S7382—GPUs Unleashed: Analysis of Petascale Molecular Dynamics Simulations with VMD

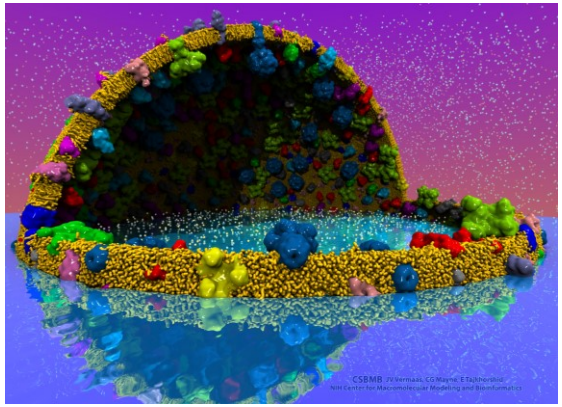
John E. Stone
Theoretical and Computational Biophysics Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

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

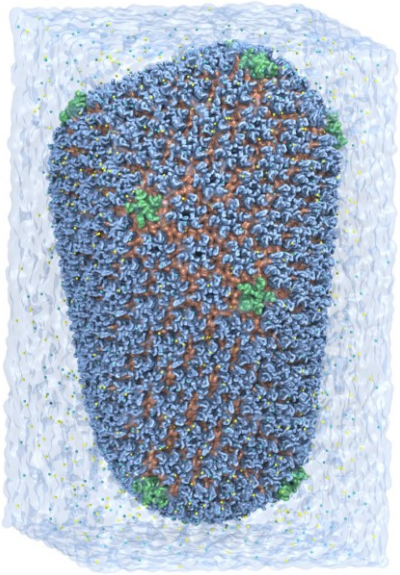
S7382, GPU Technology Conference
13:00-13:50, Room 211B, San Jose Convention Center,
San Jose, CA, Wednesday May 10th, 2017

VMD – “Visual Molecular Dynamics”

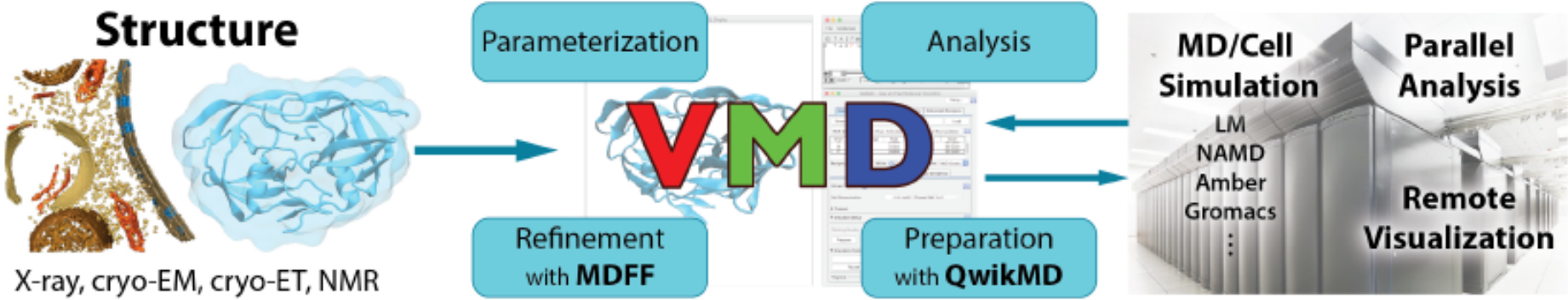
- Visualization and analysis of:
  - Molecular dynamics simulations
  - Lattice cell simulations
  - Quantum chemistry calculations
  - Sequence information
- User extensible scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/



Cell-Scale Modeling

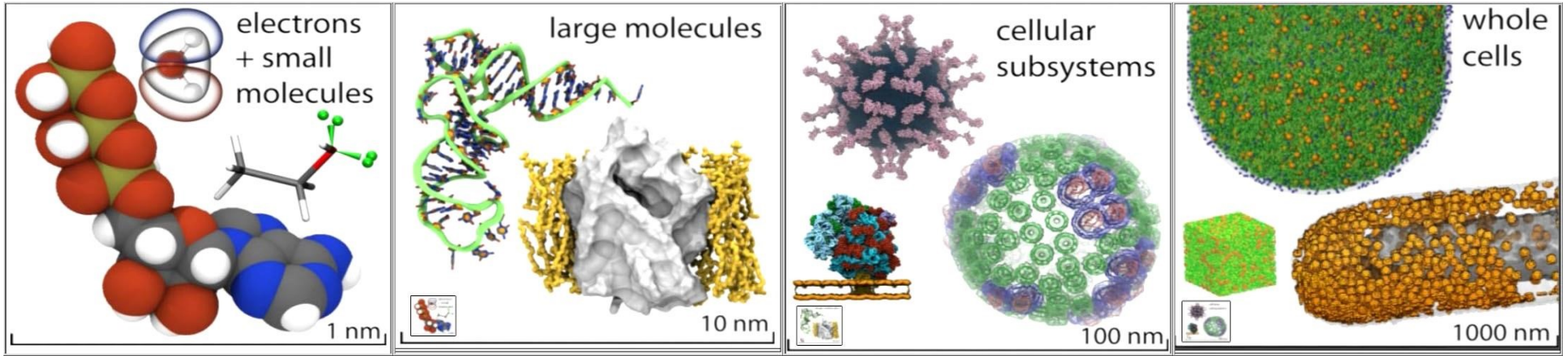


MD Simulation



VMD Interoperability Serves Many Communities

- Uniquely interoperable with a broad range of tools:
  - AMBER, CHARMM, CPMD, DL_POLY, GAMESS, GROMACS, HOOMD, LAMMPS, NAMD, and many more …
- Supports key data types, file formats, and databases
- Incorporates tools for simulation preparation, visualization, and analysis



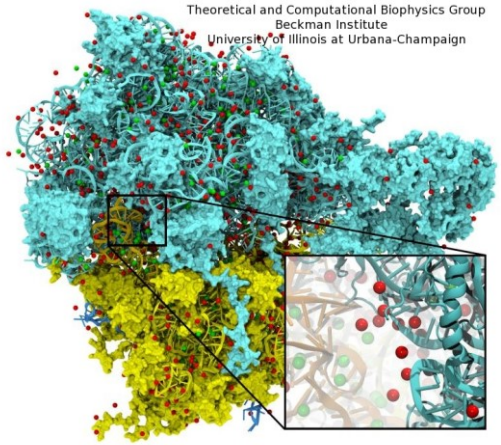
10 Years of GPU Computing in VMD

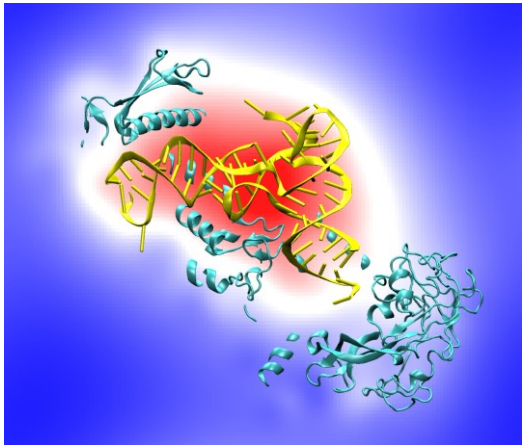
- Has stood the test of time
- Modeling, Visualization, Rendering, and Analysis

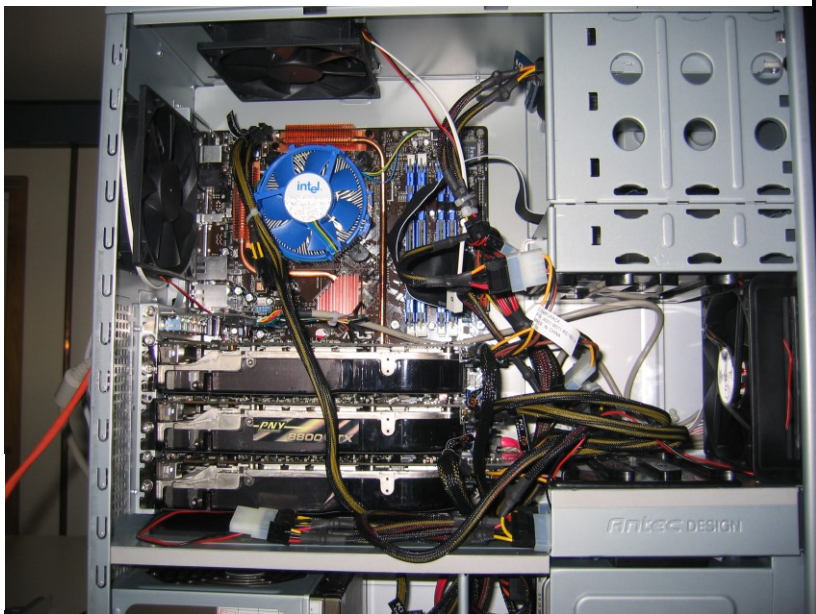
Blast from the past:

CUDA starting with version 0.7 !!!

