VMD: 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/gpu/

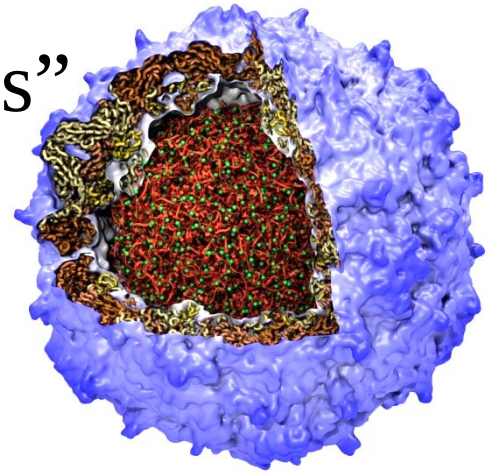
GPU Technology Conference

9:00-9:50AM, Room 212B, San Jose Convention Center,

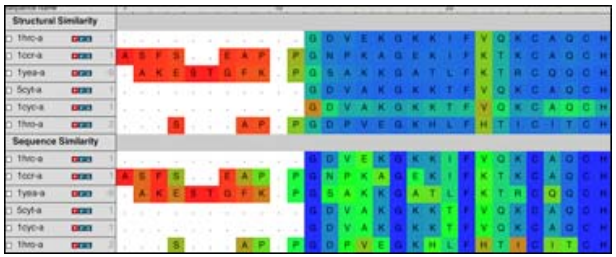
San Jose, CA, March 20, 2013

VMD – “Visual Molecular Dynamics”

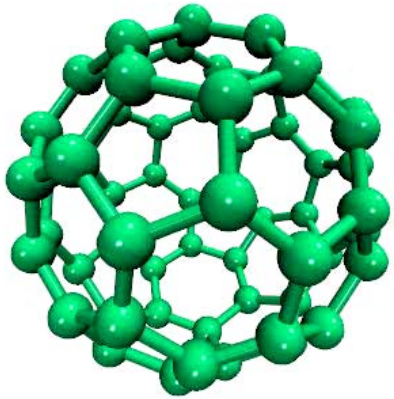
- Visualization and analysis of:
  - molecular dynamics simulations
  - quantum chemistry calculations
  - particle systems and whole cells
  - sequence data
- User extensible w/ scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/



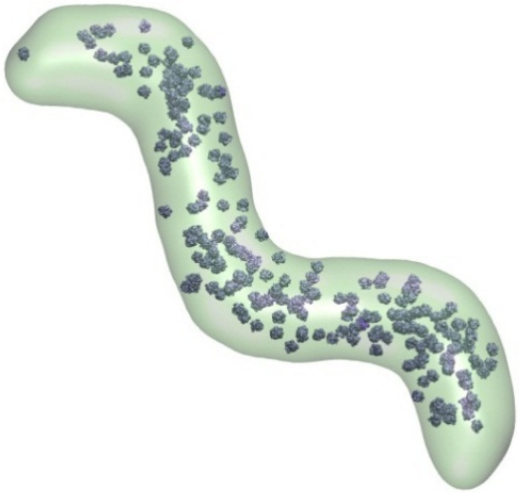
Poliovirus



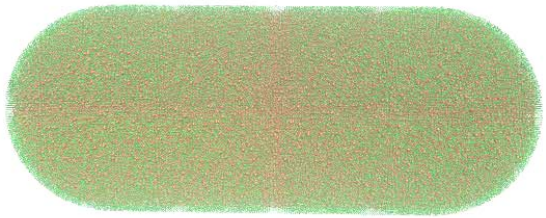
Ribosome Sequences



Electrons in
Vibrating Buckyball



Cellular Tomography,
Cryo-electron Microscopy



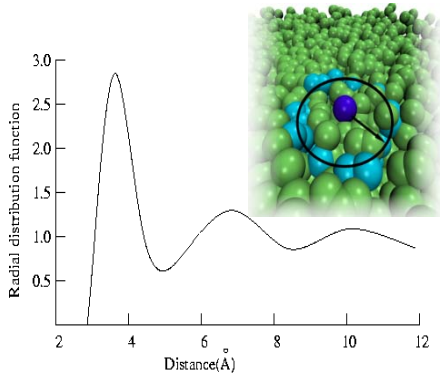
Whole Cell Simulations

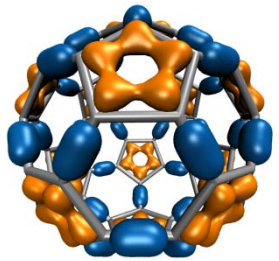
Ongoing VMD GPU Development

- Development of new CUDA kernels for common molecular dynamics trajectory analysis tasks
- Increased memory efficiency of CUDA kernels for visualization and analysis of large structures
- Improving CUDA performance for batch mode MPI version of VMD used for in-place trajectory analysis calculations:
  - GPU-accelerated commodity clusters
  - GPU-accelerated Cray XK7 supercomputers: NCSA Blue Waters, ORNL Titan

GPU Accelerated Trajectory Analysis and Visualization in VMD

GPU-Accelerated Feature	Peak speedup vs. 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





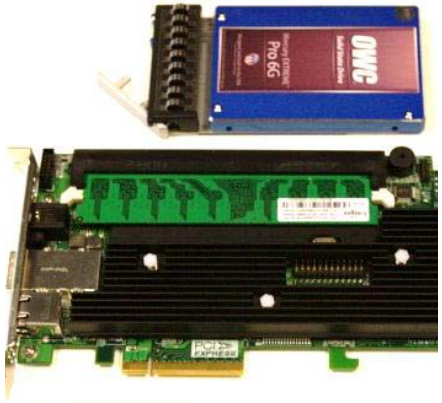
Interactive Display & Analysis of Terabytes of Data:

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

450MB/sec

to 8GB/sec

TWO DVD movies per second!



