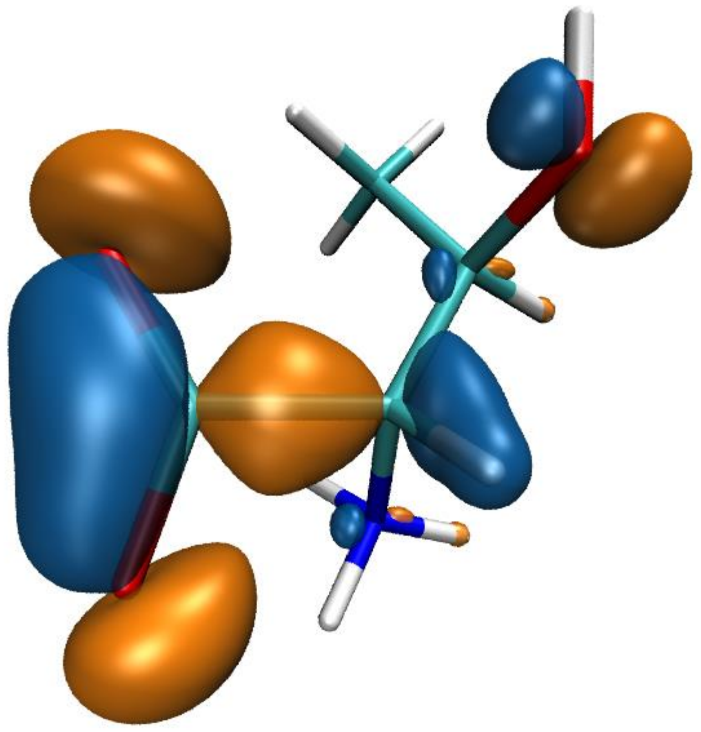


**RHDV CC Timeline**



# Animating Molecular Orbitals

- Animation of (classical mechanics) molecular dynamics trajectories provides insight into simulation results
- To do the same for QM or QM/MM simulations one must compute MOs at ~**10 FPS** or more
- **>100x** speedup (GPU) over existing tools now makes this possible!



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.

# 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

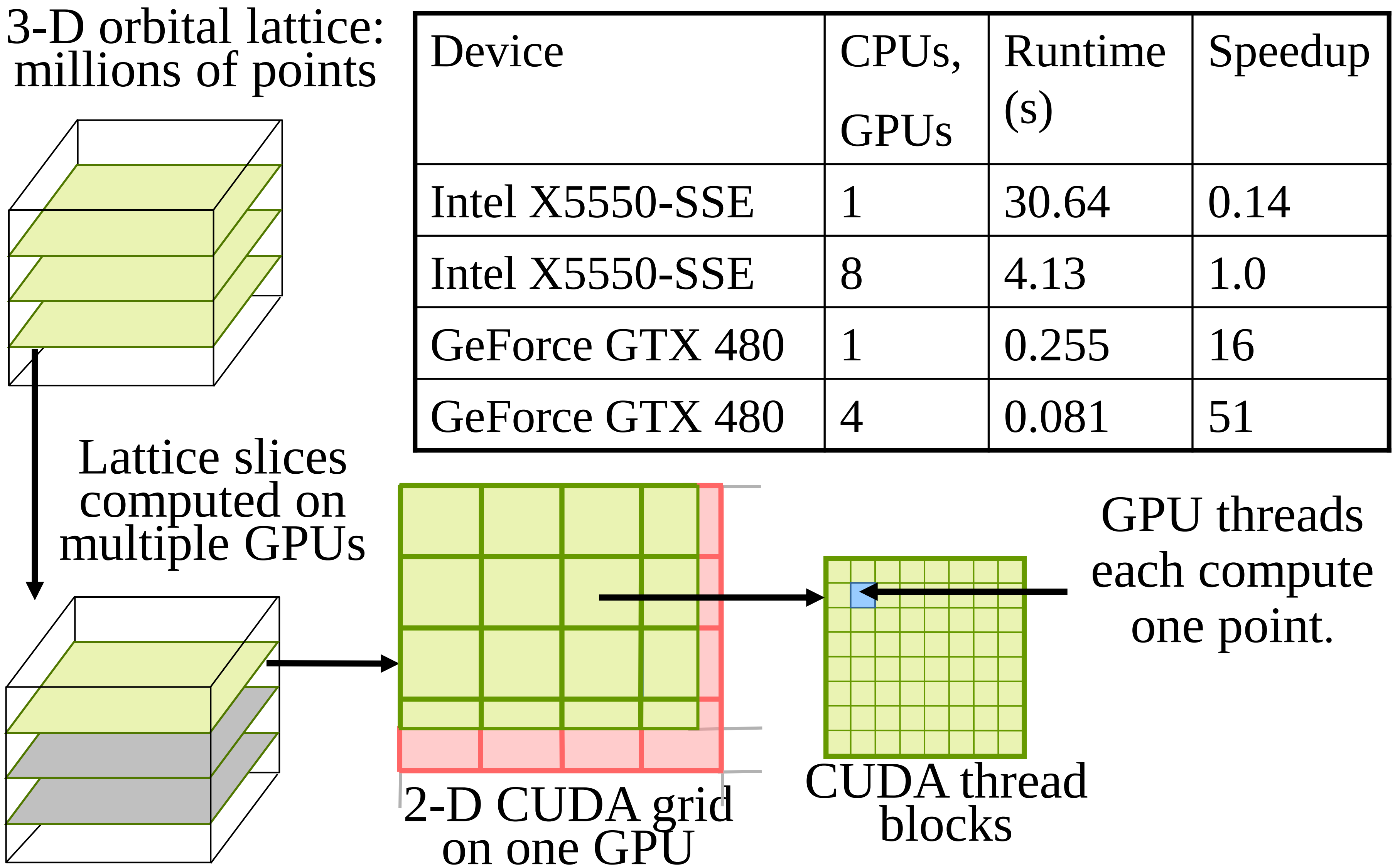
Loop over primitives: largest component of runtime, due to expf()

Loop over angular momenta

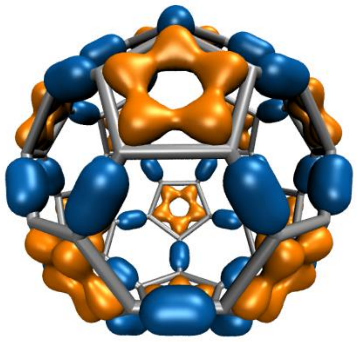
(unrolled in real code)

# GPU Solution: Computing $C_{60}$ Molecular Orbitals

| Device | CPUs, GPUs | Runtime (s) | Speedup |
|---|---|---|---|
| Intel X5550-SSE | 1 | 30.64 | 0.14 |
| Intel X5550-SSE | 8 | 4.13 | 1.0 |
| GeForce GTX 480 | 1 | 0.255 | 16 |
| GeForce GTX 480 | 4 | 0.081 | 51 |


3-D orbital lattice: millions of points
Lattice slices computed on multiple GPUs
2-D CUDA grid on one GPU
CUDA thread blocks
GPU threads each compute one point.