Quad core Intel QX6700, three NVIDIA GeForce 8800GTX GPUs, RHEL4 Linux





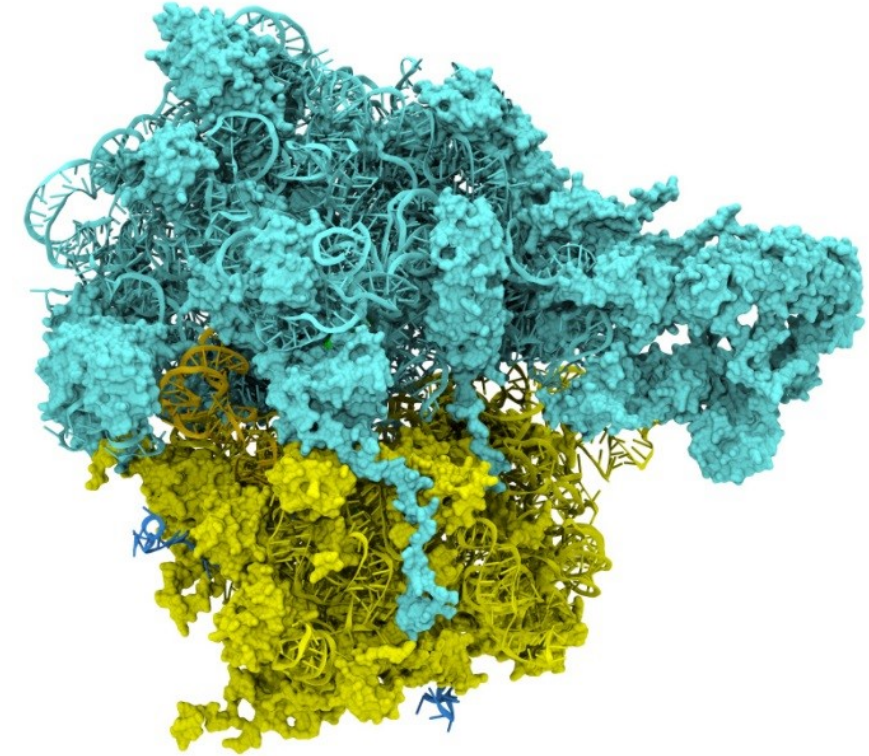


Accelerating molecular modeling applications with graphics processors. J. Stone, J. Phillips, P. Freddolino, D. Hardy, L. Trabuco, K. Schulten. *J. Comp. Chem.*, 28:2618-2640, 2007.

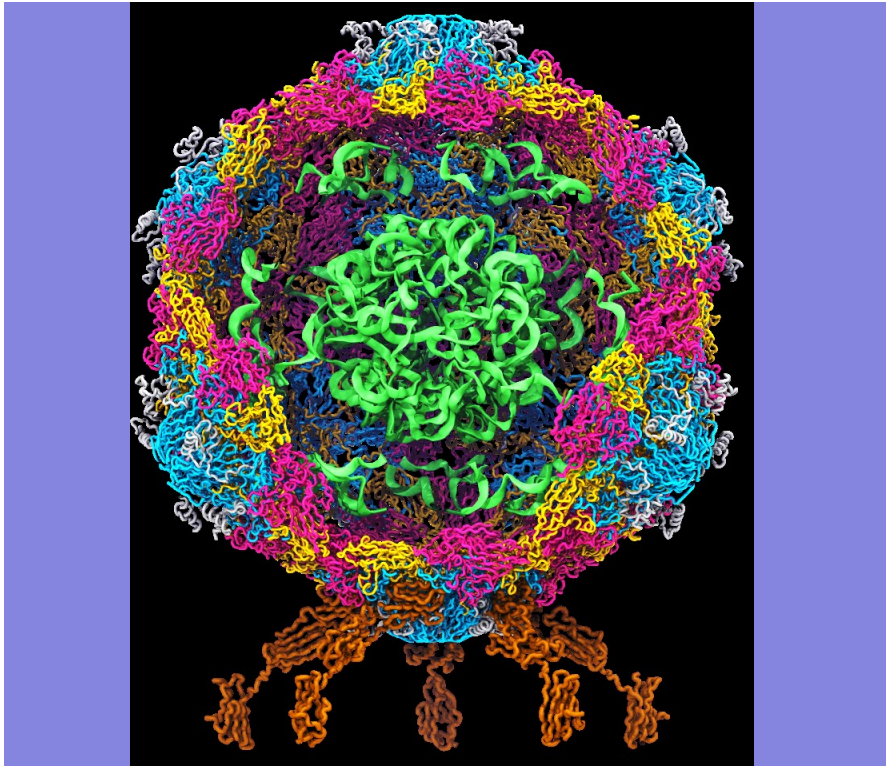
Goal: A Computational Microscope

Study the molecular machines in living cells

Ribosome: target for antibiotics



Poliovirus



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
- Supports GPU-accelerated Cray XK7 nodes for both visualization and analysis:
  - **GPU accelerated trajectory analysis w/ CUDA**
  - **OpenGL and GPU ray tracing for visualization and movie rendering**
- Parallel I/O rates up to **275 GB/sec** on 8192 Cray XE6 nodes – can read in **231 TB in 15 minutes!**

Parallel VMD currently available on:

ORNL Titan, NCSA Blue Waters, Indiana Big Red II, CSCS Piz Daint, and similar systems

NCSA Blue Waters Hybrid Cray XE6 / XK7
22,640 XE6 dual-Opteron CPU nodes
4,224 XK7 nodes w/ Telsa K20X GPUs

VMD on IBM OpenPOWER

- VMD has been running on various POWER hardware since 1998!
- Now runs on POWER8 w/ Linux in little-endian mode:
  - VMD 1.9.3 release on Nov 2016 includes first OpenPOWER release
  - Src supports CUDA 7.x [1], and **CUDA 8.x w/ P100 and NVLink**
  - Uses P8 VSX instructions for hand-coded and vectorized kernels [1]
- In-progress VMD 1.9.4 development:
  - **VMD supports full OpenGL via GLX and EGL on POWER now!**
    - Latest public NVIDIA driver version 375.66
  - Ongoing improvements to CPU/GPU work scheduling, **NUMA** optimality, and use of new **NVLink** interconnect on latest **IBM "Minsky"** hardware

[1] Early Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms. J. E. Stone, A.-P. Hynninen, J. C. Phillips, K. Schulten. International Workshop on OpenPOWER for HPC (IWOPH'16), LNCS 9945, pp. 188-206, 2016.

Ongoing VMD CUDA Work on POWER8

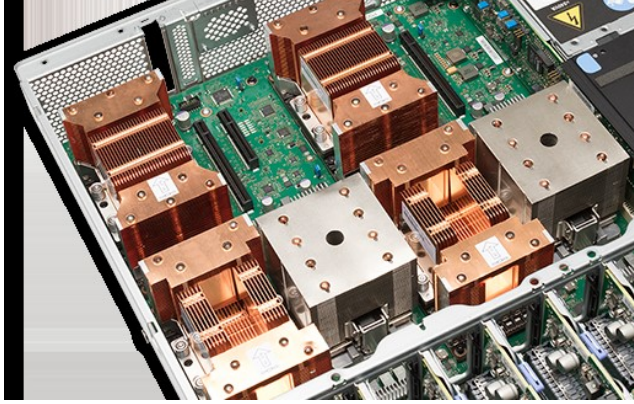
- CUDA kernels all run correctly on previous-gen PCIe Tesla K40s, and new Tesla P100 w/ NVLink
- Early observations about P8+CUDA+NVLink so far:
  - P8 single-thread perf more of an issue than on x86 for small untuned parts of existing code
  - **P8+CUDA NUMA-correctness w/ NVLink much more important** than PCIe (e.g. x86) due to **larger benefits/penalties** when NVLink is used effectively vs. not
  - **P8 "Minsky"** systems get extra benefits for algorithms that have lots of host-GPU DMA transfers, where the NVLink interconnect speeds greatly outpeform PCIe

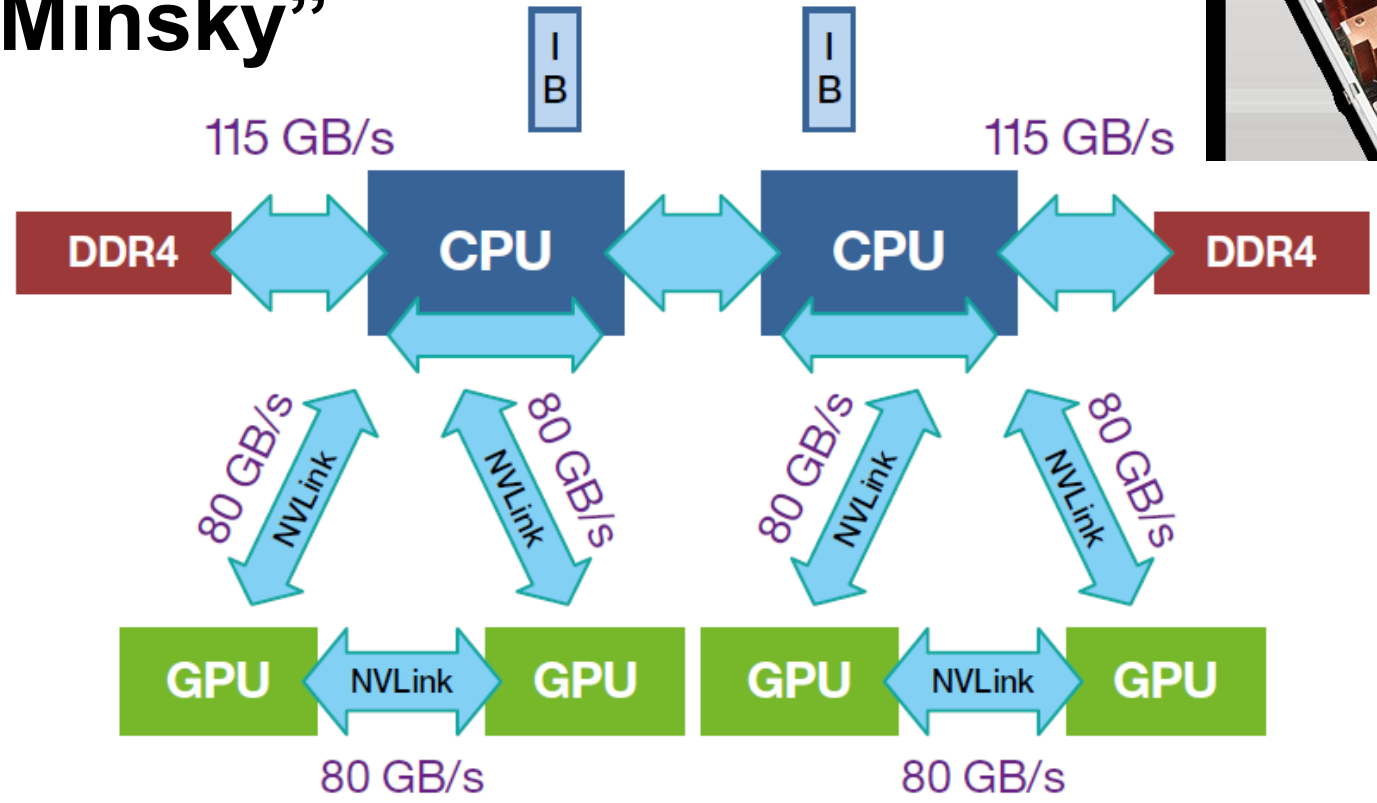
Benefits of P8+NVLink for VMD

- Rapid access to host-side data too large to fit entirely in P100 GPU memory
  - Many existing VMD CUDA kernels already used this strategy w/ PCIe, performance gains from NVLink are large and immediate
- Rapid peer-to-peer GPU data transfers:
  - Bypass host whenever possible, perform nearest-neighbor exchanges for pairwise calculations, e.g. those that arise in algorithms for simulation trajectory clustering
  - Use aggregate GPU memory to collectively store/cache large data – well suited for high-fidelity ray tracing of scenes containing massive amounts of geometry