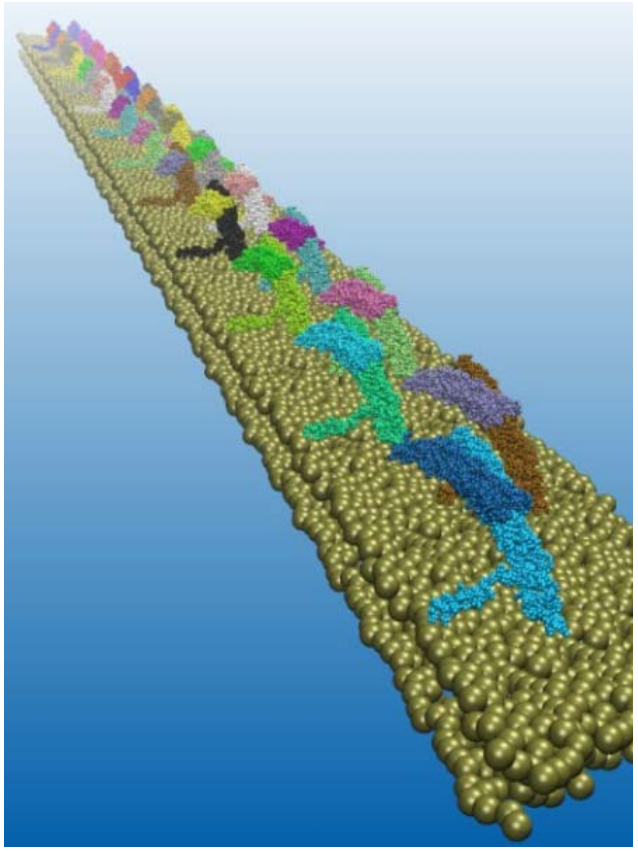
Commodity SSD, SSD RAID

- 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

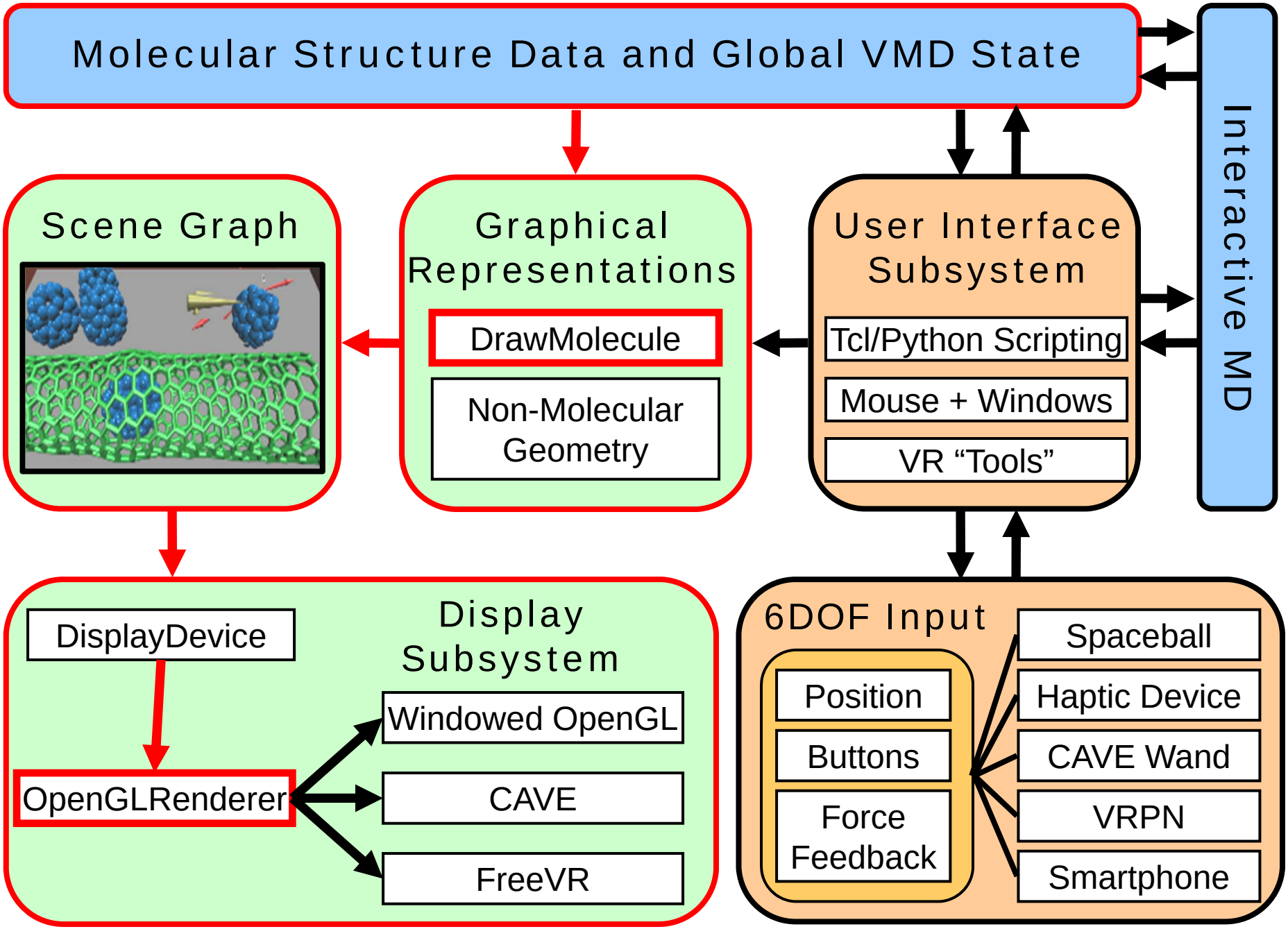
Immersive out-of-core visualization of large-size and long-timescale molecular dynamics trajectories. J. Stone, K. Vandivort, and K. Schulten. *Lecture Notes in Computer Science*, 6939:1-12, 2011.

Challenges for Immersive Visualization of Dynamics of Large Structures

- Graphical representations re-computed each trajectory timestep
- Visualizations often focus on interesting regions of substructure
- Fast display updates require rapid sparse traversal+gathering of molecular data for use in GPU computations and OpenGL display
  - Hand-vectorized SSE/AVX CPU atom selection traversal code increased performance of per-frame updates by another ~6x for several 100M atom test cases
- Graphical representation optimizations:
  - Reduce host-GPU bandwidth for displayed geometry
  - Optimized graphical representation generation routines for large atom counts, sparse selections



116M atom BAR domain test case:
200,000 selected atoms,
stereo trajectory animation 70 FPS,
static scene in stereo 116 FPS

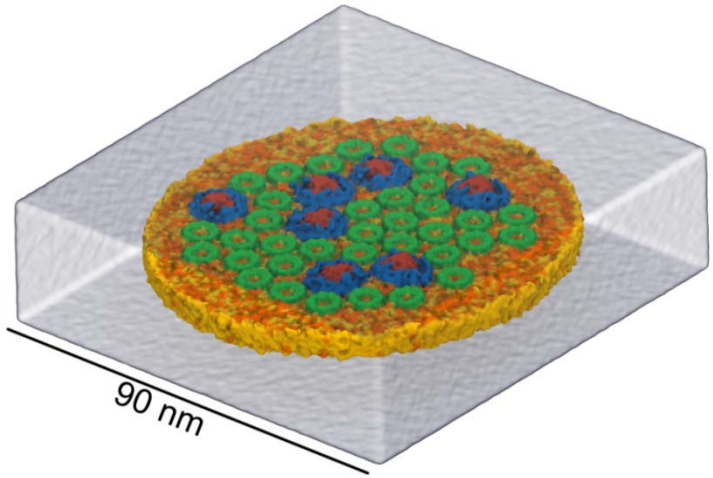

Molecular Structure Data and Global VMD State
Interactive MD
Scene Graph
Graphical Representations
DrawMolecule
Non-Molecular Geometry
User Interface Subsystem
Tcl/Python Scripting
Mouse + Windows
VR "Tools"
DisplayDevice
Display Subsystem
OpenGLRenderer
Windowed OpenGL
CAVE
FreeVR
6DOF Input
Position
Buttons
Force Feedback
Spaceball
Haptic Device
CAVE Wand
VRPN
Smartphone

Improving Performance for Large Datasets

- As the performance of GPUs has continued to increase, formerly "insignificant" CPU routines are becoming bottlenecks
  - A key feature of VMD is the ability to perform visualization and analysis operations on arbitrary user-selected subsets of the molecular structure
  - CPU-side atom selection traversal performance has begun to be a potential bottleneck when working with large structures of tens of millions of atoms
  - Both OpenGL rendering and CUDA analysis kernels (currently) depend on the CPU to gather selected atom data into buffers that are sent to the GPU
  - Hand-coded SSE/AVX optimizations have now improved the performance of these CPU preprocessing steps by up to 6x, keeping the CPU "out of the way"

20M atoms:

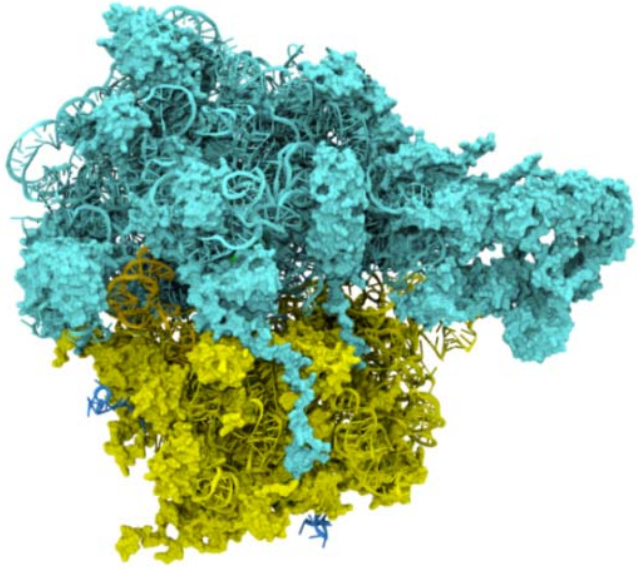
membrane patch and solvent



Improving Performance for Large Datasets: Make Key Data Structures GPU-Resident

- Eliminating the dependency on the host CPU to traverse, collect, and pack atom data will enable much higher GPU performance
- Long-term, best performance will be obtained by storing all molecule data locally in on-board GPU memory
  - GPU needs enough memory to store **both** molecular information, as well as the generated vertex arrays and texture maps used for rendering
  - With sufficient memory, only per-timestep time-varying data will have to copied into the GPU on-the-fly, and most other data can remain GPU-resident
  - Today's GPUs have insufficient memory for very large structures, where the resulting performance increases would have the greatest impact
  - Soon we should begin to see GPUs with 16GB of on-board memory – enough to keep all of the static molecular structure data on the GPU full-time
- Once the full molecular data is GPU-resident, CUDA kernels can directly incorporate atom selection traversal for themselves
- CUDA Dynamic Parallelism will make more GPUs self sufficient