## Molecular Orbital Inner Loop, Hand-Coded x86 SSE
## Hard to Read, Isn't It? (And this is the "pretty" version!)

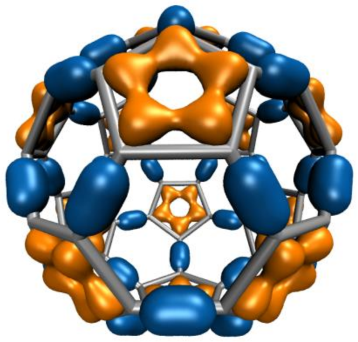


```
for (shell=0; shell < maxshell; shell++) {
    __m128 Cgto = _mm_setzero_ps();
    for (prim=0; prim<num_prim_per_shell[shell_counter]; prim++) {
        float exponent        = -basis_array[prim_counter     ];
        float contract_coeff =  basis_array[prim_counter + 1];
        __m128 expval = _mm_mul_ps(_mm_load_ps1(&exponent), dist2);
        __m128 ctmp = _mm_mul_ps(_mm_load_ps1(&contract_coeff), exp_ps(expval));
        Cgto = _mm_add_ps(contracted_gto, ctmp);
        prim_counter += 2;
    }
    __m128 tshell = _mm_setzero_ps();
    switch (shell_types[shell_counter]) {
        case S_SHELL:
            value = _mm_add_ps(value, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), Cgto));    break;
        case P_SHELL:
            tshell = _mm_add_ps(tshell, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), xdist));
            tshell = _mm_add_ps(tshell, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), ydist));
            tshell = _mm_add_ps(tshell, _mm_mul_ps(_mm_load_ps1(&wave_f[ifunc++]), zdist));
            value = _mm_add_ps(value, _mm_mul_ps(tshell, Cgto));      break;
```

Writing SSE kernels for CPUs requires assembly language, compiler intrinsics, various libraries, or a really smart autovectorizing compiler **and lots of luck...**

# Molecular Orbital Inner Loop in CUDA

```
for (shell=0; shell < maxshell; shell++) {
    float contracted_gto = 0.0f;
    for (prim=0; prim<num_prim_per_shell[shell_counter]; prim++) {
      float exponent       = const_basis_array[prim_counter    ];
      float contract_coeff = const_basis_array[prim_counter + 1];
      contracted_gto += contract_coeff * exp2f(-exponent*dist2);
      prim_counter += 2;
    }
    float tmpshell=0;
    switch (const_shell_symmetry[shell_counter]) {
      case S_SHELL:
        value += const_wave_f[ifunc++] * contracted_gto;   break;
      case P_SHELL:
        tmpshell += const_wave_f[ifunc++] * xdist;
        tmpshell += const_wave_f[ifunc++] * ydist
        tmpshell += const_wave_f[ifunc++] * zdist;
        value += tmpshell * contracted_gto;   break;
```

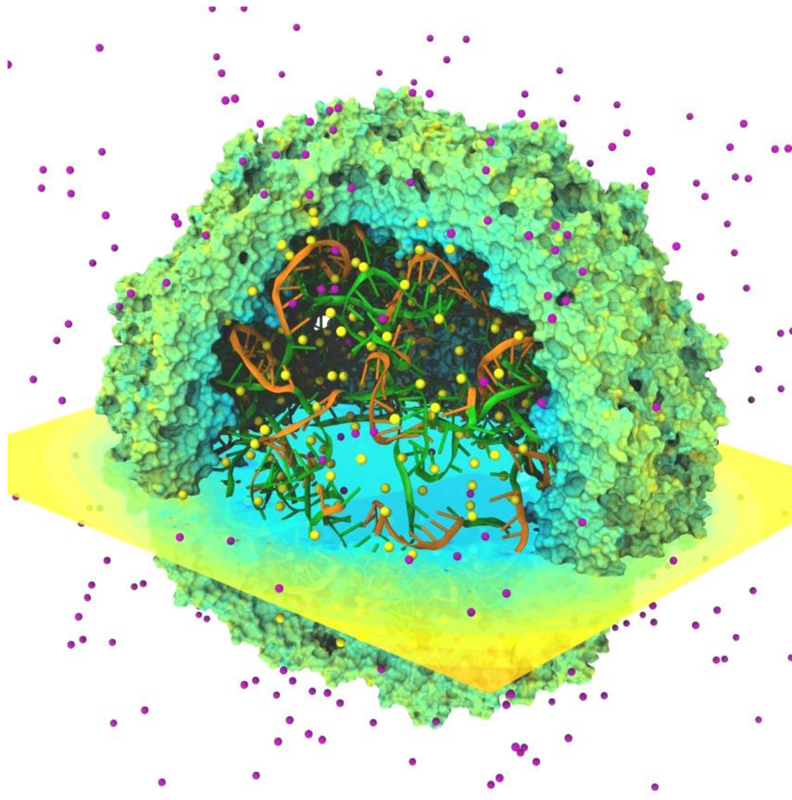


Aaaaahhhh….

Data-parallel CUDA kernel looks like normal C code for the most part….

# Time-Averaged Electrostatics Analysis on Energy-Efficient GPU Cluster

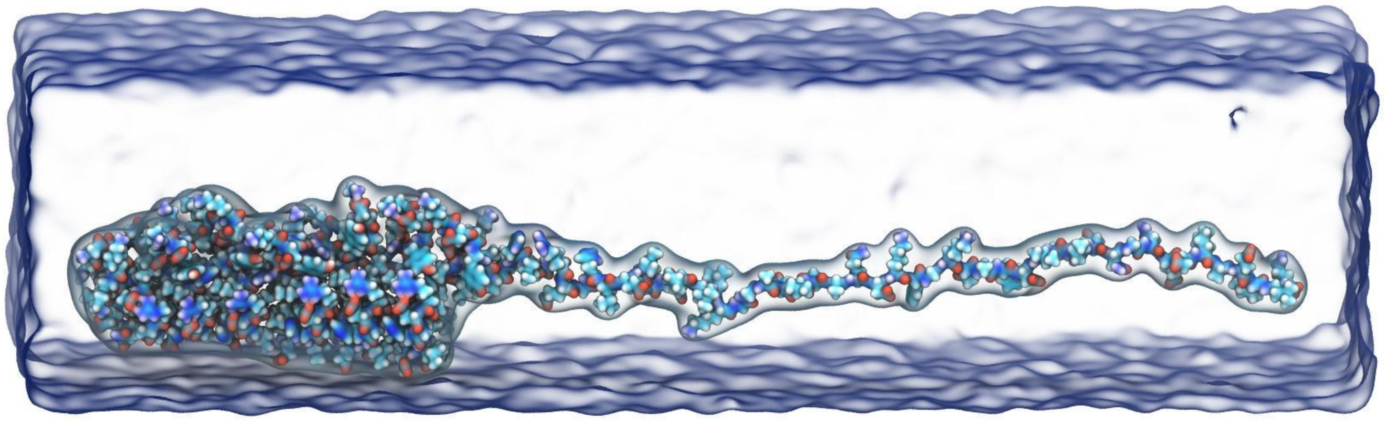
- **1.5 hour** job (CPUs) reduced to **3 min** (CPUs+GPU)
- Electrostatics of thousands of trajectory frames averaged
- Per-node power consumption on NCSA "AC" GPU cluster:
  - CPUs-only: 448 Watt-hours
  - CPUs+GPUs: 43 Watt-hours
- GPU Speedup: **25.5x**
- Power efficiency gain: **10.5x**