IBM S822LC w/ NVLink

“Minsky”

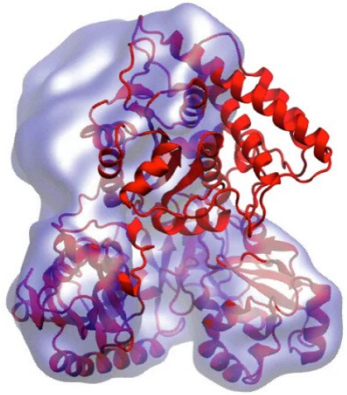




Molecular Dynamics Flexible Fitting (MDFF)

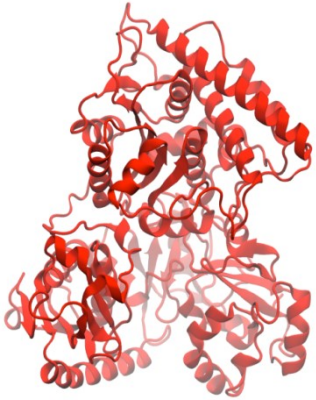
X-ray crystallography → MDFF ← Electron microscopy

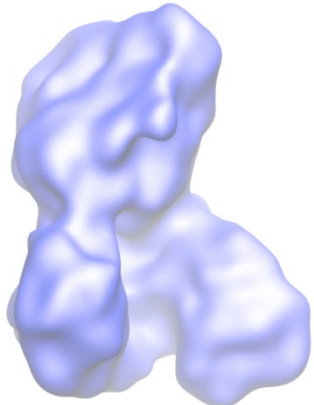
APS at Argonne



FEI microscope

ORNL Titan





Molecular dynamics-based refinement and validation for sub-5Å cryo-electron microscopy maps. A. Singharoy, I. Teo, R. McGreevy, J. E. Stone, J. Zhao, and K. Schulten. eLife 2016;10.7554/eLife.16105

Molecular Dynamics Flexible Fitting - Theory

Two terms are added to the MD potential

$$U_{total} = U_{MD} + U_{EM} + U_{SS}$$

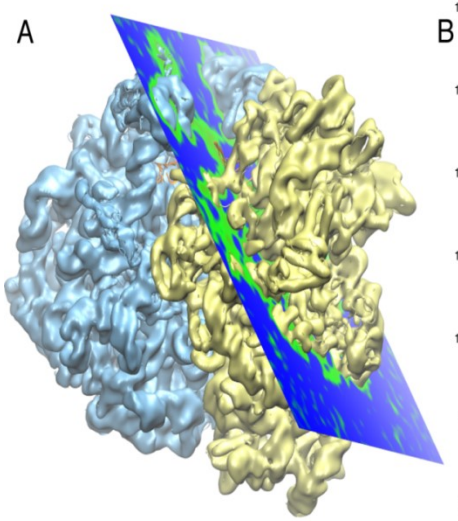
An external potential derived from the EM map is defined on a grid as

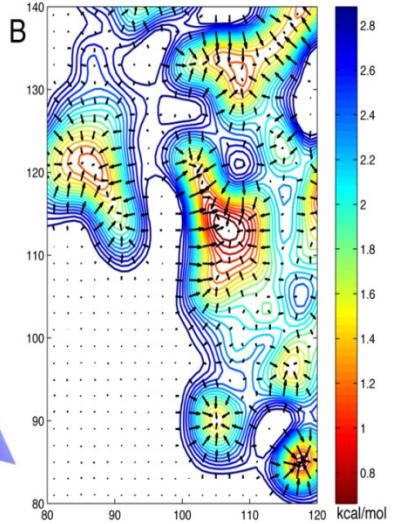
$$U_{EM}(\mathbf{R}) = \sum_j w_j V_{EM}(\mathbf{r}_j)$$

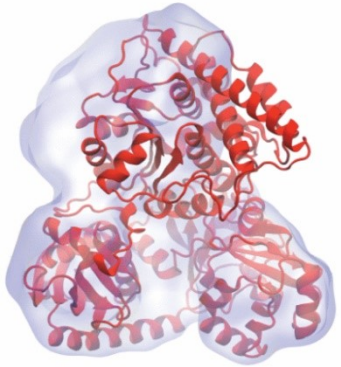
$$V_{EM}(\mathbf{r}) = \begin{cases} \xi \left(1 - \frac{\Phi(\mathbf{r}) - \Phi_{thr}}{\Phi_{max} - \Phi_{thr}}\right) & \text{if } \Phi(\mathbf{r}) \geq \Phi_{thr}, \\ \xi & \text{if } \Phi(\mathbf{r}) < \Phi_{thr}. \end{cases}$$

A mass-weighted force is then applied to each atom

$$\mathbf{f}_i^{EM} = -\nabla U_{EM}(\mathbf{R}) = -w_i \partial V_{EM}(\mathbf{r}_i)/\partial r_i$$

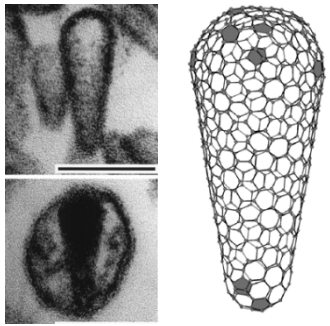






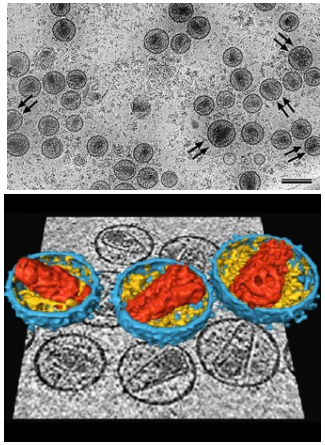
Structural Route to the all-atom HIV-1 Capsid

1st TEM (1999)



Ganser et al. *Science*, 1999

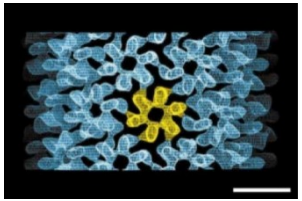
1st tomography (2003)



Briggs et al. *EMBO J*, 2003

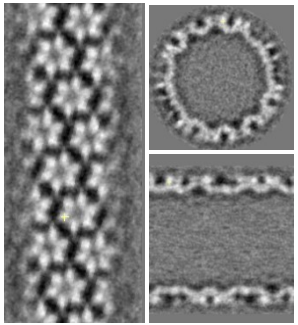
Briggs et al. *Structure*, 2006

hexameric tubule



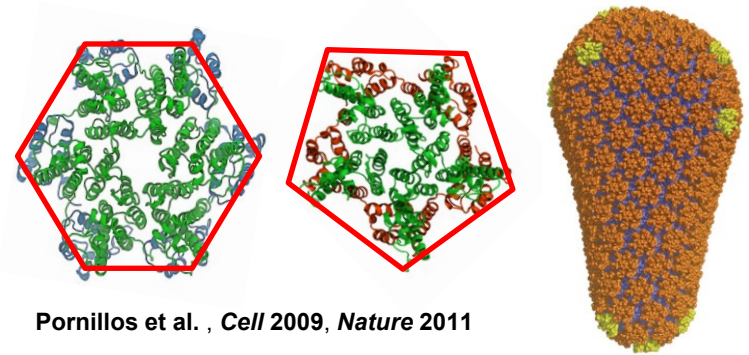
Li et al., *Nature*, 2000

cryo-ET (2006)