VMD Out-of-Core Trajectory I/O Performance: SSD Trajectory Format, PCIe3 8-SSD RAID

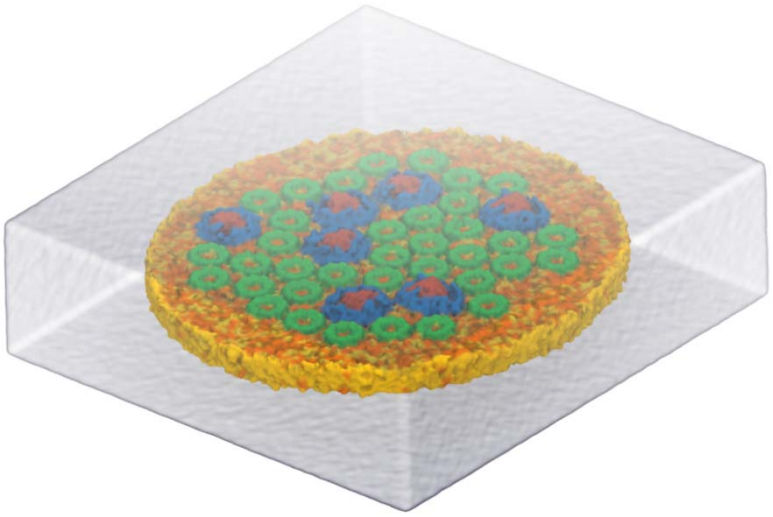


Ribosome w/ solvent

3M atoms

3 frames/sec w/ HD

77 frames/sec w/ SSDs



Membrane patch w/ solvent

20M atoms

0.4 frames/sec w/ HD

10 frames/sec w/ SSDs

New SSD Trajectory File Format 2x Faster vs. Existing Formats

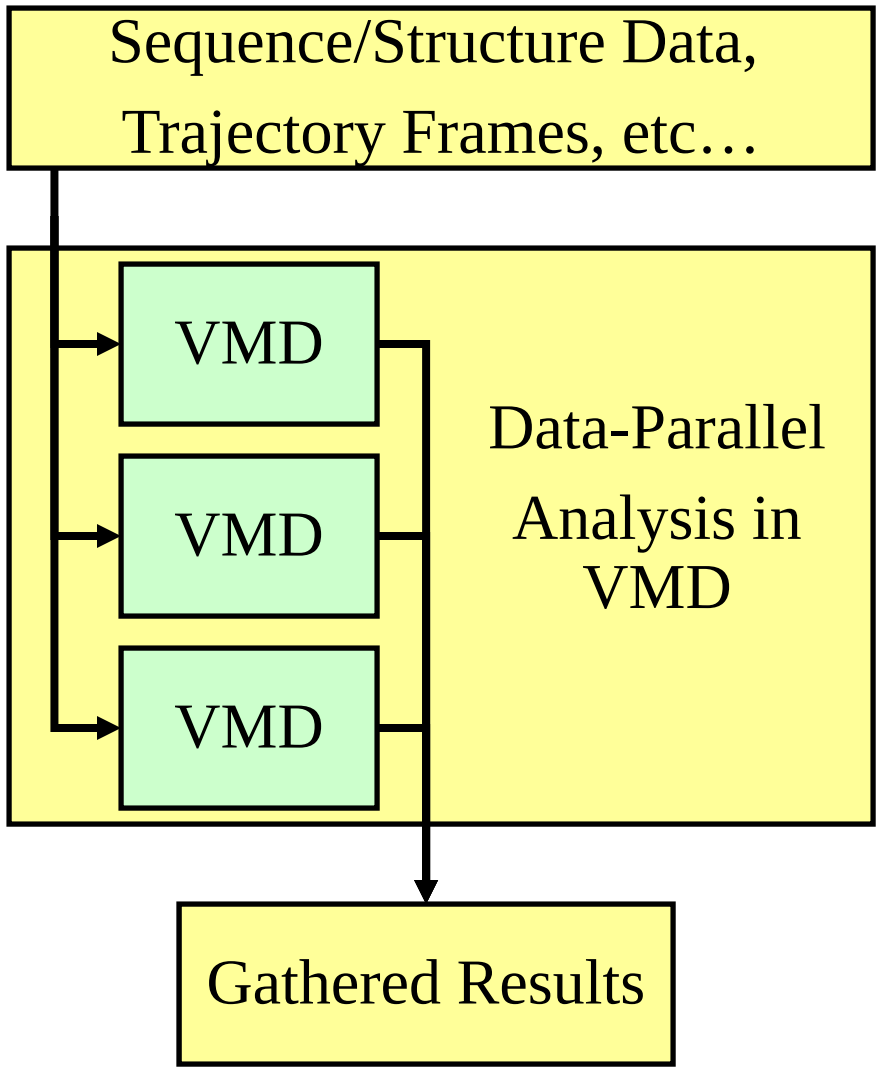
VMD I/O rate ~2.7 GB/sec w/ 8 SSDs in a single PCIe3 RAID0

Challenges for High Throughput Trajectory Visualization and Analysis

- It is not currently possible to fully exploit full I/O bandwidths when streaming data from SSD arrays (>4GB/sec) to GPU global memory **due to copies**
- Need to eliminated copies from disk controllers to host memory – bypass host entirely and perform zero-copy DMA operations straight from disk controllers to GPU global memory
- **Goal: GPUs directly pull in pages from storage systems bypassing host memory entirely**

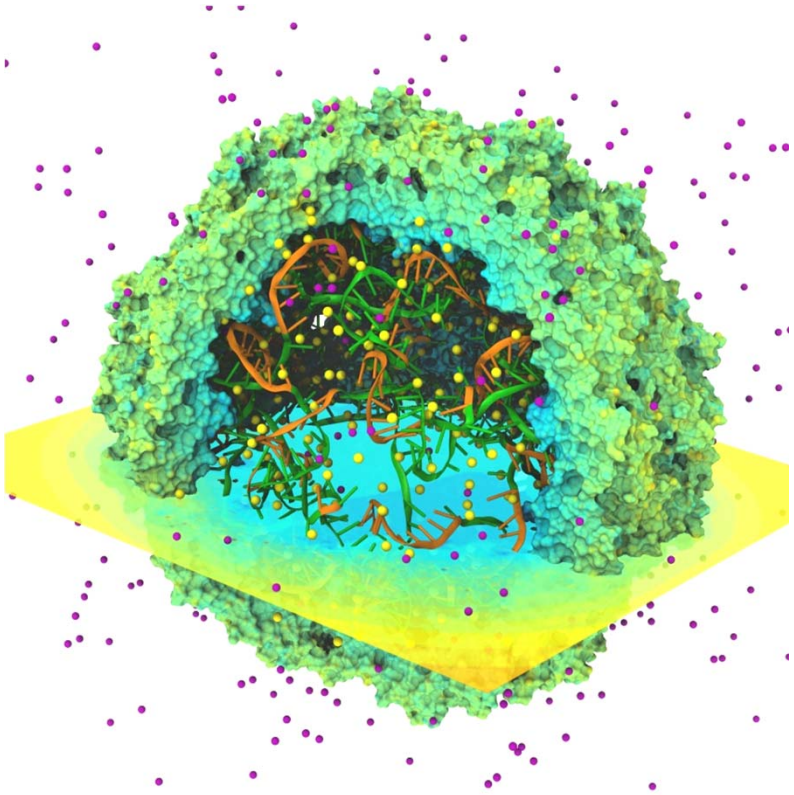
VMD for Demanding Analysis Tasks
Parallel VMD Analysis w/ MPI

- Analyze trajectory frames, structures, or sequences in parallel on clusters and supercomputers:
  - Compute time-averaged electrostatic fields, MDFF quality-of-fit, etc.
  - Parallel rendering, movie making
- Addresses computing requirements beyond desktop
- User-defined parallel reduction operations, data types
- Dynamic load balancing:
  - Tested with up to 15,360 CPU cores
- **Supports GPU-accelerated clusters and supercomputers**



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.