**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.

# 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

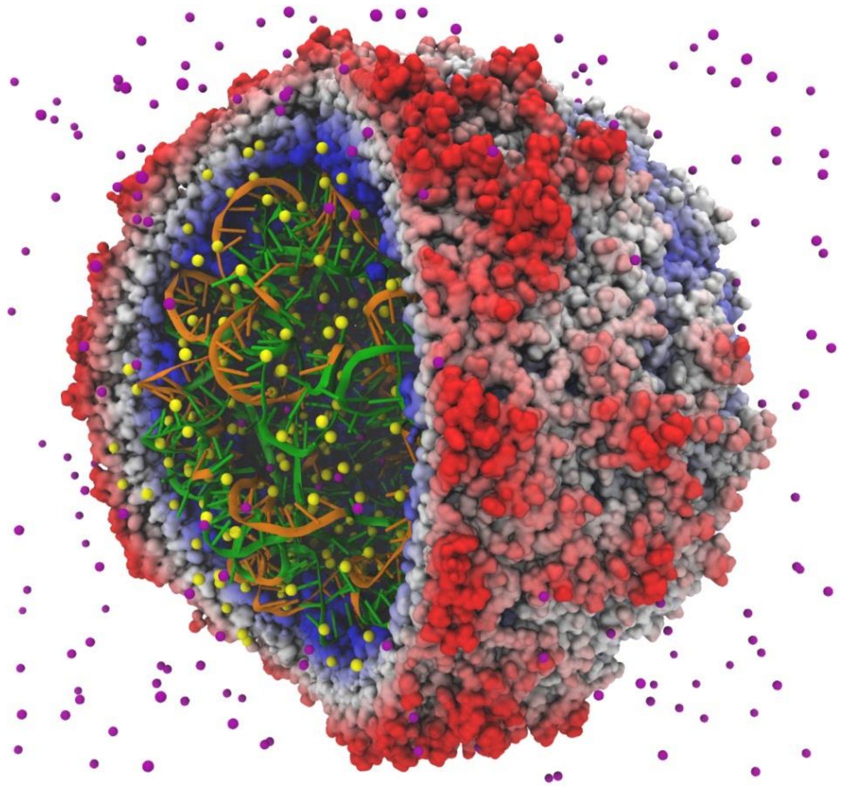


**Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.**

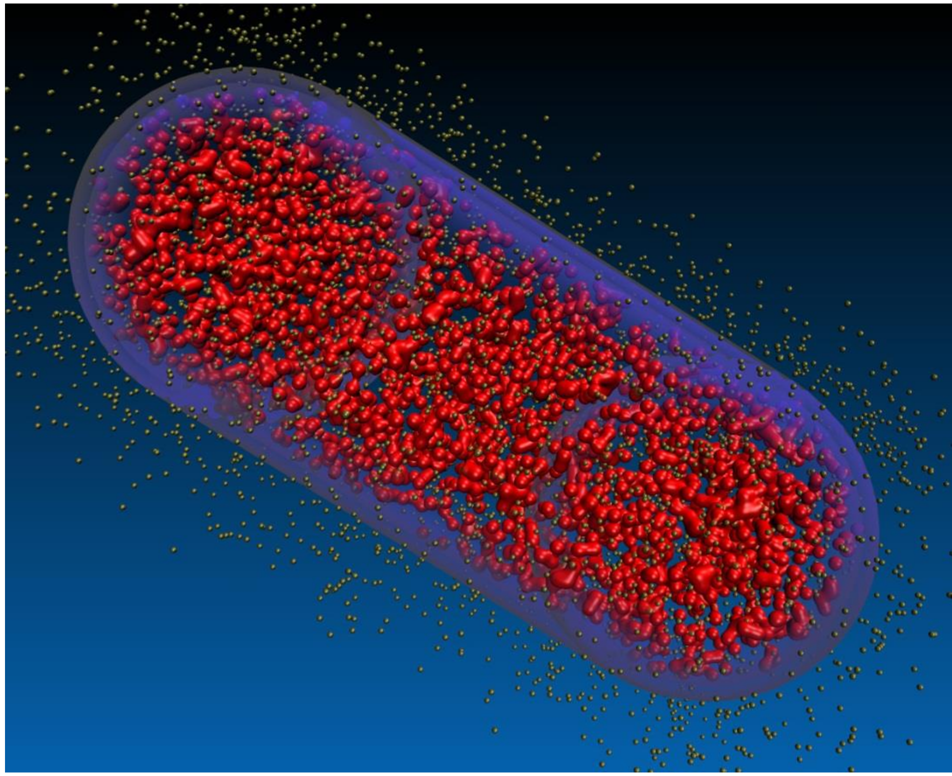
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

# 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**



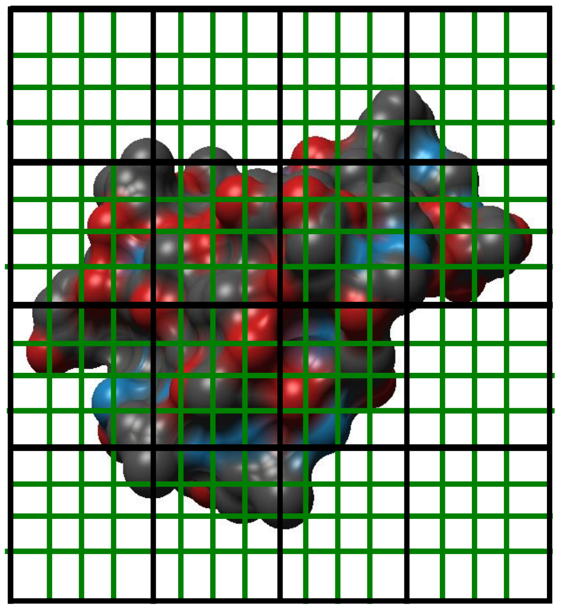
**Lattice Cell Simulations**

# QuickSurf Algorithm Overview

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

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

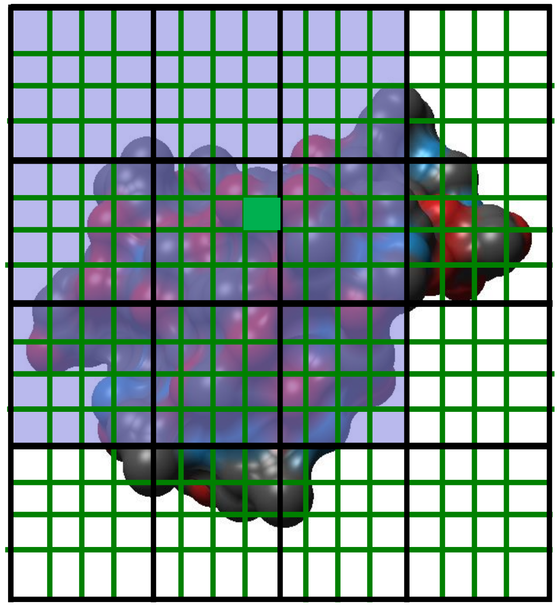
- Extract isosurface for a user-defined density value