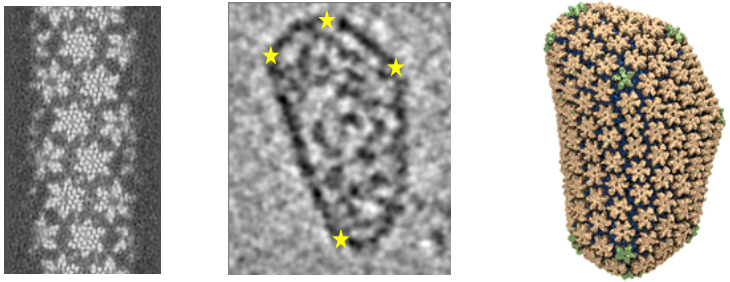
Byeon et al., *Cell* 2009

Crystal structures of separated hexamer and pentamer



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

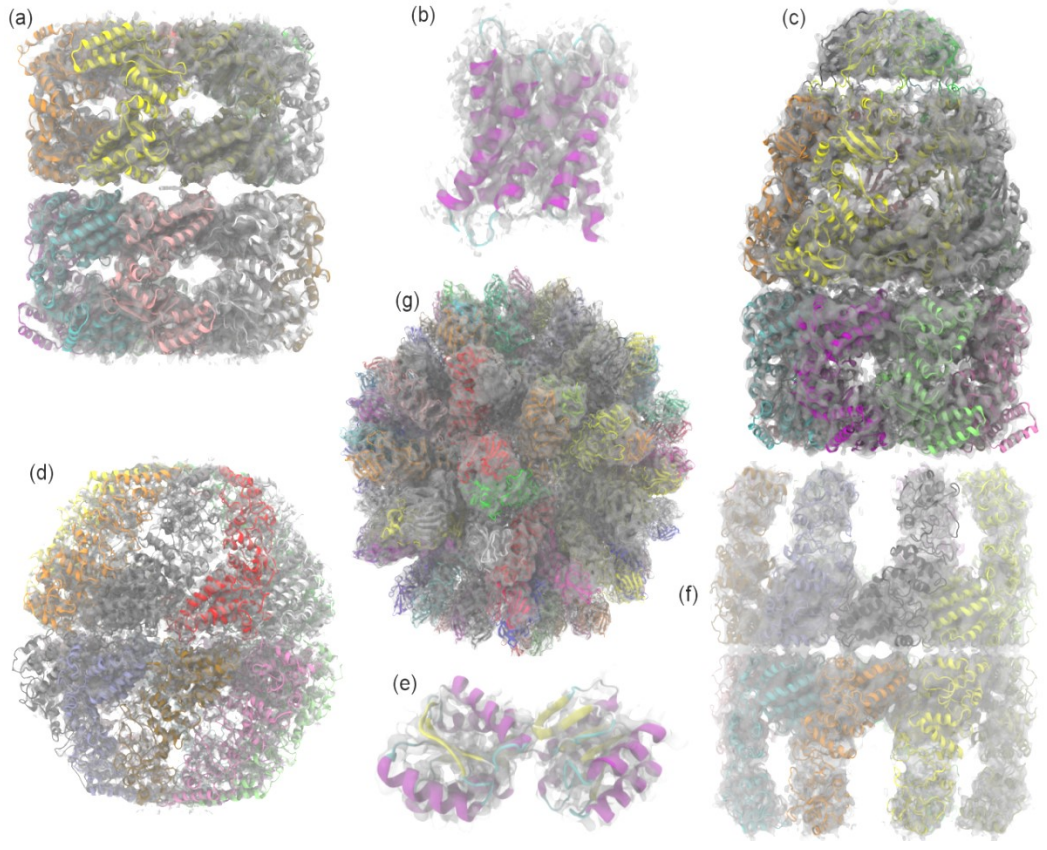
High res. EM of hexameric tubule, tomography of capsid,
all-atom model of capsid by MDFF w/ NAMD & VMD,
NSF/NCSA Blue Waters computer at Illinois



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

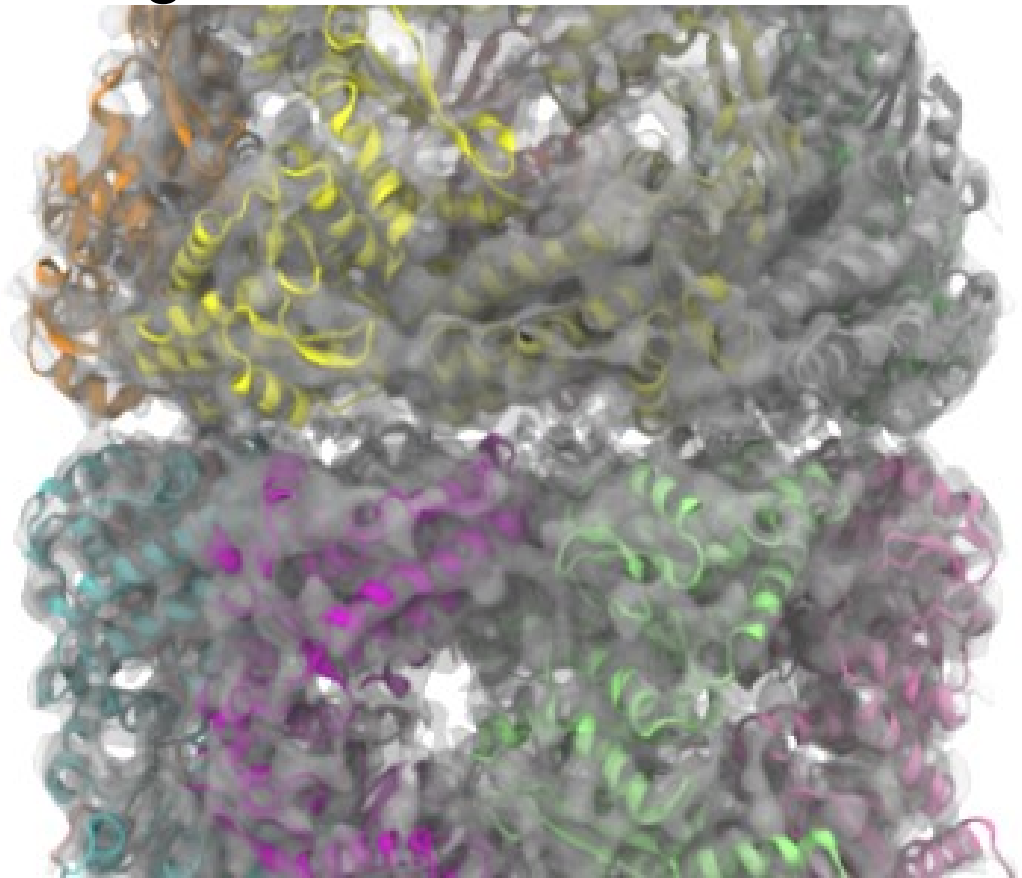
Evaluating Quality-of-Fit for Structures Solved by Hybrid Fitting Methods

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**.



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

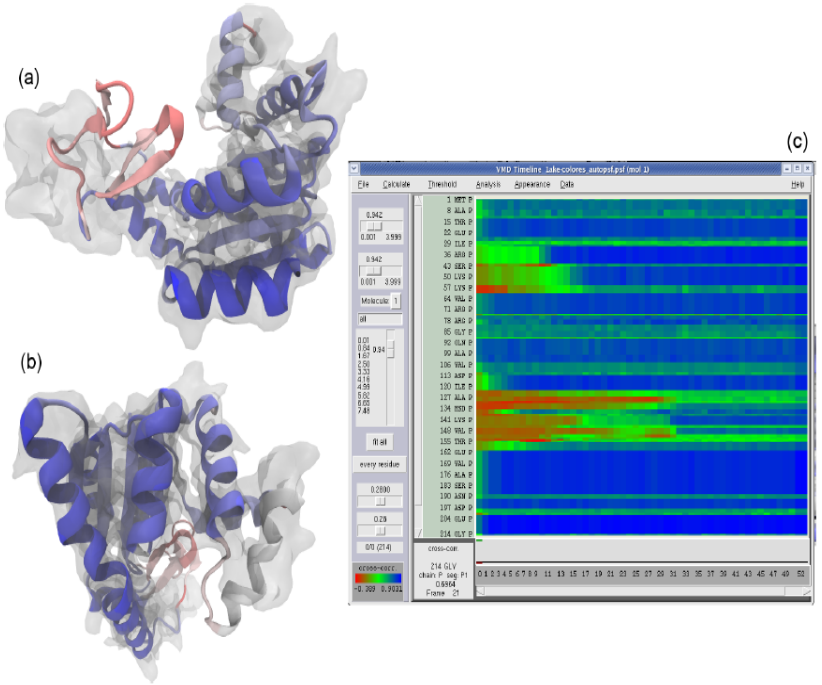
Compute Pearson correlation to evaluate quality-of-fit between a reference **cryo-EM density map** and a **simulated density map** produced from an **all-atom structure**.



GPUs Can Reduce MDFF 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 that were previously impractical, allowing detailed study of very large structures such as viruses



GPU-accelerated MDFF Cross Correlation Timeline

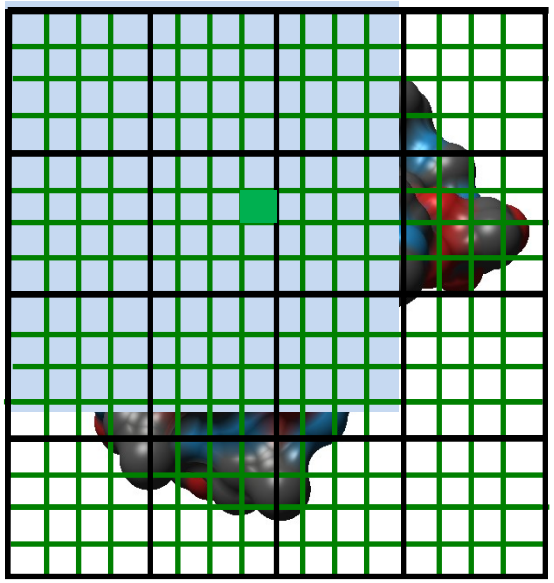
Regions with poor fit **Regions with good fit**

MDFF Density Map Algorithm

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

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

- Truncated Gaussian and spatial acceleration grid ensure linear time-complexity



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

Single-Pass MDFF GPU Cross-Correlation

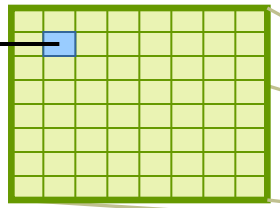
3-D density map decomposes into 3-D grid of 8x8x8 tiles containing CC partial sums and local CC values

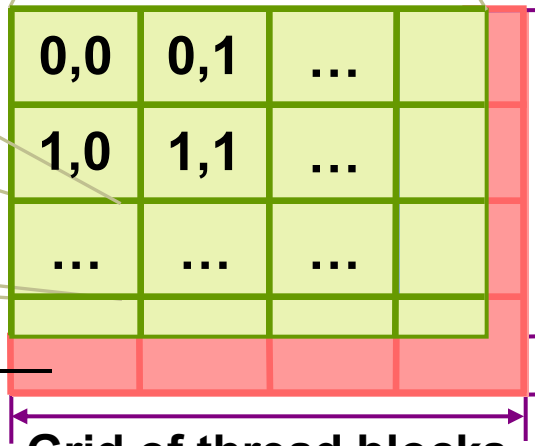
Fusion of density and CC calculations into a single CUDA kernel!!!