NCSA Blue Waters Early Science System
Cray XK6 nodes w/ NVIDIA Tesla X2090

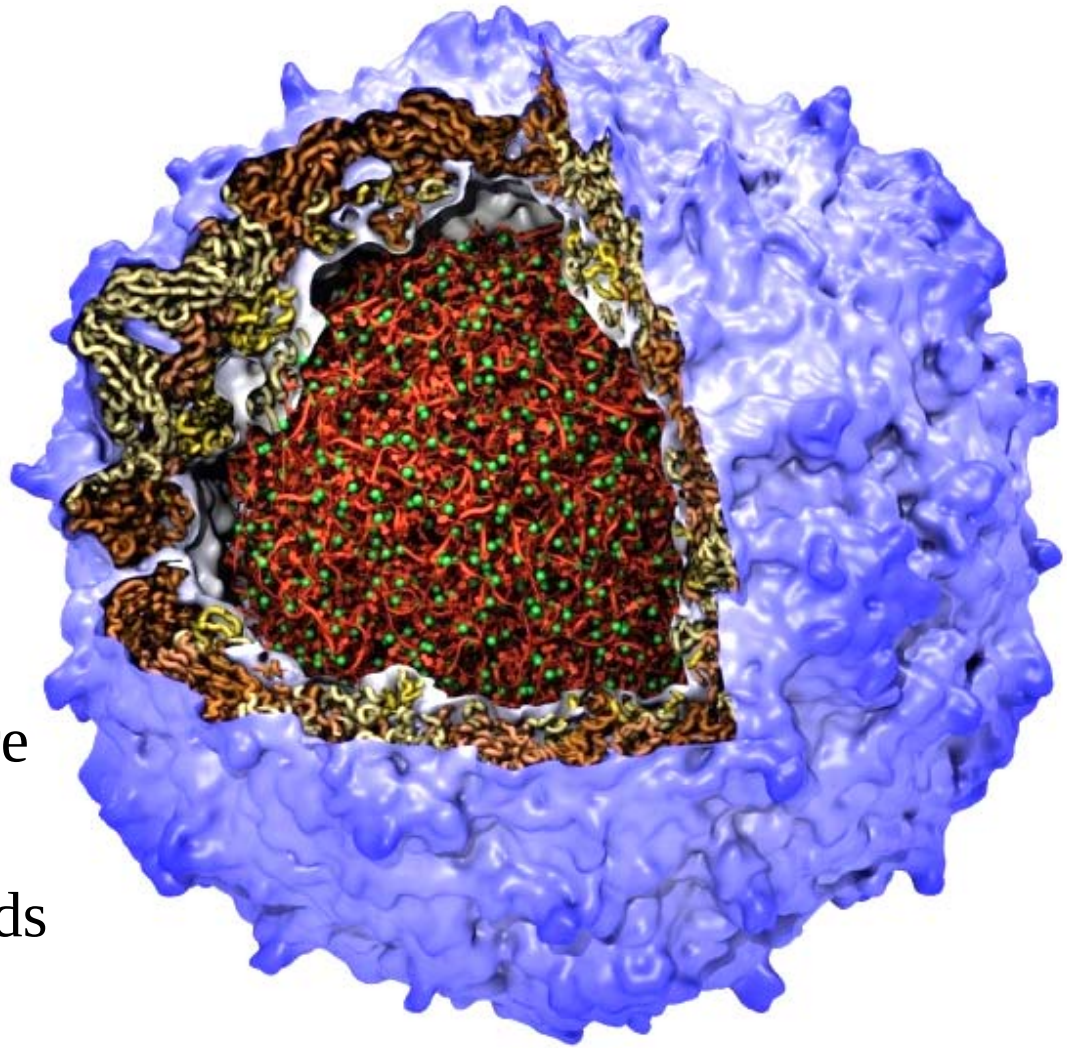
Time-Averaged Electrostatics Analysis on NCSA Blue Waters

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	**GPU nodes are 4.15x faster overall**
Early tests on XK7 nodes indicate MSM is becoming CPU-bound with the Kepler K20X GPU **Performance is not much faster (yet) than Fermi X2090** **May need to move spatial hashing and other algorithms onto the GPU.**	**In progress….**

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

Molecular Surface Visualization

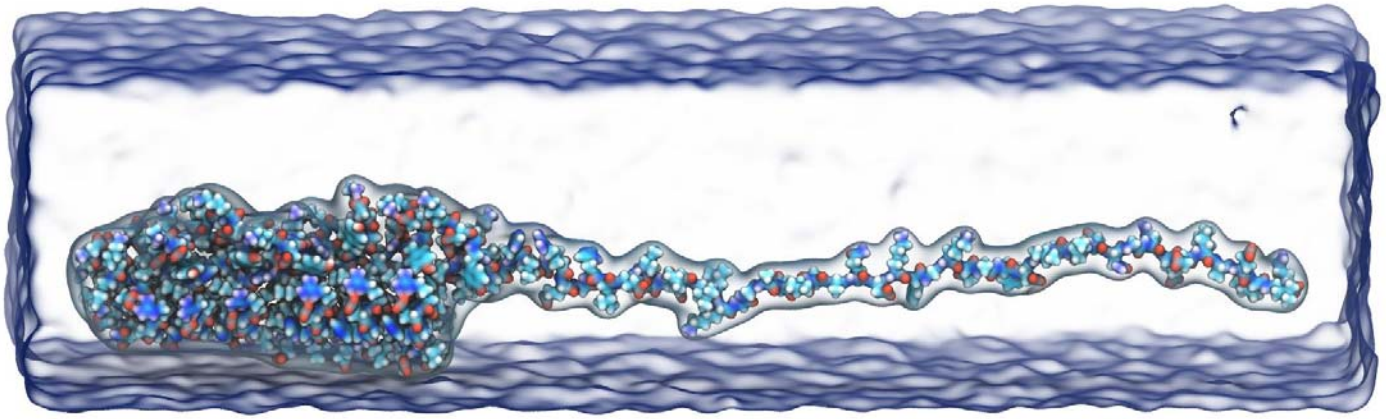
- Large biomolecular complexes are difficult to interpret with atomic detail graphical representations
- Even secondary structure representations become cluttered
- Surface representations are easier to use when greater abstraction is desired, but are computationally costly
- Most surface display methods incapable of animating dynamics of large structures w/ millions of particles



Poliovirus

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

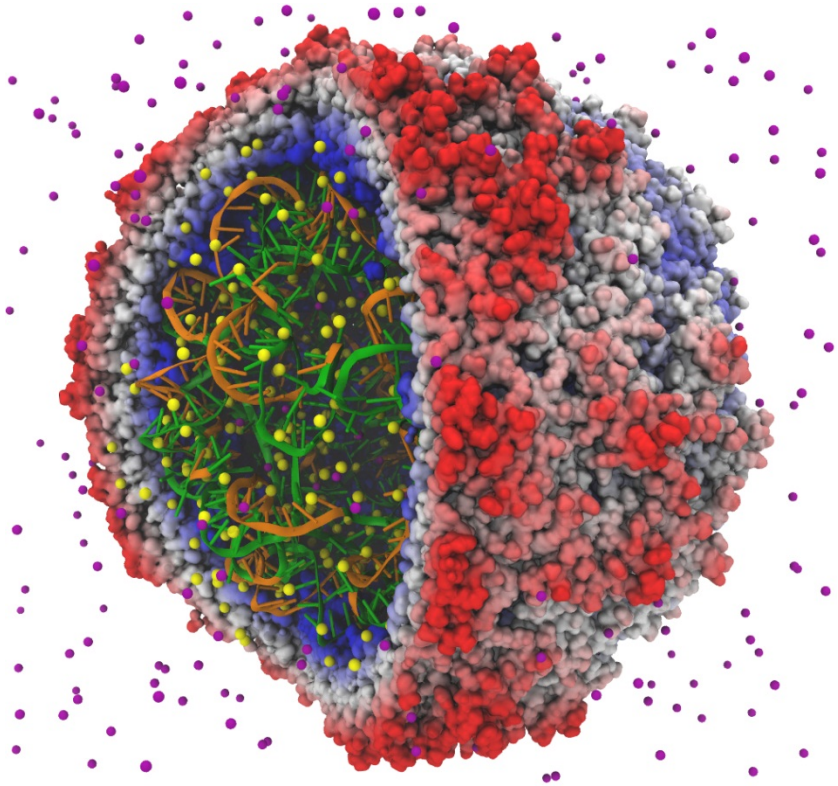


Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories.