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

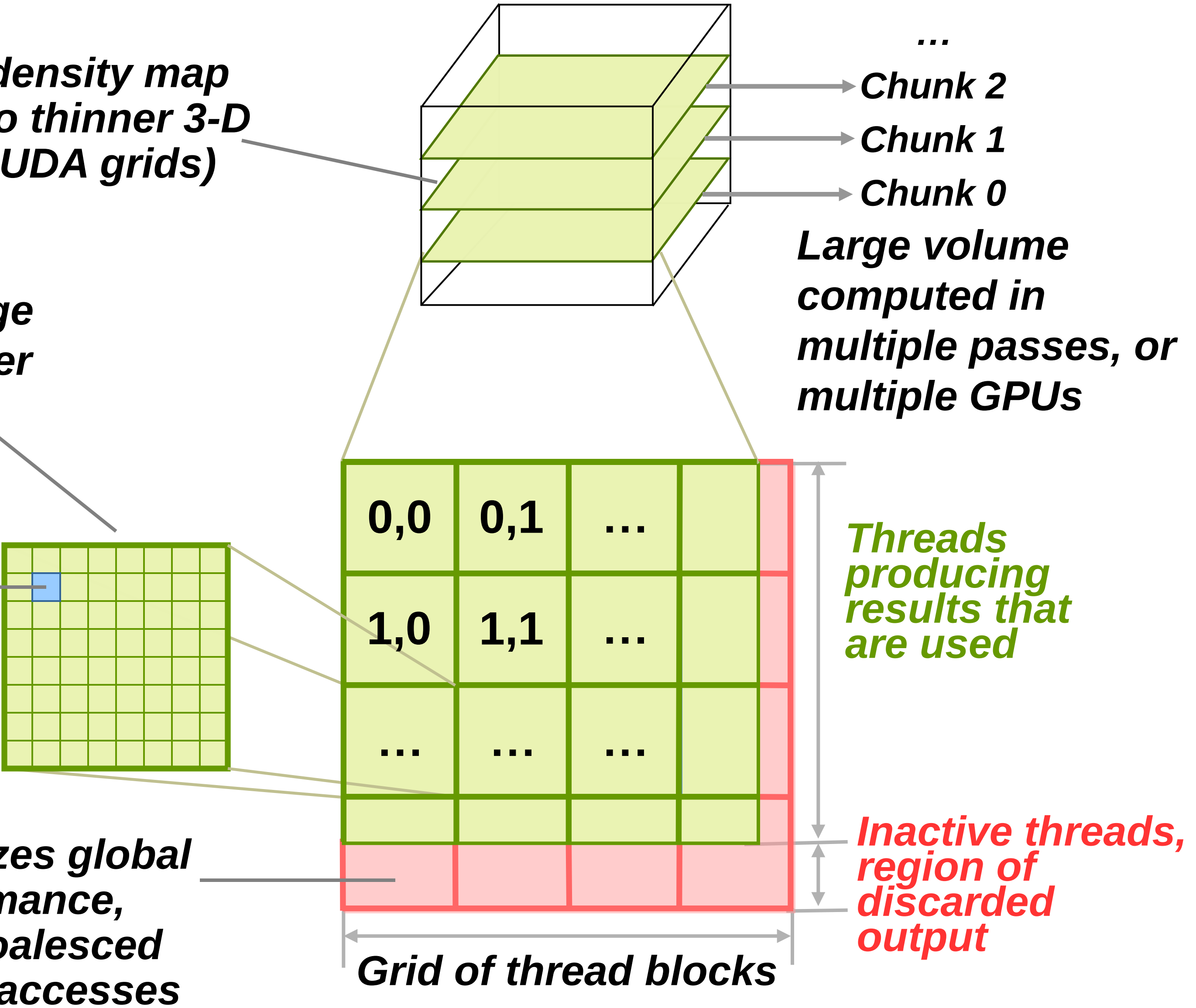
# QuickSurf Density Map Algorithm

- 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



...
Chunk 2
Chunk 1
Chunk 0
0,0
0,1
...
1,0
1,1
...
...
...
...
Threads producing results that are used
Inactive threads, region of discarded output
Grid of thread blocks

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

*Large volume computed in multiple passes, or multiple GPUs*

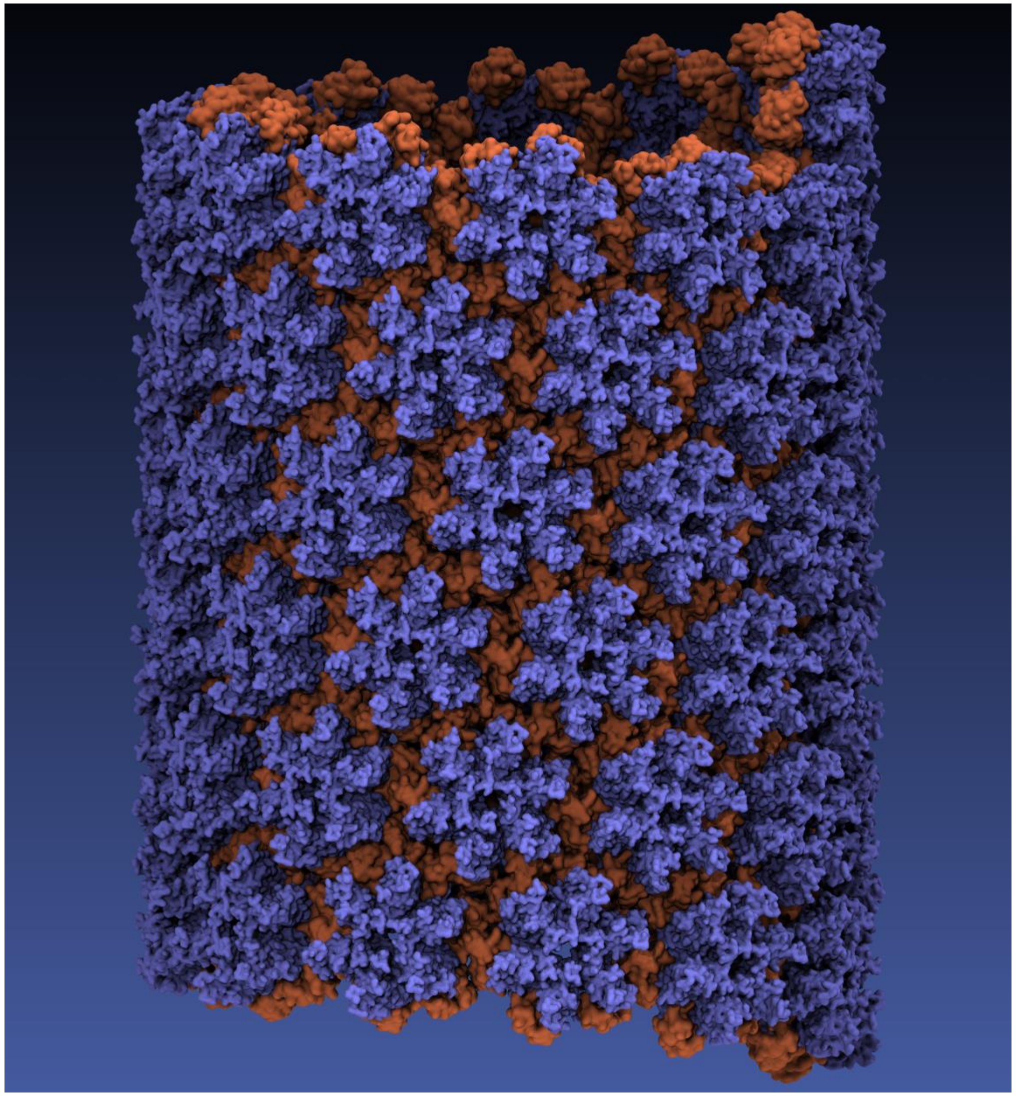
*Small 8x8 thread blocks afford large per-thread register count, shared memory*

*Each thread computes one or more density map lattice points*

*Padding optimizes global memory performance, guaranteeing coalesced global memory accesses*