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

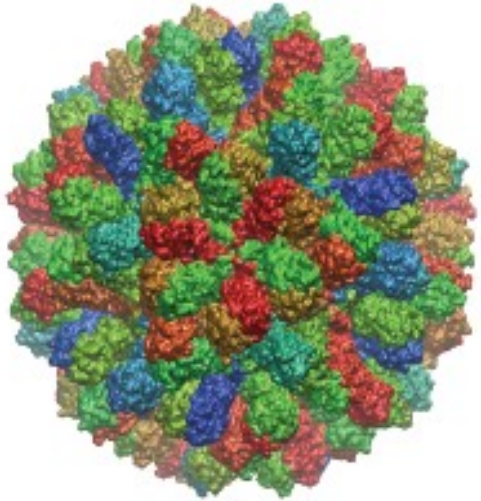




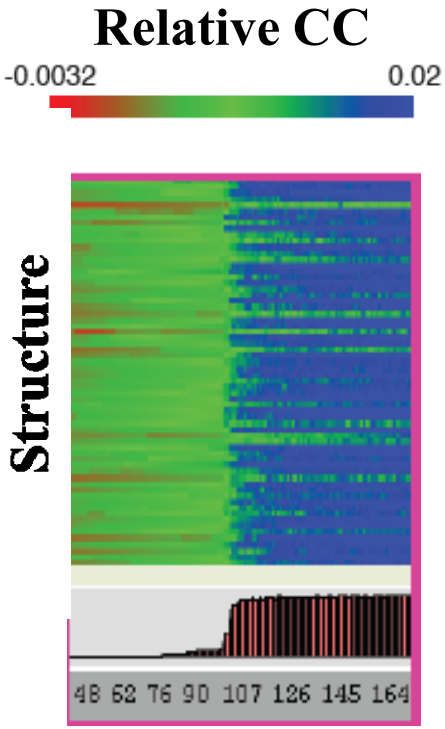
Padding optimizes global memory performance, guaranteeing coalesced global memory accesses

Parallel MDFF Cross Correlation Analysis on Cray XK7

Rabbit Hemorrhagic Disease Virus (RHDV)	
Traj. frames	10,000
Structure component selections	720
Single-node XK7 (projected)	336 hours (14 days)
128-node XK7	3.2 hours 105x speedup
2048-node XK7	19.5 minutes 1035x speedup



RHDV colored by relative CC



Calculation of 7M CCs would take **5 years** using serial CPU algorithm!

Stone et al., Faraday Discuss., 169:265-283, 2014.

VMD Tesla P100 Cross Correlation Performance

Rabbit Hemorrhagic Disease Virus: 702K atoms, 6.5Å resolution

P100 Die-Stacked Mem Accelerates Bandwidth Intensive Calculation

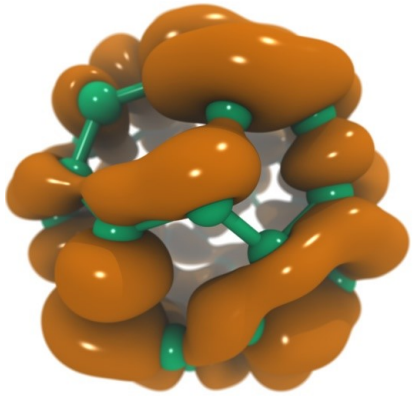
Hardware platform	Runtime,	Speedup vs. Chimera,	VMD+GPU
Chimera Xeon E5-2687W (2 socket) [1]	15.860s,	1x	
VMD-CPU IBM Power8 VSX SIMD (2 socket) [2]	1.334s,	12x	
VMD-CPU Intel Xeon E5-2660v3 SIMD (2 socket) [2]	0.905s,	17x	
VMD-CUDA IBM Power8 + 1x Tesla K40 [2]	0.488s,	32x	**0.9x**
VMD-CUDA Intel Xeon E5-2687W + 1x Quadro K6000 [1,2]	0.458s,	35x	**1.0x**
VMD-CUDA Intel Xeon E5-2698v3 + 1x Tesla P100	**0.090s,**	176x	**5.1x**
VMD-CUDA IBM Power8 "Minsky" + 1x Tesla P100	**0.080s,**	198x	**5.7x**

[1] GPU-Accelerated Analysis and Visualization of Large Structures Solved by Molecular Dynamics Flexible Fitting. J. E. Stone, R. McGreevy, B. Isralewitz, and K. Schulten. Faraday Discussions 169:265-283, 2014.

[2] Early Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms. J. E. Stone, A.-P. Hynninen, J. C. Phillips, K. Schulten. International Workshop on OpenPOWER for HPC (IWOPH'16), LNCS 9945, pp. 188-206, 2016.

NIH

VMD Tesla P100 Performance for C_{60} Molecular Orbitals, 516x519x507 grid



Hardware platform	Runtime,	Speedup
IBM Power8 (2 socket) (ORNL 'crest') [1]	8.03s,	0.4x
Intel Xeon E5-2660v3 (2 socket) [1]	7.14s,	0.5x
IBM Power8 (ORNL 'crest') + 1x Tesla K40 [1]	3.49s,	**1.0x**
Intel Xeon E5-2698v3 + 1x Tesla P100	**1.35s,**	**2.5x**
IBM Power8 "Minsky" + 1x Tesla P100	**1.09s,**	**3.3x**
IBM Power8 (ORNL 'crest') + 4x Tesla K40 [1]	0.91s,	3.8x
Intel Xeon E5-2698v3 + 4x Tesla P100	**0.37s,**	**9.4x**
IBM Power8 "Minsky" + 4x Tesla P100	**0.30s,**	**11.6x**

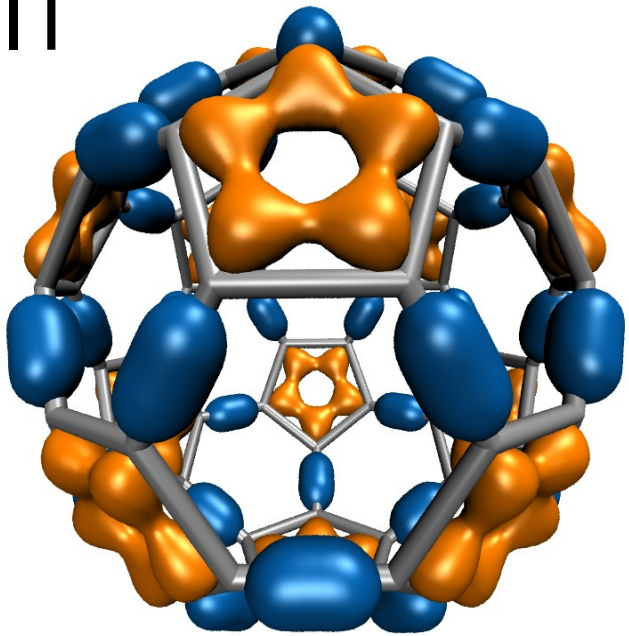
NVLink perf. boost w/ no code tuning (YET)

[1] Early Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms. J. E. Stone, A.-P. Hynninen, J. C. Phillips, K. Schulten. International Workshop on OpenPOWER for HPC (IWOPH'16), LNCS 9945, pp. 188-206, 2016.

NIH

Molecular Orbitals w/ NVRTC JIT

- Visualization of MOs aids in understanding the chemistry of molecular system
- MO spatial distribution is correlated with probability density for an electron(s)
- **Animation** of (classical mechanics) molecular dynamics trajectories provides insight into simulation results
  - To do the same for QM or QM/MM simulations MOs must be computed at **10 FPS** or more
  - **Large GPU speedups (up to 30x vs. current generation 4-core CPUs)** over existing tools makes this possible!
- **Run-time code generation** (JIT) and compilation via **CUDA NVRTC** enable further optimizations and the **highest performance to date: 1.8x faster than fully-general data-driven loops**

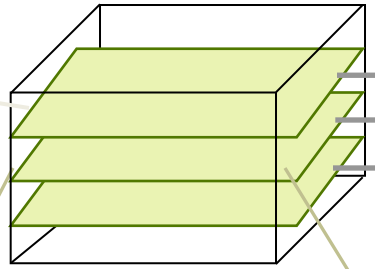


C_{60}

High Performance Computation and Interactive Display of Molecular Orbitals on GPUs and Multi-core CPUs. J. E. Stone, J. Saam, D. Hardy, K. Vandivort, W. Hwu, K. Schulten, *2nd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU-2), ACM International Conference Proceeding Series*, volume 383, pp. 9-18, 2009.

MO GPU Parallel Decomposition

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



…

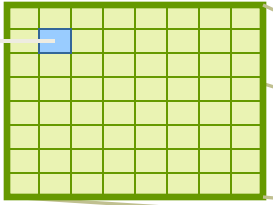
GPU 2

GPU 1

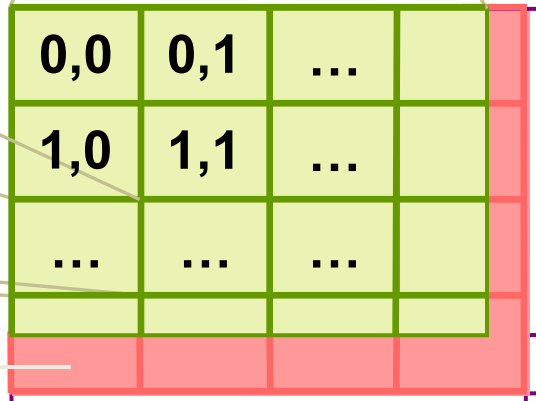
GPU 0

Lattice computed using multiple GPUs

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



Each thread computes one MO lattice point.