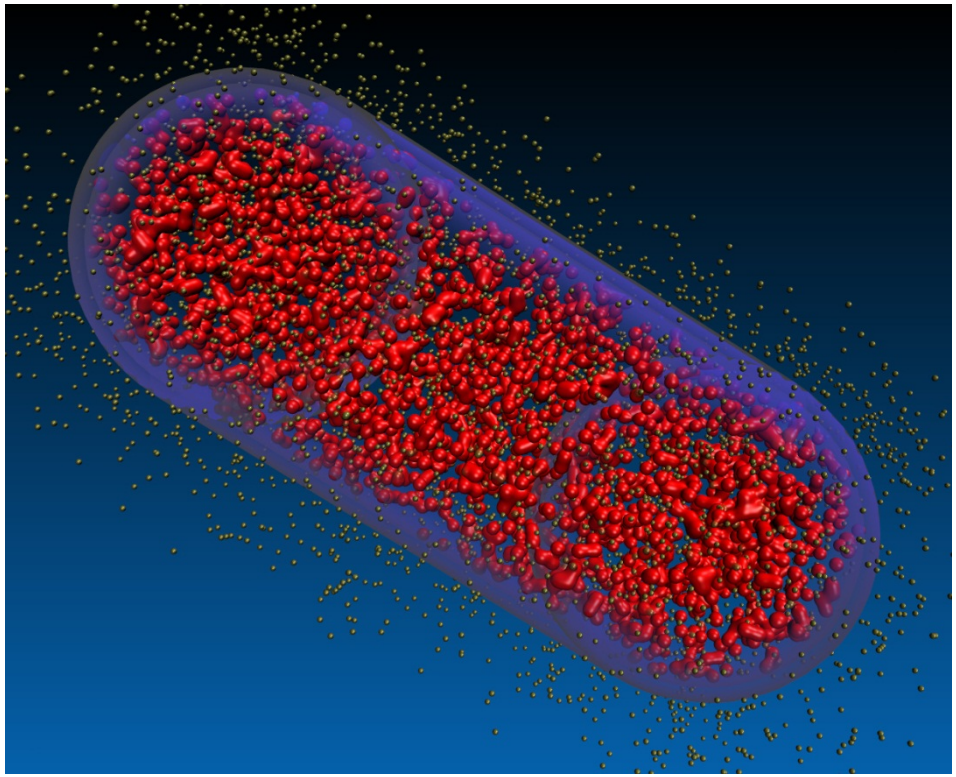
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

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

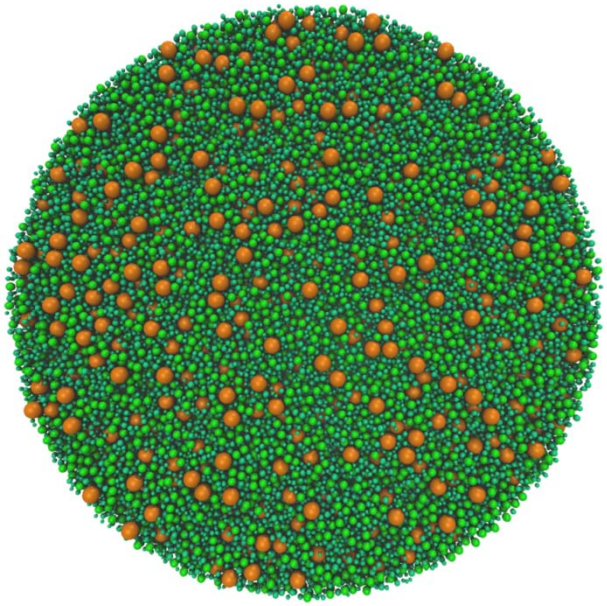


Satellite Tobacco Mosaic Virus

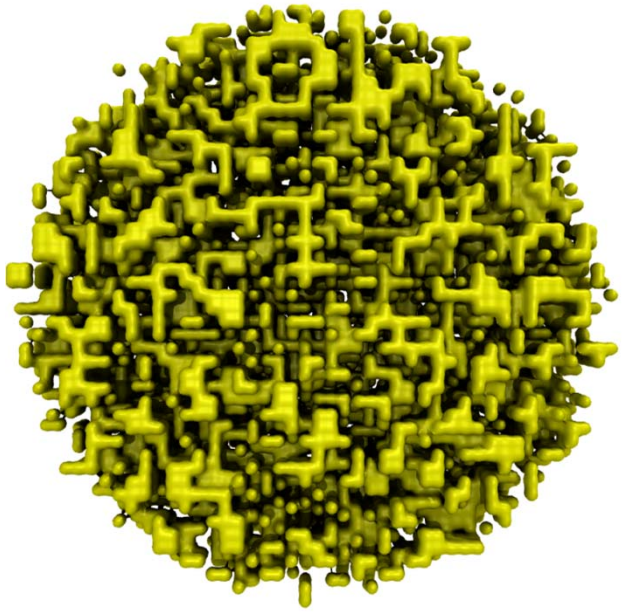


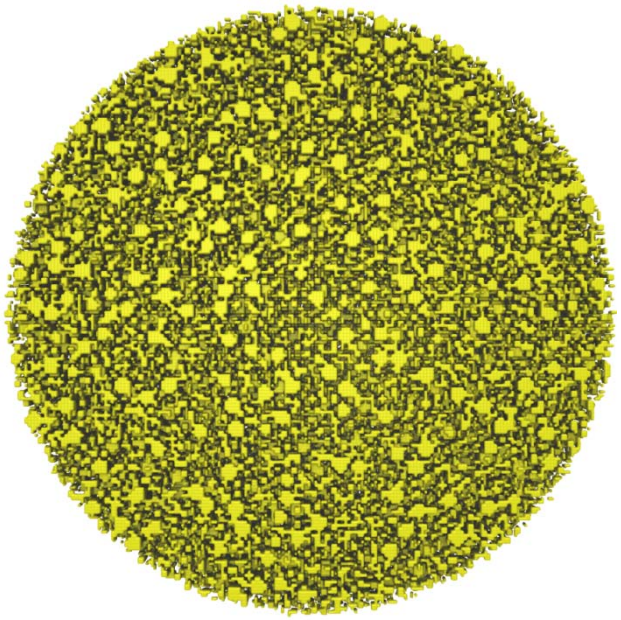
Lattice Cell Simulations

QuickSurf Representation of Lattice Cell Models



Continuous particle based model – often 70 to 300 million particles





Discretized lattice models derived from continuous model shown in VMD QuickSurf representation

Lattice Microbes: High-performance stochastic simulation method for the reaction-diffusion master equation

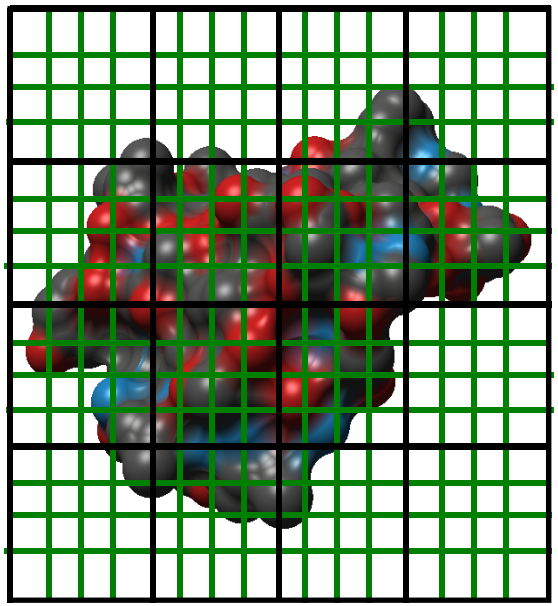
E. Roberts, J. E. Stone, and Z. Luthey-Schulten.
J. Computational Chemistry 34 (3), 245-255, 2013.