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

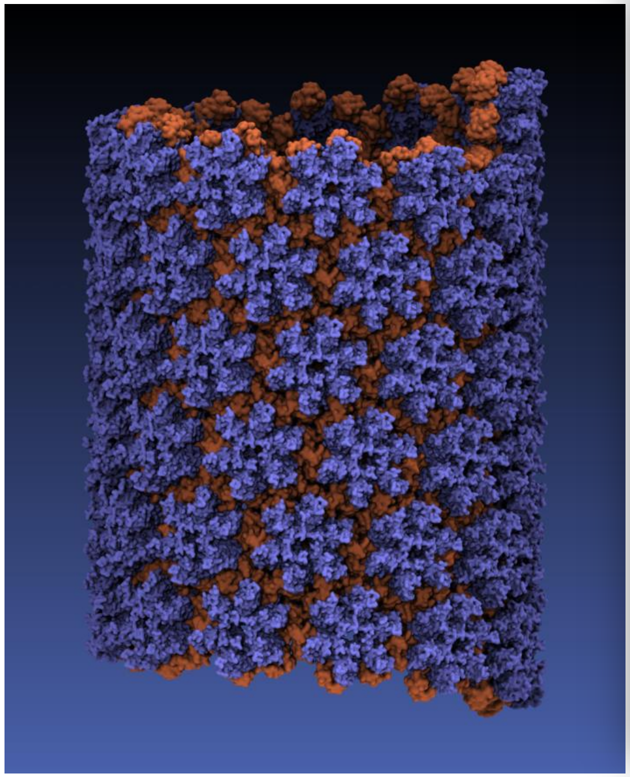
- Structures such as HIV initially needed all XK7 GPU memory to generate detailed surface renderings
- Goals and approach:
  - **Avoid slow CPU-fallback!**
  - Incrementally change algorithm phases to use more compact data types, while maintaining performance
  - Specialize code for different precision/performance/memory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



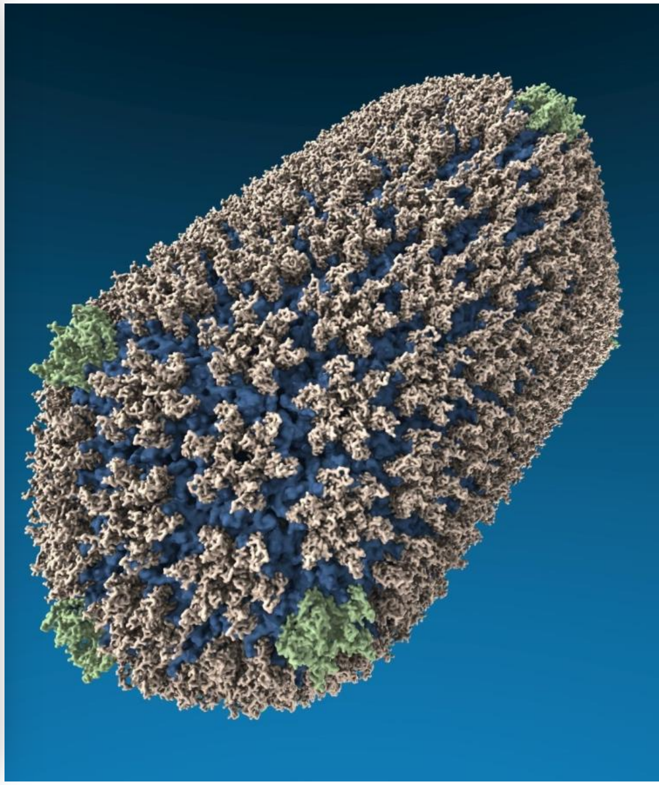
## Net Result of QuickSurf Memory Efficiency Optimizations

- **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!

# VMD "QuickSurf" Representation, Ray Tracing



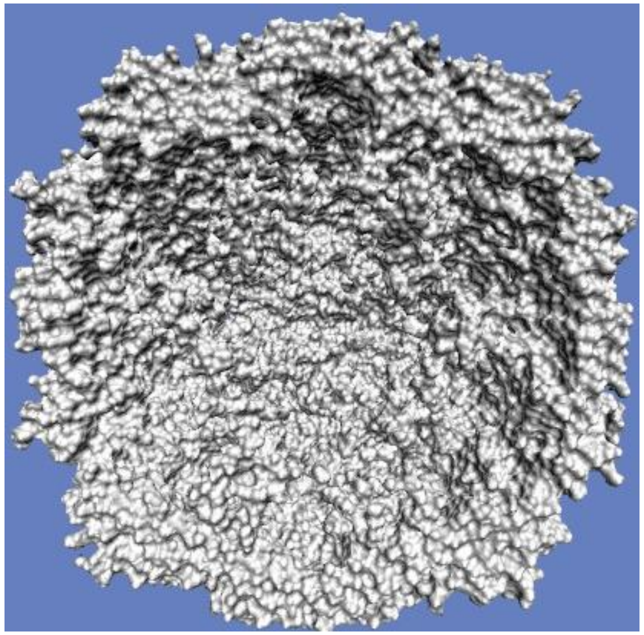
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
THE HIV-1 CAPSID
Atomic structure of the AIDS pathogen's protein coat
PAGE 643
COSMOLOGY
THE FIRST LIGHT
In pursuit of the most distant galaxies
PAGE 554
CITATION
CROSSING THE BORDERS
International collaborations make the most impact
PAGE 557
ANTICANCER DRUGS
A SITTING TARGET
An indirect hit on 'undruggable' KRAS protein
PAGES 577 & 638
NATURE.COM/NATURE
30 May 2013