Threads producing results that are used

Threads producing results that are discarded

Grid of thread blocks

Padding optimizes global memory performance, guaranteeing coalesced global memory accesses

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

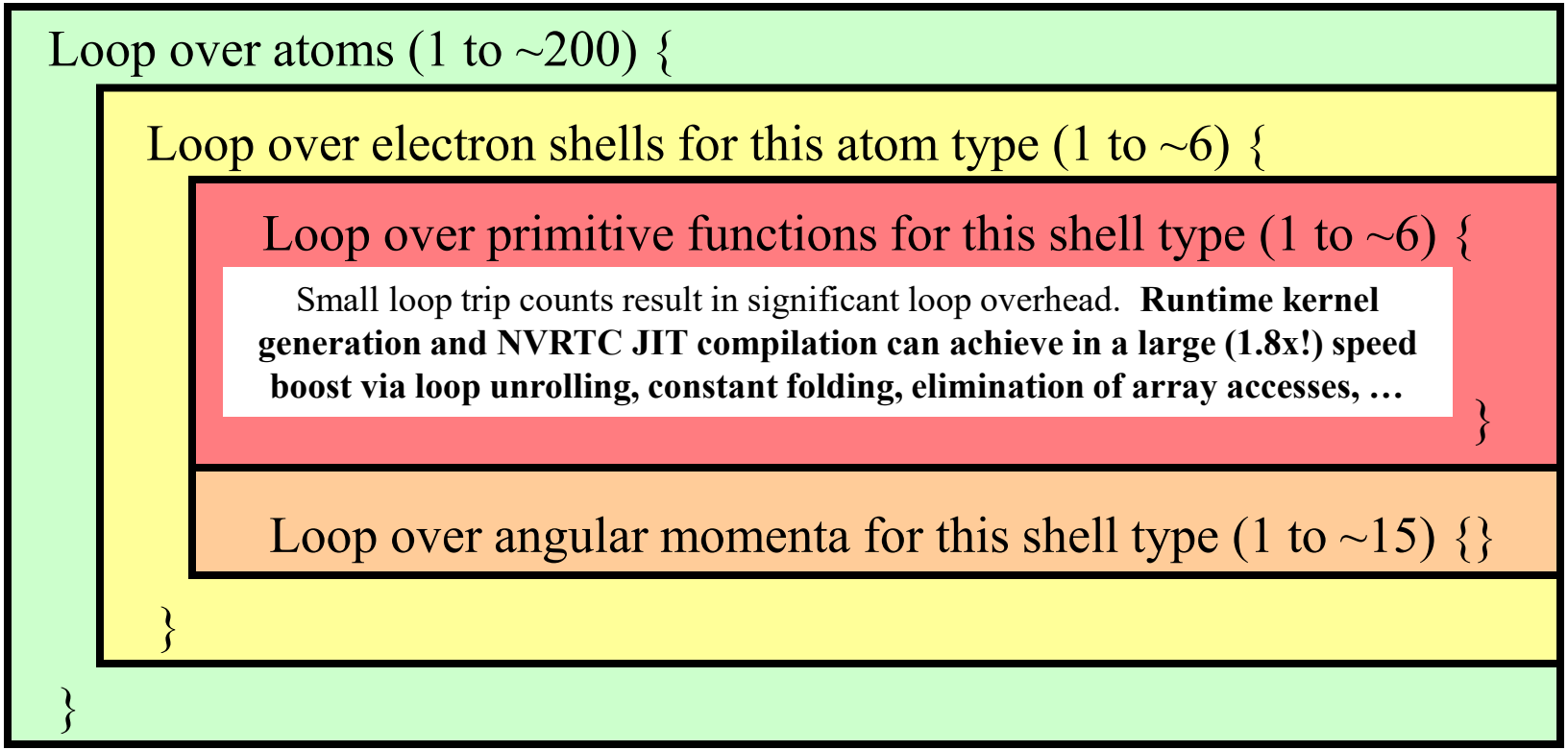
Loop over primitives: largest component of runtime, due to **expf()**

Loop over angular momenta

(unrolled in real code)

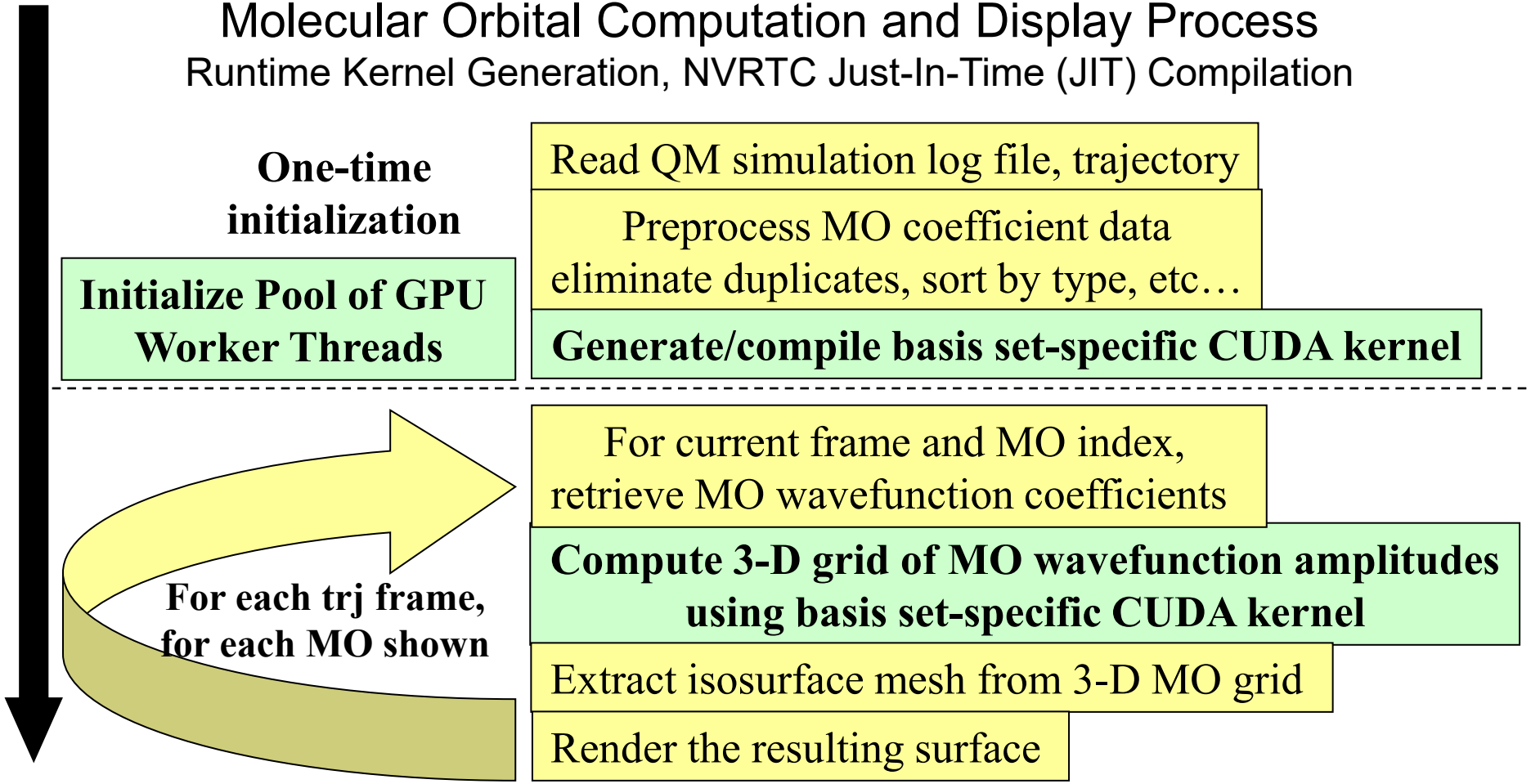
MO Kernel Structure, Opportunity for NRTC JIT…

Data-driven execution, but representative loop trip counts in (…)



Molecular Orbital Computation and Display Process

Runtime Kernel Generation, NVRTC Just-In-Time (JIT) Compilation



```
for (shell=0; shell < maxshell; shell++)  {
  float contracted_gto = 0.0f;

  // Loop over the Gaussian primitives of CGTO
  int maxprim = const_num_prim_per_shell[shell_counter];
  int shell_type = const_shell_symmetry[shell_counter];
  for (prim=0; prim < maxprim;  prim++) {
    float exponent         = const_basis_array[prim_counter     ];
    float contract_coeff = const_basis_array[prim_counter + 1];
    contracted_gto += contract_coeff  * expf(-exponent*dist2);
    prim_counter += 2;
  }
```

General loop-based data-dependent MO CUDA kernel

Runtime-generated data-specific MO CUDA kernel compiled via **CUDA NVRTC** JIT…

```
contracted_gto = 1.832937 * expf(-7.868272*dist2);
contracted_gto += 1.405380 * expf(-1.881289*dist2);
contracted_gto += 0.701383 * expf(-0.544249*dist2);
```

1.8x Faster

```
for (shell=0; shell < maxshell; shell++)  {
 float contracted_gto = 0.0f;

 // Loop over the Gaussian primitives of CGTO
 int maxprim = const_num_prim_per_shell[shell_counter];
 int shell_type = const_shell_symmetry[shell_counter];
 for (prim=0; prim < maxprim;  prim++) {
  float exponent         = const_basis_array[prim_counter     ];
  float contract_coeff = const_basis_array[prim_counter + 1];
  contracted_gto += contract_coeff  * expf(-exponent*dist2);
  prim_counter += 2;
 }

 float tmpshell=0;
 switch (shell_type) {
  case S_SHELL:
   value += const_wave_f[ifunc++] * contracted_gto;
   break;
[…..]
  case D_SHELL:
   tmpshell += const_wave_f[ifunc++] * xdist2;
   tmpshell += const_wave_f[ifunc++] * ydist2;
   tmpshell += const_wave_f[ifunc++] * zdist2;
   tmpshell += const_wave_f[ifunc++] * xdist * ydist;
```

General loop-based data-dependent MO CUDA kernel

Runtime-generated data-specific MO CUDA kernel compiled via **CUDA NVRTC** JIT…

1.8x Faster

```
contracted_gto = 1.832937 * expf(-7.868272*dist2);
contracted_gto += 1.405380 * expf(-1.881289*dist2);
contracted_gto += 0.701383 * expf(-0.544249*dist2);
// P_SHELL
tmpshell = const_wave_f[ifunc++] * xdist;
tmpshell += const_wave_f[ifunc++] * ydist;
tmpshell += const_wave_f[ifunc++] * zdist;
value += tmpshell * contracted_gto;

contracted_gto = 0.187618 * expf(-0.168714*dist2);
// S_SHELL
value += const_wave_f[ifunc++] * contracted_gto;

contracted_gto = 0.217969 * expf(-0.168714*dist2);
// P_SHELL
tmpshell = const_wave_f[ifunc++] * xdist;
tmpshell += const_wave_f[ifunc++] * ydist;
tmpshell += const_wave_f[ifunc++] * zdist;
value += tmpshell * contracted_gto;

contracted_gto = 3.858403 * expf(-0.800000*dist2);
// D_SHELL
tmpshell = const_wave_f[ifunc++] * xdist2;
tmpshell += const_wave_f[ifunc++] * ydist2;
```