QuickSurf Algorithm Overview

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

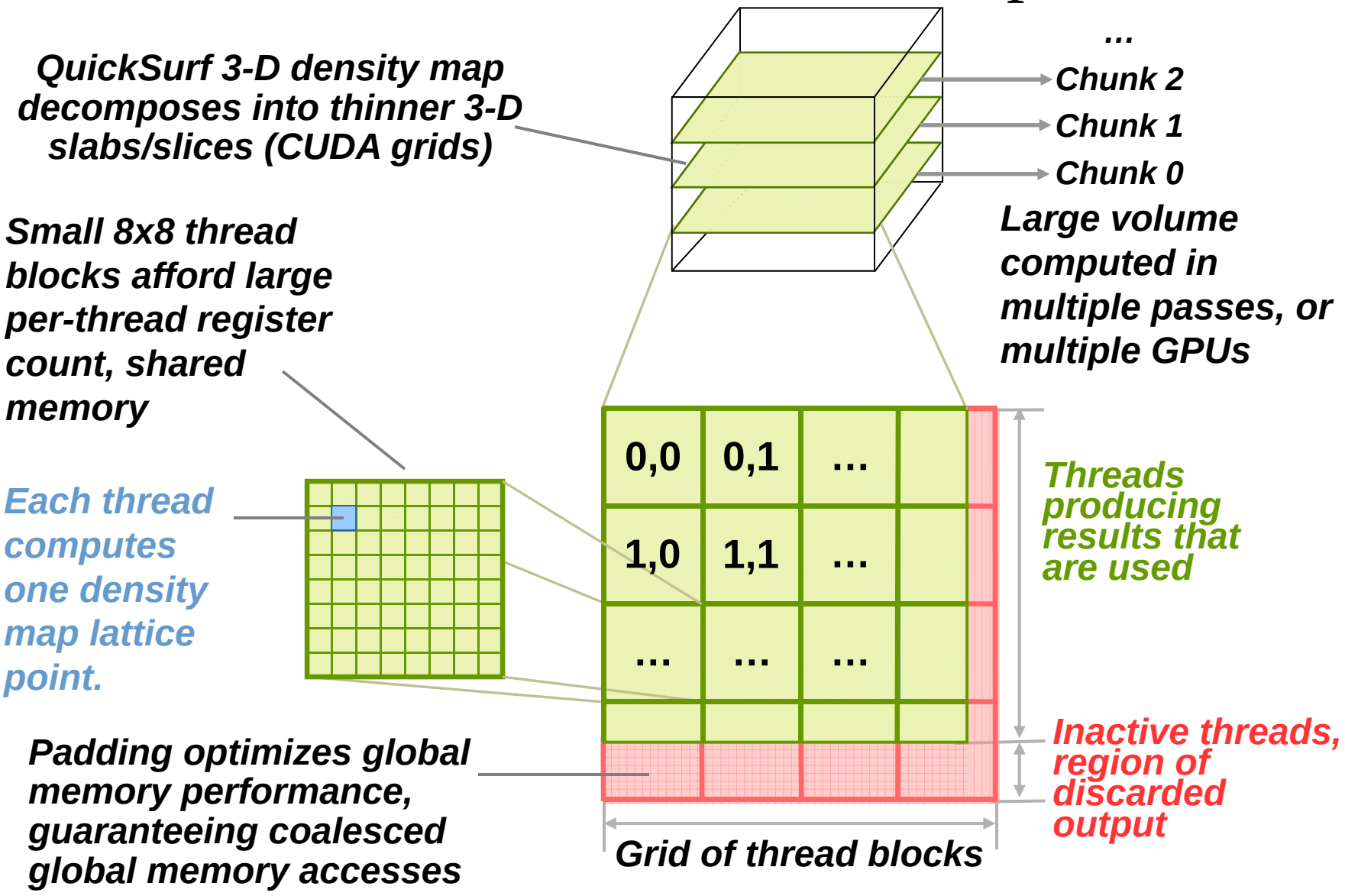
$$\rho(\vec{r}; \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \sum_{i=1}^{N} e^{\frac{-|\vec{r}-\vec{r}_i|^2}{2\alpha^2}}$$

- Extract isosurface for a user-defined density value



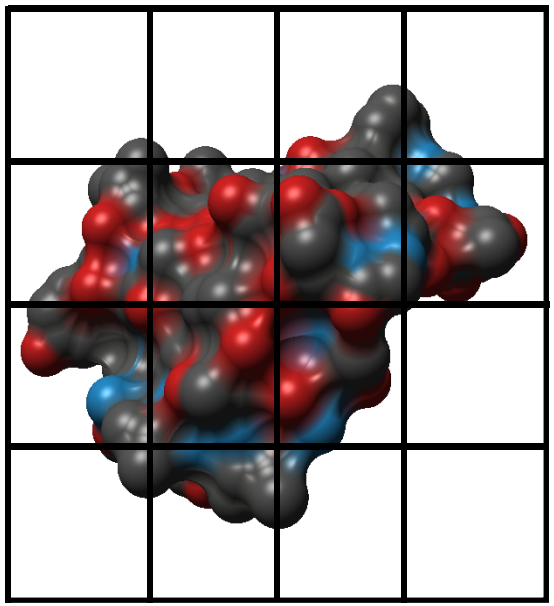
3-D density map lattice and extracted surface

QuickSurf GPU Parallel Decomposition



QuickSurf Particle Sorting, Bead Generation, Spatial Hashing

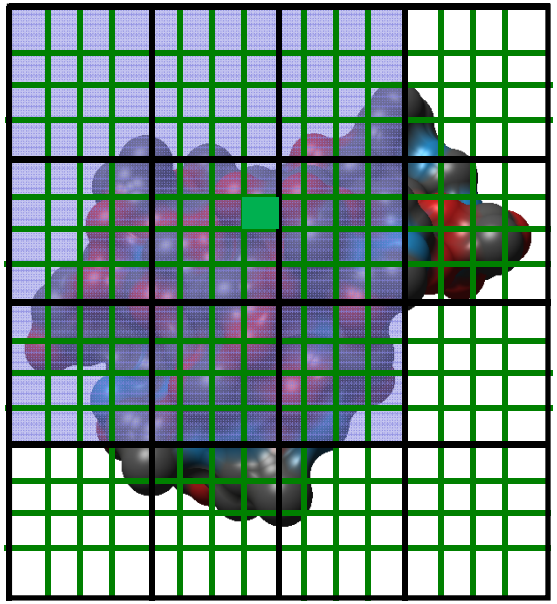
- Particles sorted into spatial acceleration grid:
  - Selected atoms or residue “beads” converted lattice coordinate system
  - Each particle/bead assigned cell index, sorted w/NVIDIA Thrust template library
- Complication:
  - Thrust allocates GPU mem. on-demand, no recourse if insufficient memory, have to re-gen QuickSurf data structures if caught by surprise!
- Workaround:
  - Pre-allocate guesstimate workspace for Thrust
  - Free the Thrust workspace right before use
  - Newest Thrust allows user-defined allocator code…



Coarse resolution spatial acceleration grid

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

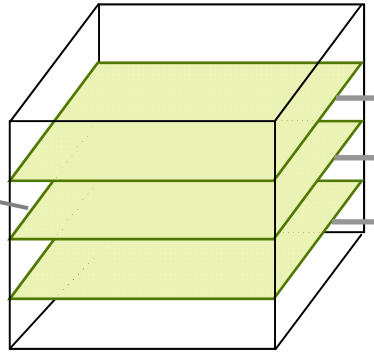


3-D density map lattice point and the neighboring spatial acceleration cells it references

QuickSurf Marching Cubes Isosurface Extraction

- Isosurface is extracted from each density map “chunk”, and either copied back to the host, or rendered directly out of GPU global memory via CUDA/OpenGL interop
- All MC memory buffers are pre-allocated to prevent significant overhead when animating a simulation trajectory

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



…

Chunk 2

Chunk 1

Chunk 0

Large volume computed in multiple passes

QuickSurf Marching Cubes Isosurface Extraction

- Our optimized MC implementation computes per-vertex surface normals, colors, and outperforms the NVIDIA SDK sample by a fair margin on Fermi GPUs
- Complications:
  - Even on a 6GB Quadro 7000, GPU global memory is under great strain when working with large molecular complexes, e.g. viruses
  - Marching cubes involves a parallel prefix sum (scan) to compute target indices for writing resulting vertices
  - We use Thrust for scan, has the same memory allocation issue mentioned earlier for the sort, so we use the same workaround
  - The number of output vertices can be huge, but we rarely have sufficient GPU memory for this – we use a fixed size vertex output buffer and hope our buffer size heuristics don't fail us