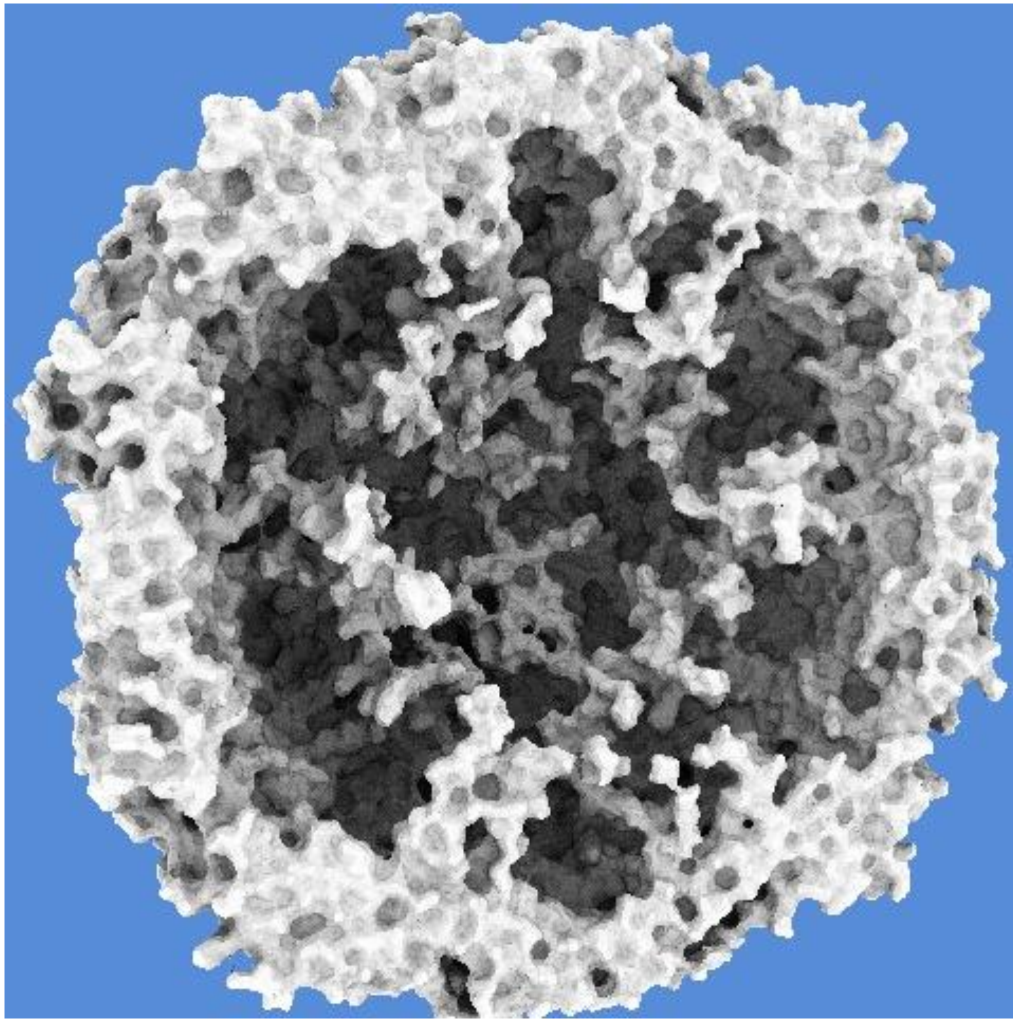


**All-atom HIV capsid simulations w/ up to 64M atoms on Blue Waters**

## Ray Tracing Molecular Graphics

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


Standard OpenGL rasterization

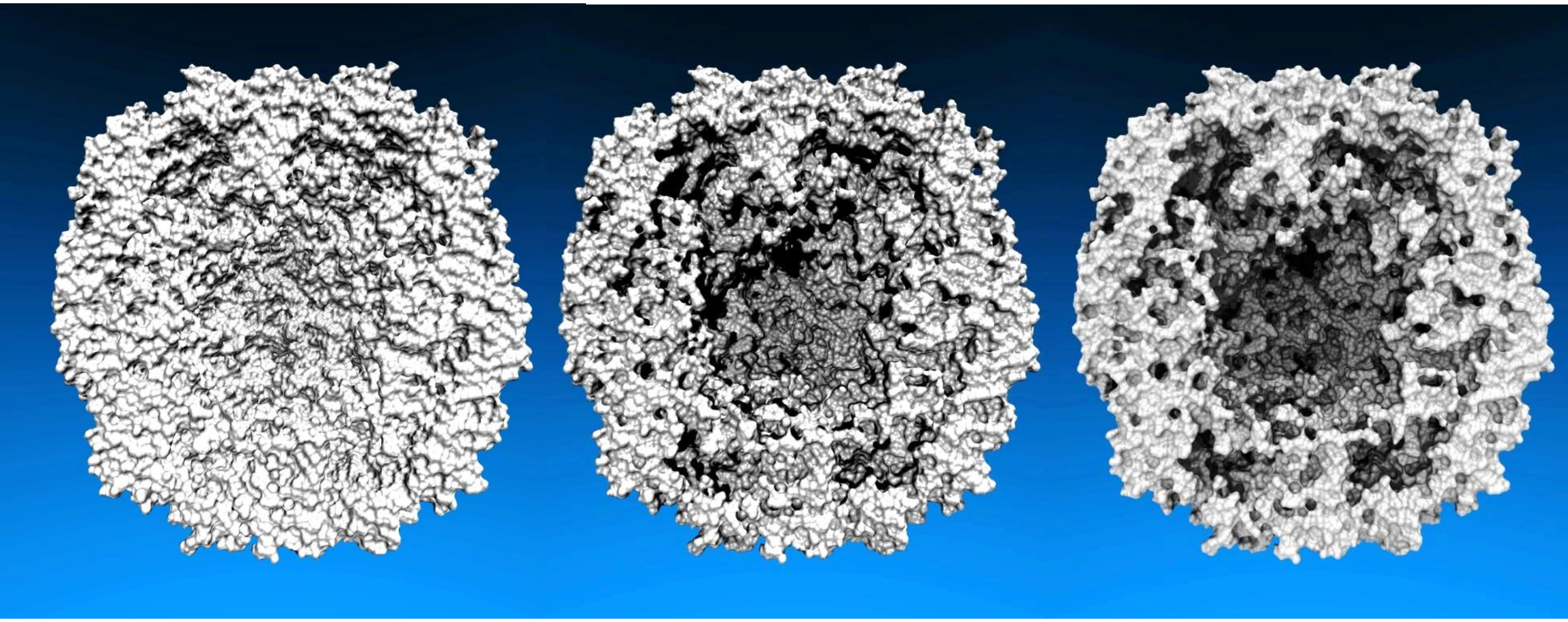

VMD w/ new GPU ray tracing engine based on CUDA + OptiX

# Lighting Comparison

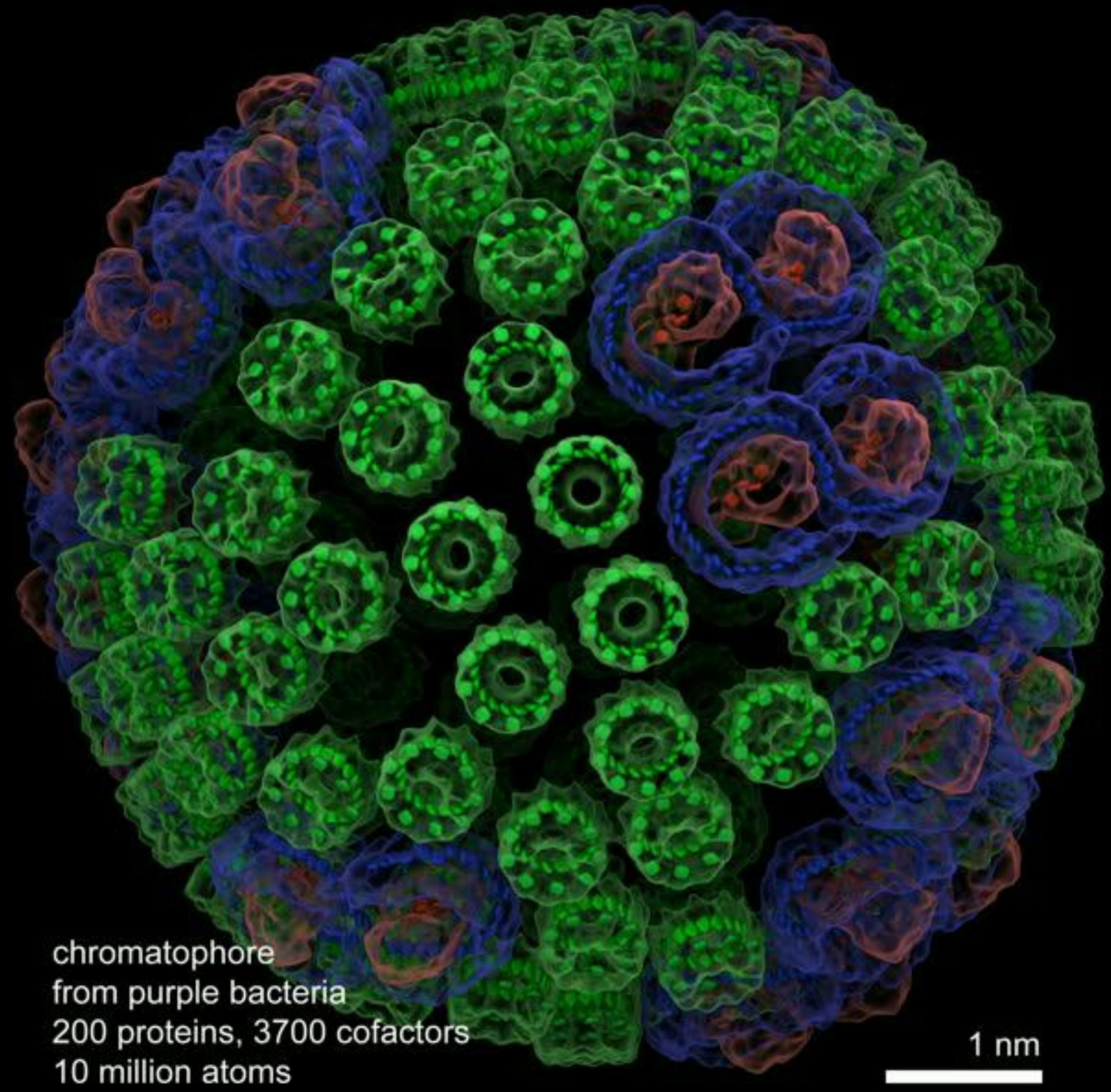
**Two lights, no shadows**

**Two lights, hard shadows, 1 shadow ray per light**

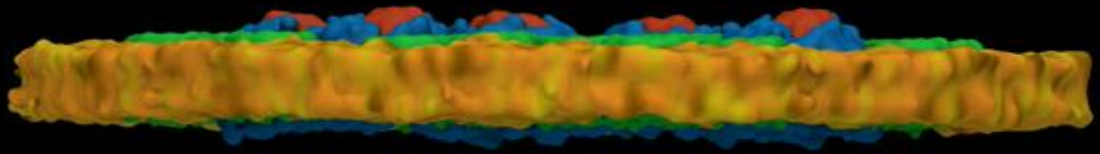
**Ambient occlusion + two lights, 144 AO rays/hit**