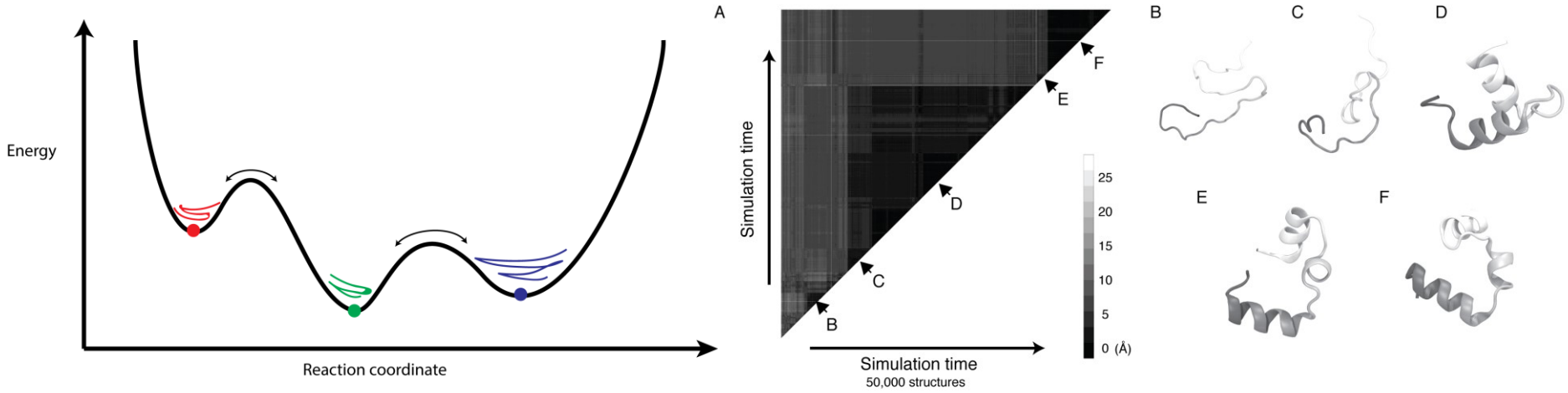
Challenges Adapting Large Software Systems for State-of-the-Art Hardware Platforms

- Initial focus on key computational kernels eventually gives way to the need to optimize an **ocean of less critical routines**, due to observance of Amdahl's Law
- Even though these less critical routines might be easily ported to CUDA or similar, the sheer number of routines often poses a challenge
- Need a low-cost approach for **getting "some" speedup** out of these second-tier routines
- In many cases, it is completely **sufficient to achieve memory-bandwidth-bound GPU performance with an existing algorithm**

Directive-Based Parallel Programming with OpenACC

- Annotate loop nests in existing code with #pragma compiler directives:
  - Annotate opportunities for parallelism
  - Annotate points where host-GPU memory transfers are best performed, indicate propagation of data
- Evolve original code structure to improve efficacy of parallelization
  - Eliminate false dependencies between loop iterations
  - Revise algorithms or constructs that create excess data movement

Clustering Analysis of Molecular Dynamics Trajectories



GPU-Accelerated Molecular Dynamics Clustering Analysis with OpenACC. J.E. Stone, J.R. Perilla, C. K. Cassidy, and K. Schulten. In, Robert Farber, ed., Parallel Programming with OpenACC, Morgan Kaufmann, Chapter 11, pp. 215-240, 2016.

Serial QCP RMSD Inner Product Loop

- Simple example where directive based parallelism can be applied easily and effectively
- Such a loop is inherently a memory-bandwidth-bound algorithm, so that's the goal for acceleration

```
for (int l=0; l<cnt; l++) {
  double x1, x2, y1, y2, z1, z2;
  x1 = crdx1[l];
  y1 = crdy1[l];
  z1 = crdz1[l];

  G1 += x1*x1 + y1*y1 + z1*z1;

  x2 = crdx2[l];
  y2 = crdy2[l];
  z2 = crdz2[l];

  G2 += x2*x2 + y2*y2 + z2*z2;

  a0 += x1 * x2;
  a1 += x1 * y2;
  a2 += x1 * z2;

  a3 += y1 * x2;
  a4 += y1 * y2;
  a5 += y1 * z2;

  a6 += z1 * x2;
  a7 += z1 * y2;
  a8 += z1 * z2;
}
```

OpenACC QCP RMSD Inner Product Loop

- Simple example where directive based parallelism can be applied easily and effectively
- Such a loop is inherently a memory-bandwidth-bound algorithm, so that's the goal for acceleration

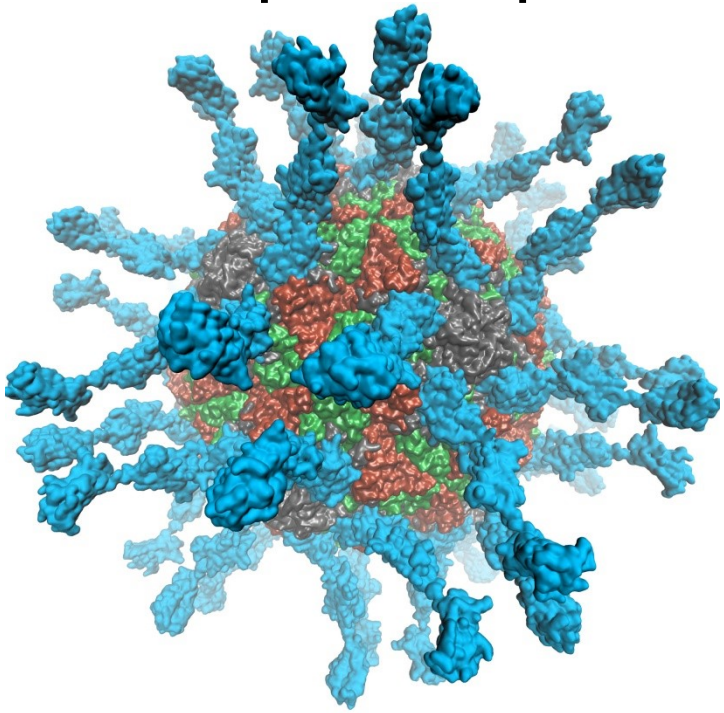
```
// excerpted code that has been abridged for brevity…
void rmsdmat_qcp_acc(int cnt, int padcnt, int framecrdsz,
                                int framecount, const float * restrict crds,
long i, j, k;
#pragma acc kernels copyin(crds[0:tsz]), copy(rmsdmat[0:msz])
  for (k=0; k<(framecount*(framecount-1))/2; k++) {
    // compute triangular matrix index 'k' in a helper function
    // to ensure that the compiler doesn't think that we have
    // conflicts or dependencies between loop iterations
    acc_idx2sub_tril(long(framecount-1), k, &i, &j);
    long x1addr = j * 3L * framecrdsz;
    long x2addr = i * 3L * framecrdsz;

#pragma acc loop vector(256)
    for (long l=0; l<cnt; l++) {
    // abridged for brevity ...

    rmsdmat[k]=rmsd; // store linearized triangular matrix
  }
}
```

OpenACC QCP RMSD Inner Product Loop Performance Results

- Xeon 2867W v3, w/ hand-coded AVX and FMA intrinsics: 20.7s
- Tesla K80 w/ OpenACC: **6.5s (3.2x speedup)**
- OpenACC on K80 achieved 65% of theoretical peak memory bandwidth, with 2016 compiler and just a few lines of #pragma directives. Excellent speedup for minimal changes to code.
- Future OpenACC compiler revs should provide higher performance yet

VMD 1.9.3 supports EGL for in-situ and parallel rendering on clouds, clusters, and supercomputers

- Eliminate dependency on windowing systems
- Simplified deployment of parallel VMD builds supporting off-screen rendering
- Maintains 100% of VMD OpenGL shaders and rendering features
- Support high-quality vendor-supported commercial OpenGL implementations in HPC systems that were previously limited to Mesa



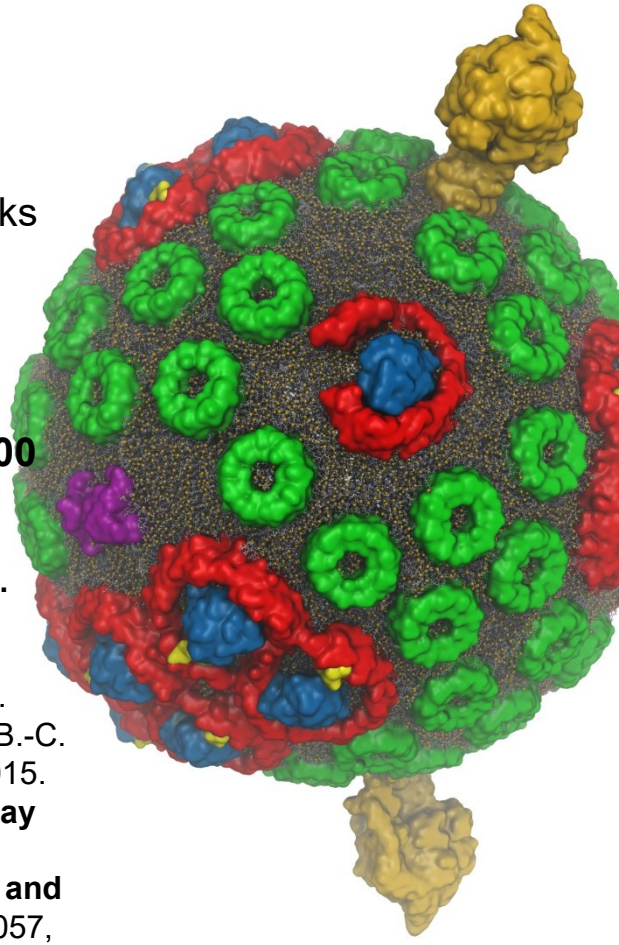
Poliovirus

EGL Is Supported Now!