QuickSurf Performance
GeForce GTX 580

Molecular system	Atoms	Resolution	T_{sort}	$T_{density}$	T_{MC}	# vertices	FPS
MscL	111,016	1.0Å	0.005	0.023	0.003	0.7 M	28
STMV capsid	147,976	1.0Å	0.007	0.048	0.009	2.4 M	13.2
Poliovirus capsid	754,200	1.0Å	0.01	0.18	0.05	9.2 M	3.5
STMV w/ water	955,225	1.0Å	0.008	0.189	0.012	2.3 M	4.2
Membrane	2.37 M	2.0Å	0.03	0.17	0.016	5.9 M	3.9
Chromatophore	9.62 M	2.0Å	0.16	0.023	0.06	11.5 M	3.4
Membrane w/ water	22.77 M	4.0Å	4.4	0.68	0.01	1.9 M	0.18

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

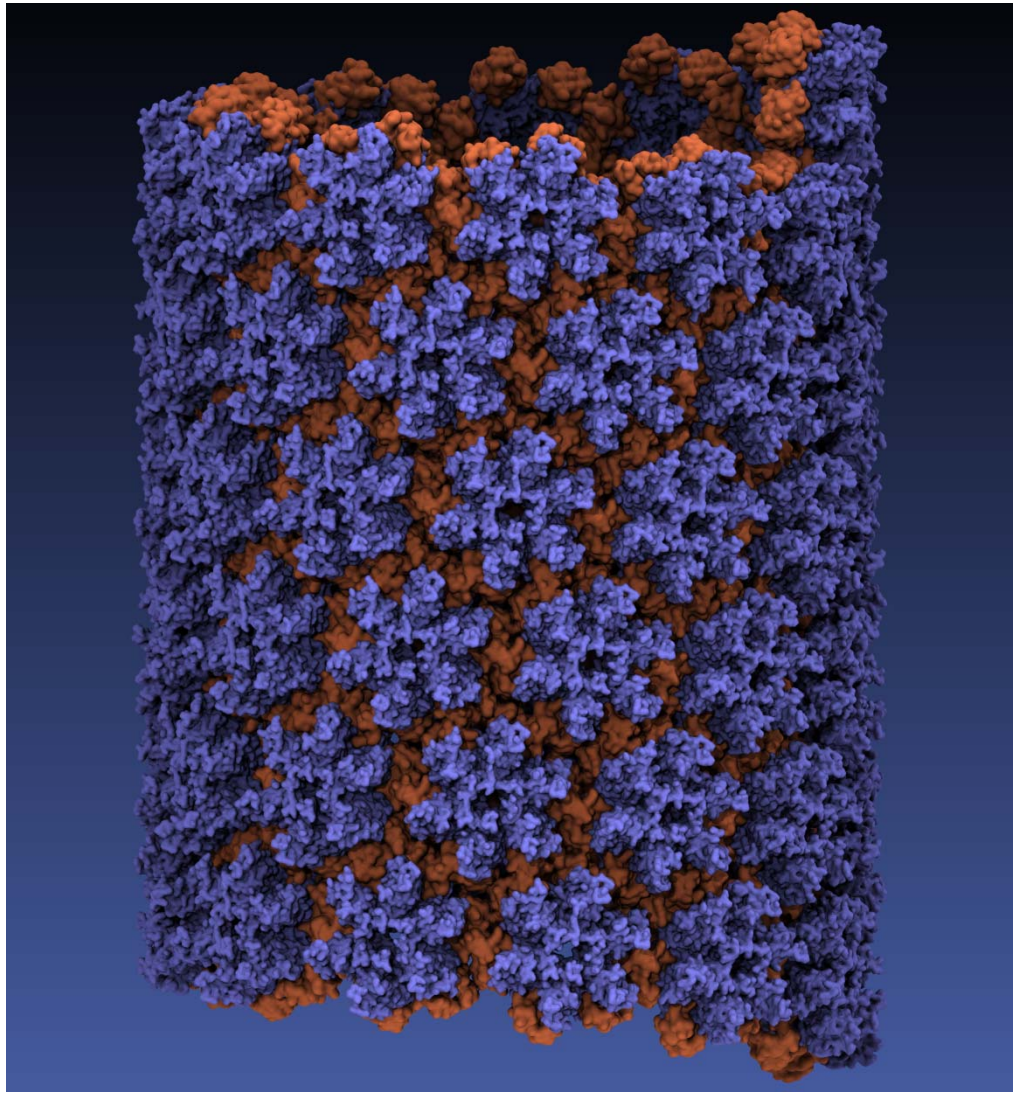
M. Krone, J. E. Stone, T. Ertl, K. Schulten. *EuroVis Short Papers*, pp. 67-71, 2012

Extensions and Analysis Uses for QuickSurf Triangle Mesh

- Curved PN triangles:
  - We have performed tests with post-processing the resulting triangle mesh and using curved PN triangles to generate smooth surfaces with a larger grid spacing, for increased performance
  - Initial results demonstrate some potential, but there can be pathological cases where MC generates long skinny triangles, causing unsightly surface creases
- Analysis uses (beyond visualization):
  - Minor modifications to the density map algorithm allow rapid computation of solvent accessible surface area by summing the areas in the resulting triangle mesh
  - Modifications to the density map algorithm will allow it to be used for MDFF (molecular dynamics flexible fitting)
  - Surface triangle mesh can be used as the input for computing the electrostatic potential field for mesh-based algorithms

Challenge: Support Interactive QuickSurf for Large Structures on Mid-Range GPUs

- Structures such as HIV initially needed large (6GB) GPU memory to generate fully-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 performance/memory capacity cases