## BW VMD/Tachyon Movie Generation



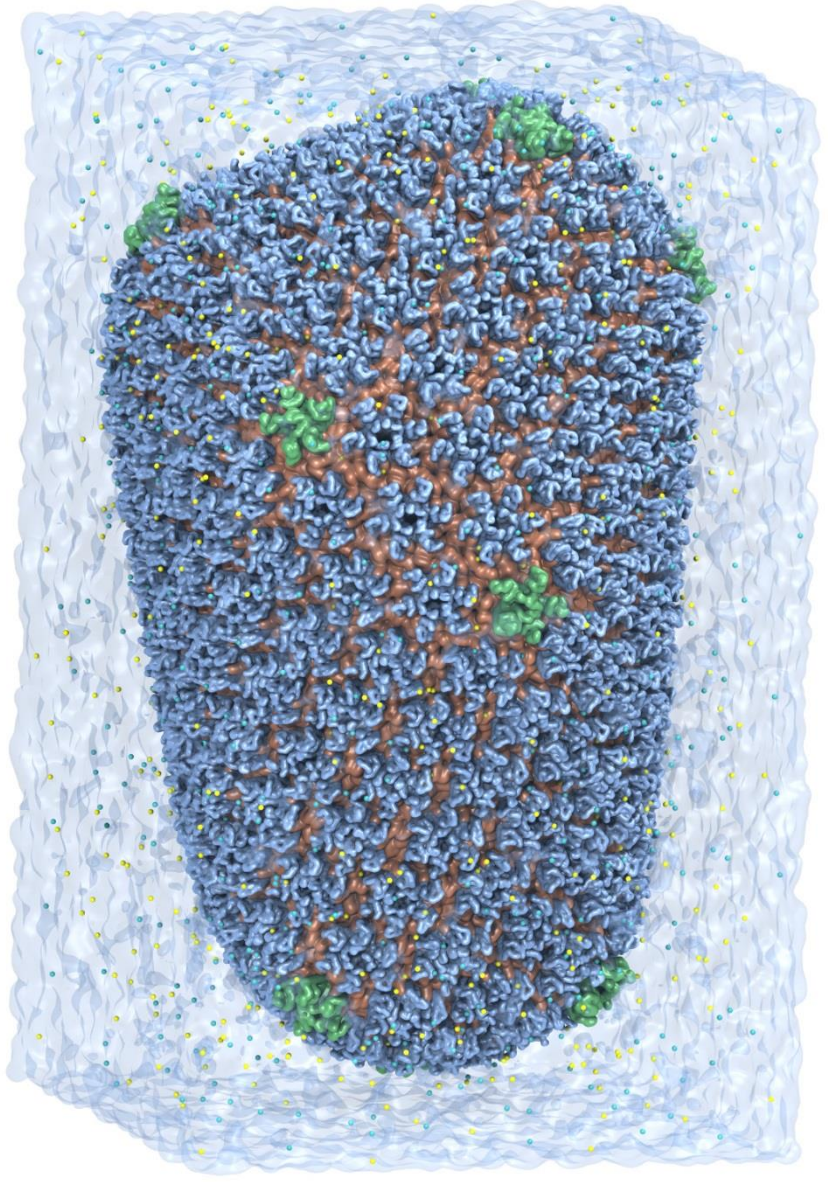
## BW VMD/Tachyon Movie Generation



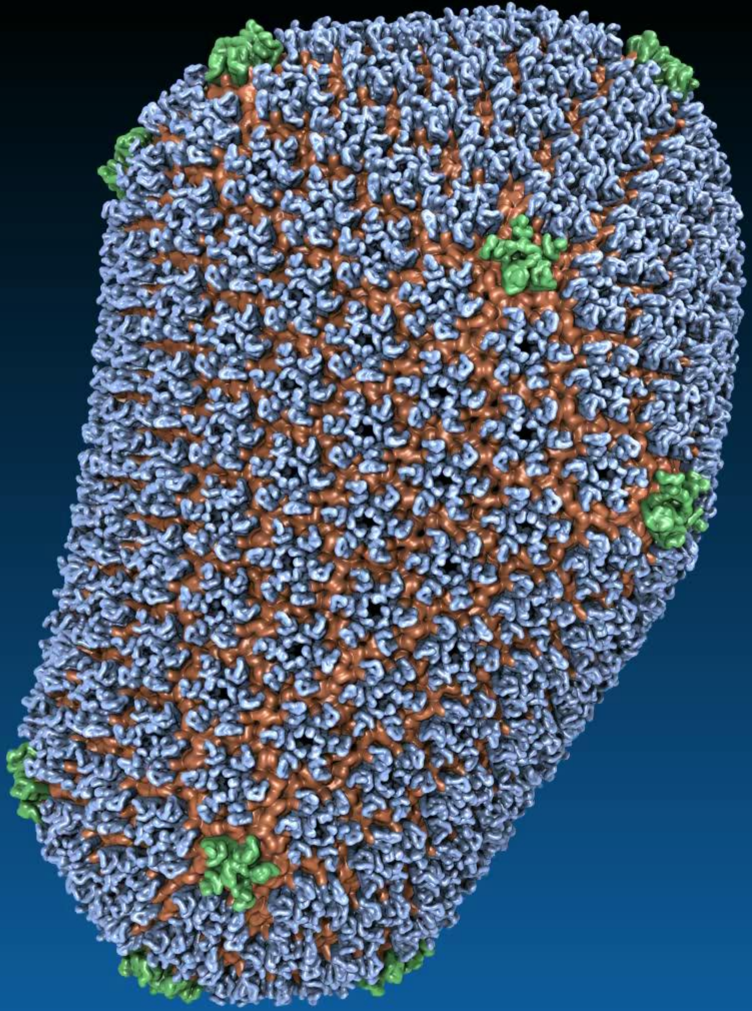
20 M atom chromatophore patch

# GPU Ray Tracing of HIV-1 on Blue Waters

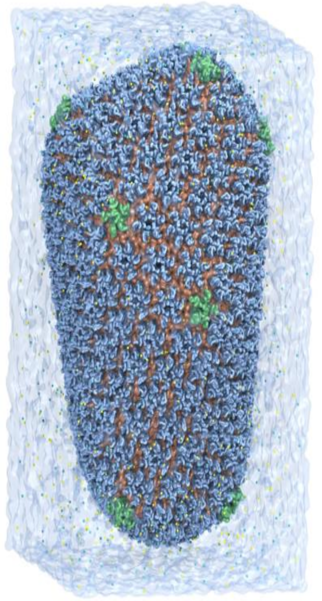
- **Ambient occlusion lighting**, shadows, transparency, antialiasing, depth cueing, **144 rays/pixel minimum**
- 64 million atom virus simulation
- 1000+ movie frames
- Surface generation and ray tracing stages each use >= 75% of GPU memory



# VMD GPU Ray Tracing of HIV-1 Capsid



# 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

Cray XE6: 2x Opteron 62xx CPUs (32-cores)

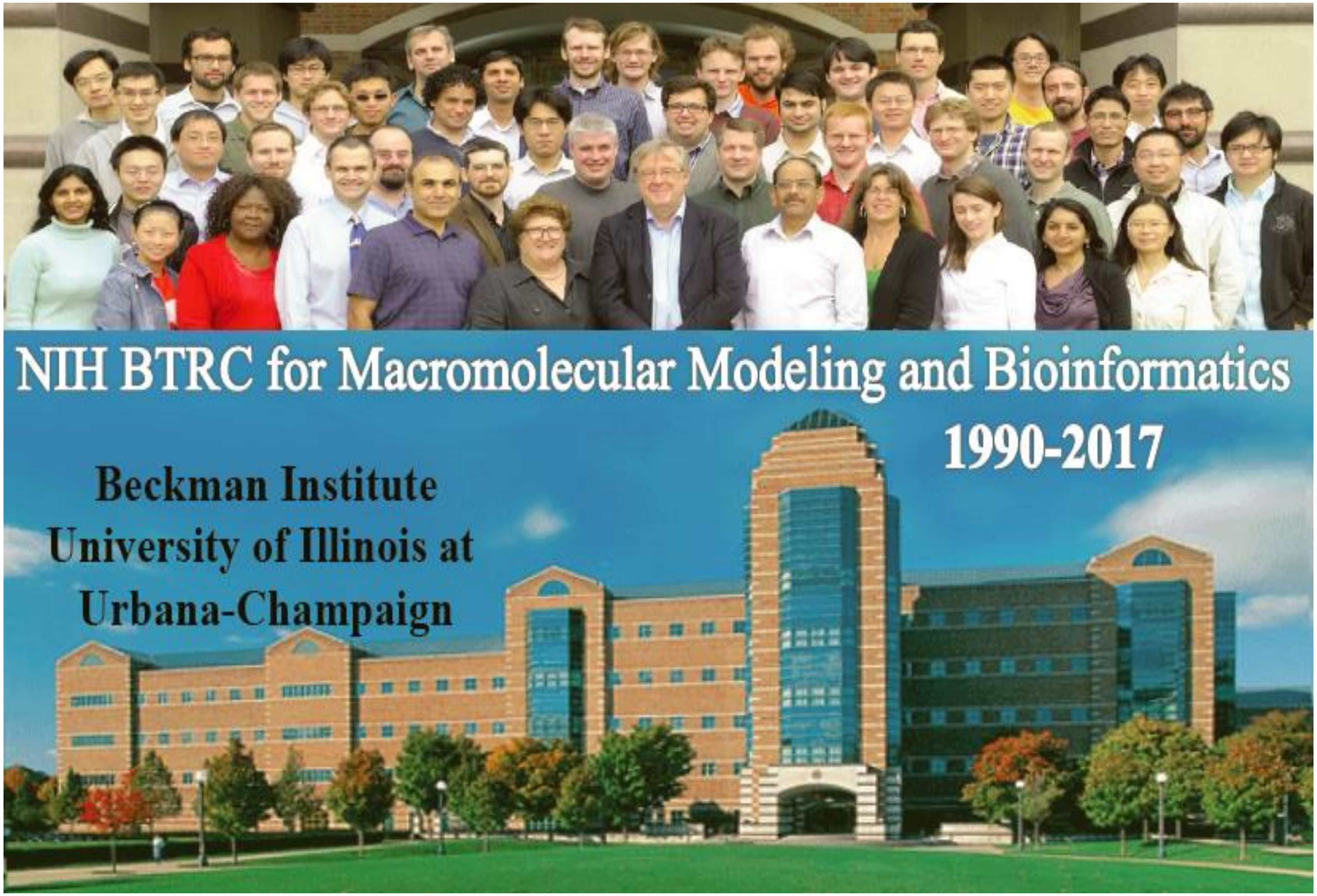
Cray XK7: 1x Opteron 62xx CPU (16-cores) + NVIDIA Tesla K20X

| **Node Type and Count** | **Script Load Time** | **State Load Time** | **Geometry + Ray Tracing** | **Total Time** |
|---|---|---|---|---|
| **256 XE6 CPU nodes** | 7 s | 160 s | **1,374 s** | **1,541 s** |
| 512 XE6 CPU nodes | 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** |

**GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms.**
Stone et al. In UltraVis'13: Eighth Workshop on Ultrascale Visualization Proceedings, 2013.

## 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


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/

- **Runtime and Architecture Support for Efficient Data Exchange in Multi-Accelerator Applications** Javier Cabezas, Isaac Gelado, John E. Stone, Nacho Navarro, David B. Kirk, and Wen-mei Hwu. IEEE Transactions on Parallel and Distributed Systems, 2014. (Accepted)
- **Unlocking the Full Potential of the Cray XK7 Accelerator** Mark Klein and John E. Stone. Cray Users Group, 2014. (In press)
- **Simulation of reaction diffusion processes over biologically relevant size and time scales using multi-GPU workstations** Michael J. Hallock, John E. Stone, Elijah Roberts, Corey Fry, and Zaida Luthey-Schulten. Journal of Parallel Computing, 2014. (In press)
- **GPU-Accelerated Analysis and Visualization of Large Structures Solved by Molecular Dynamics Flexible Fitting** John E. Stone, Ryan McGreevy, Barry Isralewitz, and Klaus Schulten. Faraday Discussion 169, 2014. (In press)
- **GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms.** J. Stone, K. L. Vandivort, and K. Schulten. UltraVis'13: Proceedings of the 8th International Workshop on Ultrascale Visualization, pp. 6:1-6:8, 2013.
- **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.
- **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**.**

# GPU Computing Publications

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

- **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.
- **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.
- **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.
- **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.

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

- **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 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.