- Cloud+Workstations with most recent NVIDIA drivers
- VMD on HPC systems w/ latest Tesla P100 GPUs:
  - Cray XC50, CSCS Piz Daint, driver 375.39
  - IBM OpenPOWER, drivers 375.66 and later support both GLX and EGL

VMD w/ OptiX 4.1

- Interactive RT on laptops, desktops, and cloud
- Large-scale parallel rendering: in situ or post hoc visualization tasks
- Remote RT on NVIDIA VCA clusters
- Stereoscopic panoramic and full-dome projections
- Omnidirectional VR for YouTube, VR HMDs
- **Top-end Pascal Tesla GPUs roughly 2x faster than Kepler**
- **GPU memory sharing via NVLink on Quadro GP100, Tesla P100**



VMD/OptiX GPU Ray Tracing of all-atom Chromatophore w/ lipids.

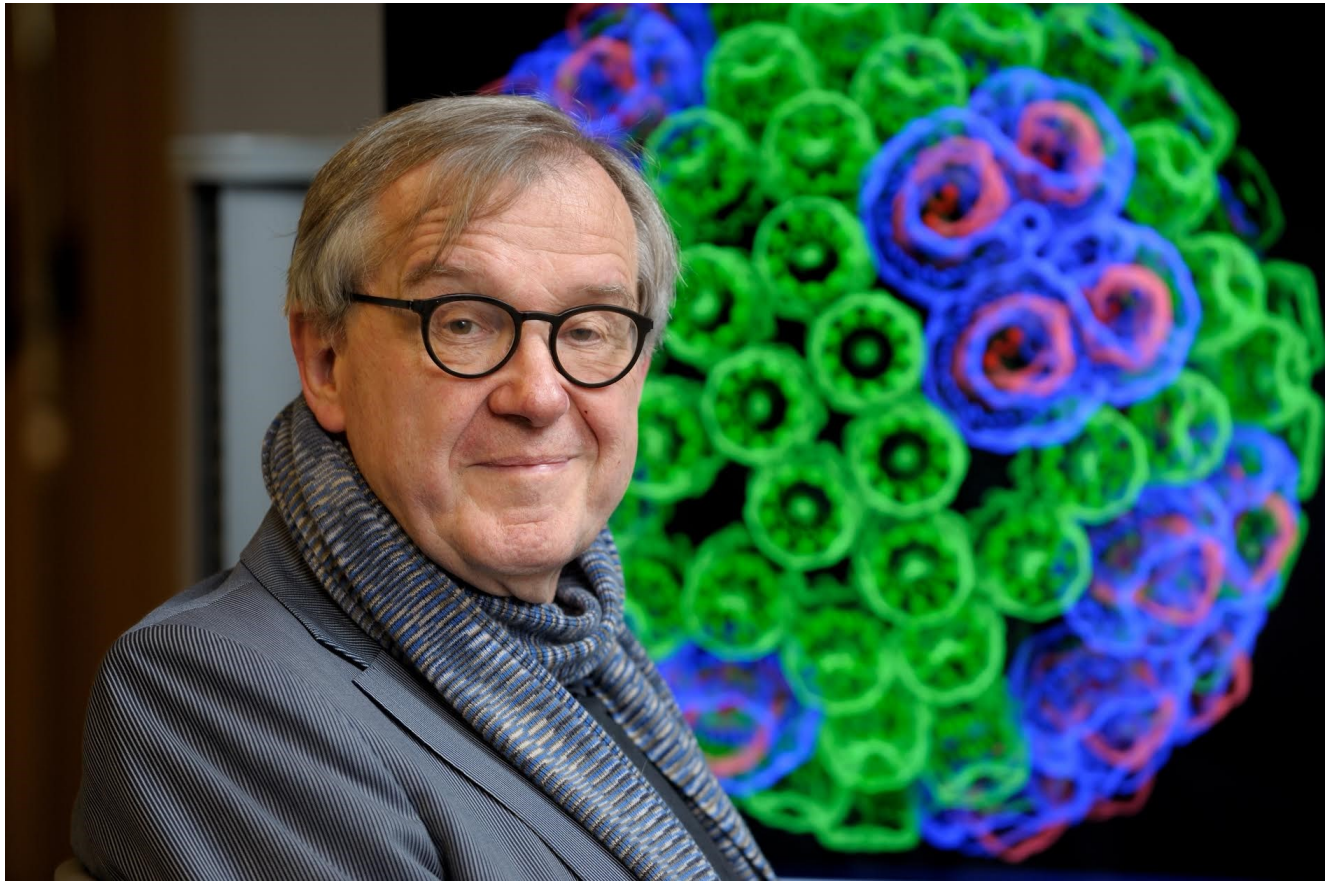
GPU-Accelerated Molecular Visualization on Petascale Supercomputing Platforms. J. E. Stone, K. L. Vandivort, and K. Schulten. UltraVis'13, pp. 6:1-6:8, 2013.

Visualization of Energy Conversion Processes in a Light Harvesting Organelle at Atomic Detail. M. Sener, et al. SC'14 Visualization and Data Analytics Showcase, 2014.

Chemical Visualization of Human Pathogens: the Retroviral Capsids. J. R. Perilla, B.-C. Goh, J. E. Stone, and K. Schulten. SC'15 Visualization and Data Analytics Showcase, 2015.

Atomic Detail Visualization of Photosynthetic Membranes with GPU-Accelerated Ray Tracing. J. E. Stone et al., J. Parallel Computing, 55:17-27, 2016.

Immersive Molecular Visualization with Omnidirectional Stereoscopic Ray Tracing and Remote Rendering J. E. Stone, W. R. Sherman, and K. HPDAV, IPDPSW, pp. 1048-1057, 2016.



“When I was a young man, my goal was to look with mathematical and computational means at the inside of cells, one atom at a time, to decipher how living systems work. That is what I strived for and I never deflected from this goal.” – Klaus Schulten

Acknowledgements

- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- CUDA Center of Excellence, University of Illinois at Urbana-Champaign
- NVIDIA CUDA and OptiX teams
- Funding:
  - DOE INCITE, ORNL Titan: DE-AC05-00OR22725
  - NSF Blue Waters:
    NSF OCI 07-25070, PRAC “The Computational Microscope”,
    ACI-1238993, ACI-1440026
  - NIH support: 9P41GM104601, 5R01GM098243-02

Related Publications

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

- **Challenges of Integrating Stochastic Dynamics and Cryo-electron Tomograms in Whole-cell Simulations.** T. M. Earnest, R. Watanabe, J. E. Stone, J. Mahamid, W. Baumeister, E. Villa, and Z. Luthey-Schulten. J. Physical Chemistry B, 121(15): 3871-3881, 2017.
- **Early Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms.** J. E. Stone, A.-P. Hynninen, J. C. Phillips, and K. Schulten. International Workshop on OpenPOWER for HPC (IWOPH'16), LNCS 9945, pp. 188-206, 2016.
- **Immersive Molecular Visualization with Omnidirectional Stereoscopic Ray Tracing and Remote Rendering.** J. E. Stone, W. R. Sherman, and K. Schulten. High Performance Data Analysis and Visualization Workshop, IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), pp. 1048-1057, 2016.
- **High Performance Molecular Visualization: In-Situ and Parallel Rendering with EGL.** J. E. Stone, P. Messmer, R. Sisneros, and K. Schulten. High Performance Data Analysis and Visualization Workshop, IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), pp. 1014-1023, 2016.
- **Evaluation of Emerging Energy-Efficient Heterogeneous Computing Platforms for Biomolecular and Cellular Simulation Workloads.** J. E. Stone, M. J. Hallock, J. C. Phillips, J. R. Peterson, Z. Luthey-Schulten, and K. Schulten.25th International Heterogeneity in Computing Workshop, IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), pp. 89-100, 2016.
- **Atomic Detail Visualization of Photosynthetic Membranes with GPU-Accelerated Ray Tracing.** J. E. Stone, M. Sener, K. L. Vandivort, A. Barragan, A. Singharoy, I. Teo, J. V. Ribeiro, B. Isralewitz, B. Liu, B.-C. Goh, J. C. Phillips, C. MacGregor-Chatwin, M. P. Johnson, L. F. Kourkoutis, C. Neil Hunter, and K. Schulten. J. Parallel Computing, 55:17-27, 2016.

Related Publications

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

- **Chemical Visualization of Human Pathogens: the Retroviral Capsids.** Juan R. Perilla, Boon Chong Goh, John E. Stone, and Klaus Schulten. SC'15 Visualization and Data Analytics Showcase, 2015.
- **Visualization of Energy Conversion Processes in a Light Harvesting Organelle at Atomic Detail.** M. Sener, J. E. Stone, A. Barragan, A. Singharoy, I. Teo, K. L. Vandivort, B. Isralewitz, B. Liu, B. Goh, J. C. Phillips, L. F. Kourkoutis, C. N. Hunter, and K. Schulten. SC'14 Visualization and Data Analytics Showcase, 2014.
  ***Winner of the SC'14 Visualization and Data Analytics Showcase**
- **Runtime and Architecture Support for Efficient Data Exchange in Multi-Accelerator Applications.** J. Cabezas, I. Gelado, J. E. Stone, N. Navarro, D. B. Kirk, and W. Hwu. IEEE Transactions on Parallel and Distributed Systems, 26(5):1405-1418, 2015.
- **Unlocking the Full Potential of the Cray XK7 Accelerator.** M. D. Klein and J. E. Stone. Cray Users Group, Lugano Switzerland, May 2014.
- **GPU-Accelerated Analysis and Visualization of Large Structures Solved by Molecular Dynamics Flexible Fitting.** J. E. Stone, R. McGreevy, B. Isralewitz, and K. Schulten. Faraday Discussions, 169:265-283, 2014.
- **Simulation of reaction diffusion processes over biologically relevant size and time scales using multi-GPU workstations**. M. J. Hallock, J. E. Stone, E. Roberts, C. Fry, and Z. Luthey-Schulten. Journal of Parallel Computing, 40:86-99, 2014.

Related Publications

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

- **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. 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. 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. 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. L. 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.

Related 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.

Related 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. E. 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.

Related 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.