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

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

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

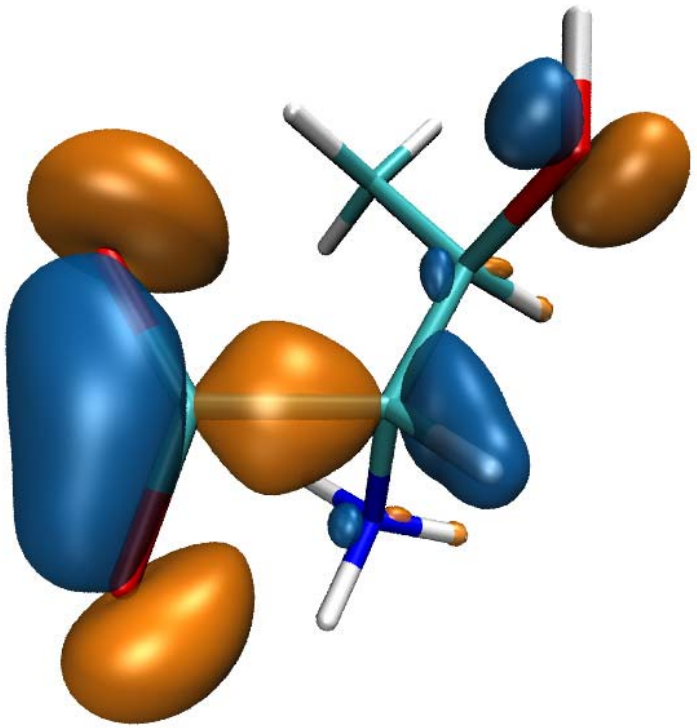
```
template<class DENSITY, class VOLTEX>
__global__ static void
gaussdensity_fast_tex_norm( …  ) {

  … 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

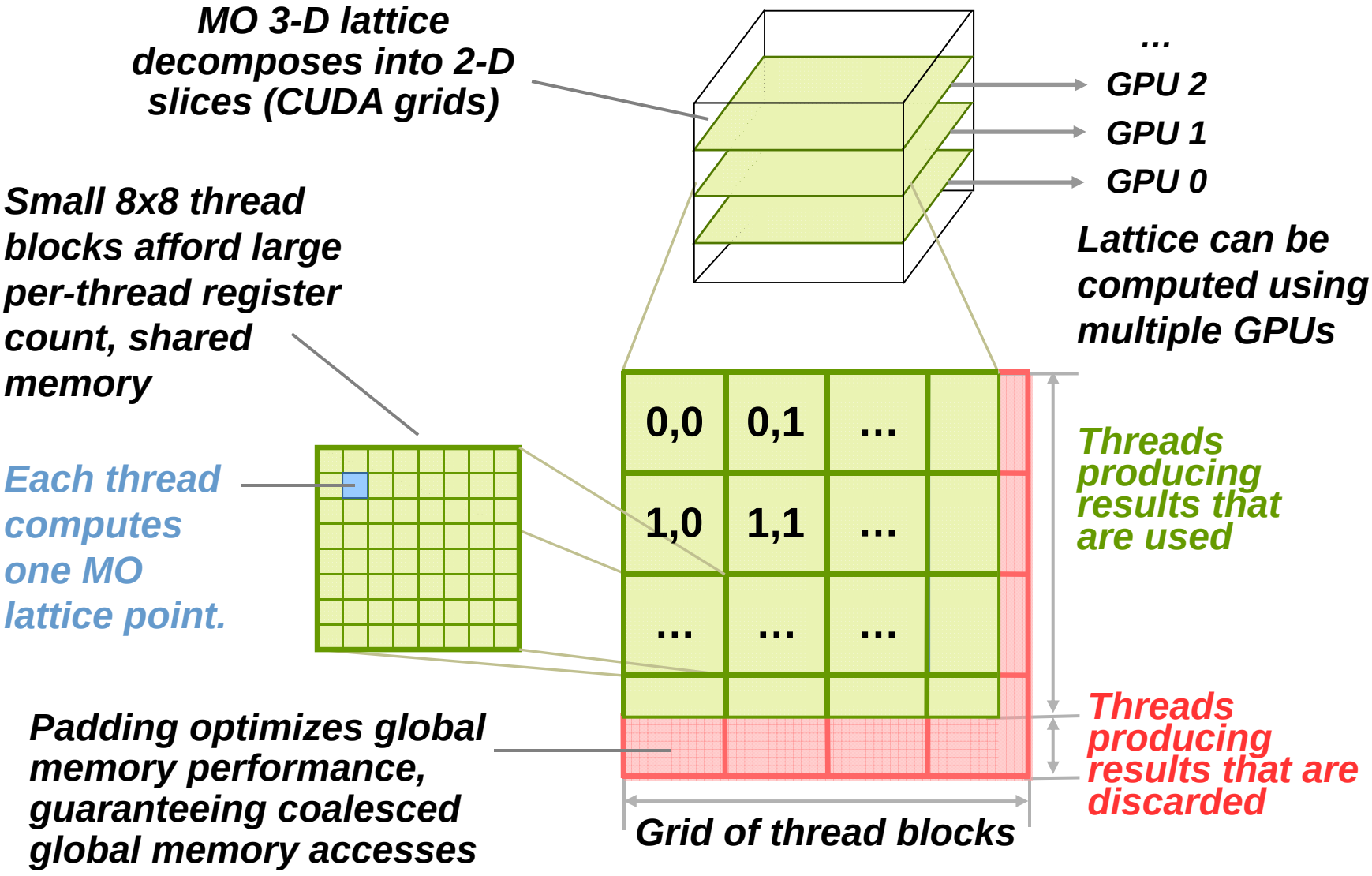
- **Halved** overall GPU memory use
- Achieved **1.5x to 2x performance gain**:
  - The “gather” density map algorithm keeps type conversion operations out of the innermost loop
  - Density map global memory writes reduced to half
  - Multiple stages of Marching Cubes operate on smaller input and output 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!

Computing Molecular Orbitals

- Visualization of MOs aids in understanding the chemistry of molecular system
- Calculation of high resolution MO grids for display can require tens to hundreds of seconds on multi-core CPUs, even with the use of hand-coded SSE



MO GPU Parallel Decomposition



VMD MO GPU Kernel Snippet:
Loading Tiles Into Shared Memory On-Demand

```
[… outer loop over atoms …]
  if ((prim_counter + (maxprim<<1)) >= SHAREDSIZE) {
    prim_counter += sblock_prim_counter;
    sblock_prim_counter = prim_counter & MEMCOAMASK;
    s_basis_array[sidx      ] = basis_array[sblock_prim_counter + sidx      ];
    s_basis_array[sidx +  64] = basis_array[sblock_prim_counter + sidx +  64];
    s_basis_array[sidx + 128] = basis_array[sblock_prim_counter + sidx + 128];
    s_basis_array[sidx + 192] = basis_array[sblock_prim_counter + sidx + 192];
    prim_counter -= sblock_prim_counter;
    __syncthreads();
  }
  for (prim=0; prim < maxprim;  prim++) {
    float exponent       = s_basis_array[prim_counter    ];
    float contract_coeff = s_basis_array[prim_counter + 1];
    contracted_gto += contract_coeff * __expf(-exponent*dist2);
    prim_counter += 2;
  }
[… continue on to angular momenta loop …]
```

Shared memory tiles:

- Tiles are checked and loaded, if necessary, immediately prior to entering key arithmetic loops
- Adds additional control overhead to loops, even with optimized implementation

VMD MO GPU Kernel Snippet:

Fermi/Kepler kernel based on caching approach

```
[… outer loop over atoms …]
  // loop over the shells/basis funcs belonging to this atom
  for (shell=0; shell < maxshell; shell++) {
    float contracted_gto = 0.0f;
    int maxprim    = shellinfo[(shell_counter<<4)      ];
    int shell_type = shellinfo[(shell_counter<<4) + 1];
    for (prim=0; prim < maxprim; 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;
   }
   […continue on to angular momenta loop …]
```

Hardware cache:

- Simplifies code!
- Reduces control overhead
- Gracefully handles arbitrary-sized problems
- Matches performance of constant memory on Fermi, slower on Kepler GK104, GK110
- Working on new kernel variants that are better-suited to Kepler

VMD Single-GPU Molecular Orbital Performance Results for C_{60} on Fermi

Intel X5550 CPU, GeForce GTX 480 GPU

Kernel	Cores/GPUs	Runtime (s)	Speedup
Xeon 5550 ICC-SSE	1	30.64	1.0
Xeon 5550 ICC-SSE	8	4.13	7.4
CUDA shared mem	1	0.37	83
CUDA L1-cache (16KB)	**1**	**0.27**	**113**
CUDA const-cache	1	0.26	117
CUDA const-cache, zero-copy	1	0.25	122

Fermi GPUs have caches: match perf. of hand-coded shared memory kernels. Zero-copy memory transfers improve overlap of computation and host-GPU I/Os.

Preliminary Single-GPU Molecular Orbital Performance Results for C_{60} on Kepler

Intel X5550 CPU, GeForce GTX 680 GPU

Kernel	Cores/GPUs	Runtime (s)	Speedup
Xeon 5550 ICC-SSE	1	30.64	1.0
Xeon 5550 ICC-SSE	8	4.13	7.4
CUDA shared mem	1	0.264	116
CUDA L1-cache (16KB)	**1**	**0.228**	**134**
CUDA const-cache	1	0.104	292
CUDA const-cache, zero-copy	1	0.0938	326

Kepler GeForce 680+Tesla K20 prefer the constant cache kernels vs. the others. Cache latencies have made the constant memory kernel the winner (**so far**) on Kepler

Optimizing Molecular Orbital Visualization Performance

- End-to-end visualization speed depends on more than MO cartesian grid computation speed:
  - Speed of isosurface extraction (Marching Cubes)
  - Preparation and rendering of OpenGL vertex arrays
- Latest GPU generations require GPU-accelerated Marching Cubes to achieve peak performance
- Exploited the new multi-precision Marching Cubes approach developed for QuickSurf:
  - fully-GPU-native MO code path
  - Reduces host-device memory bandwidth by eliminating copies and using compact normal and color arrays
- End-to-end visualization performance gain ranges from **2x to 3x faster** for C_{60} test cases

Acknowledgements

- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- NCSA Blue Waters Team
- NCSA Innovative Systems Lab
- NVIDIA CUDA Center of Excellence, University of Illinois at Urbana-Champaign
- The CUDA team at NVIDIA
- NIH support: P41-RR005969

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.