ATPESC Track 7: Scalable Molecular Visualization and Analysis Tools in 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/
Argonne Training Program on Exascale Computing (ATPESC)
3:30pm-4:30pm, St. Charles Amphitheater, Q Center,
St. Charles, IL, Thursday August 10\textsuperscript{th}, 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/

Structure

- X-ray, cryo-EM, cryo-ET, NMR

Parameterization

Refinement with MDFF

Analysis

Preparation with QwikMD

MD/Cell Simulation

Parallel Analysis

Remote Visualization

Cell-Scale Modeling

MD Simulation
VMD Tutorial Home Page

- http://www.ks.uiuc.edu/Training/Tutorials/
  - Main VMD tutorial
  - QwikMD simulation preparation and analysis plugin
  - VMD images and movies tutorial
  - Structure check
  - VMD quantum chemistry visualization tutorial
  - Visualization and analysis of CPMD data with VMD
  - Parameterizing small molecules using ffTK
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
VMD is a Platform for Developing Research Tools
Over 110 VMD Plugins, Half Developed by Users

- VMD user-extensible scripting w/ Tcl/Tk, Python
- User-developed plugins:
  - Alanine Scanning
  - Collective Variable Analyzer
  - Clustering Tool
  - Carbon Nanostructure Builder
  - TorsionPlot
  - RMSD Trajectory Tool
  - Many others…

---

Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics
Beckman Institute, University of Illinois at Urbana-Champaign - www ks.uiuc.edu
QwikMD: Guided MD Simulation and Training

Smoothes initial learning curve (non-expert users)

Training: used in 4 Center workshops to-date

Speed up tedious simulation preparation tasks (expert users)

Reproducibility: detailed log of all steps

Interactive preparation, simulation, and analysis

**Selected VMD Plugins: Center Developed, and **User Developed**

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Modeling</th>
<th>Visualization</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>APBSRun</td>
<td>AutoIonize</td>
<td>Clipping Plane Tool</td>
<td>Remote Control</td>
</tr>
<tr>
<td>CatDCD</td>
<td>AutoPSF</td>
<td>Clone Rep</td>
<td>Data Import and Plotting</td>
</tr>
<tr>
<td>Contact Map</td>
<td>Chirality</td>
<td>DemoMaster</td>
<td>Multiplot</td>
</tr>
<tr>
<td><strong>GofRGUI</strong></td>
<td>Cionize</td>
<td>DipoleWatcher</td>
<td>PDBTool</td>
</tr>
<tr>
<td><strong>HeatMapper</strong></td>
<td>Cispeptide</td>
<td><strong>Intersurf</strong></td>
<td>MultiText</td>
</tr>
<tr>
<td><strong>ILSTools</strong></td>
<td>CGTools</td>
<td>Navigate</td>
<td></td>
</tr>
<tr>
<td><strong>IRSpecGUI</strong></td>
<td>Dowser</td>
<td>NavFly</td>
<td></td>
</tr>
<tr>
<td>MultiSeq</td>
<td>fTK</td>
<td><strong>MultiMolAnim</strong></td>
<td></td>
</tr>
<tr>
<td>NAMD Energy</td>
<td>InorganicBuilder</td>
<td>Color Scale Bar</td>
<td></td>
</tr>
<tr>
<td>NAMD Plot</td>
<td>MDFP</td>
<td>Remote</td>
<td></td>
</tr>
<tr>
<td>NetworkView</td>
<td>Membrane</td>
<td>Palette Tool</td>
<td></td>
</tr>
<tr>
<td><strong>NMWiz</strong></td>
<td>MergeStructs</td>
<td>ViewChangeRender</td>
<td></td>
</tr>
<tr>
<td>ParseFEP</td>
<td>Molefacture</td>
<td>ViewMaster</td>
<td></td>
</tr>
<tr>
<td>PBCTools</td>
<td>Mutator</td>
<td>Virtual DNA Viewer</td>
<td></td>
</tr>
<tr>
<td>PMEpot</td>
<td><strong>NanoTube</strong></td>
<td>VMD Movie Maker</td>
<td></td>
</tr>
<tr>
<td><strong>PropKa GUI</strong></td>
<td>Psfgen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ramapo</td>
<td>RESPTool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSD Tool</td>
<td><strong>RESPTool</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSD Trajectory Tool</td>
<td>RNAView</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSD Visualizer Tool</td>
<td>Solvate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Salt Bridges</td>
<td>SSRestraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Viewer</td>
<td>TopoTools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetry Tool</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TorsionPlot</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VolMap</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Simulation**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AlaScan</td>
<td>AutoIMD</td>
<td>NAMD GUI</td>
<td></td>
</tr>
<tr>
<td>AutoIMD</td>
<td>IMDMenu</td>
<td>NAMD Server</td>
<td></td>
</tr>
<tr>
<td>IMDDMenu</td>
<td>NAMD GUI</td>
<td>QMTool</td>
<td></td>
</tr>
</tbody>
</table>

**Externally Hosted Plugins and Extensions**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Check sidechains</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MultiMSMS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interactive Essential Dynamics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mead Ionize</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustering Tool</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iTrajComp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap RMSD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SurfVol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vmdICE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**75 MolFile I/O Plugins:**
structure, trajectory, sequence, and density map

http://www.ks.uiuc.edu/Research/vmd/plugins/
Goal: A Computational Microscope
Study the molecular machines in living cells

Ribosome: target for antibiotics
Poliovirus
Structure Visualization

Molecular representations provide different levels of abstraction, atomic detail vs. higher level organizational information:

- Atoms, VdW spheres, bonds, ball-stick, …
- Molecular orbitals (quantum chemistry)
- Molecular surfaces
- Coarse-grained “beads”
- Ribbons, secondary structure, “cartoon” reps, RNA/DNA
VMD Representation Examples

Ribosome, J. Frank

GroEL /w Situs

Aquaporin channel
Selection, Filtering

- Most viz tools allow interactive visual picking, menu-driven selections of structure components to display or operate on.
- VMD also extensively uses a text-based selection language (think google):
  - water within 10 of protein and z > 0
  - nucleic or protein or ions
  - segname BR
  - name “C.*”
  - Allows selection on user-defined data fields
  - **Promotes synergy between interactive and scripting interfaces, visualization and quantitative analysis tasks**
  - Works well with huge time-varying structures
Computing Molecular Properties

Compute properties, e.g., density, distance, occupancy, electrostatic potential over thousands of MD simulation trajectory frames

Example: display binding sites for diffusively bound ions as probability density isosurfaces

Run calculations in parallel on clusters, supercomputers with thousands of nodes, Blue Waters, Titan, etc …

Reduce 5 years of runtime to 20 minutes on 2048 GPU-accelerated nodes!

Parallel I/O: 275 GB/sec on 8192 nodes
Computed Properties

- Smoothing of thermal noise
- Secondary structure
- Hydrogen bonds, salt bridges
- Forces, energies, stress, strain
- Time averaging of electrostatic fields, occupancy maps
- Quality-of-fit cross correlation with cryo-EM density maps
- Normal modes, principal component analysis, essential dynamics
- Cluster simulation trajectory timesteps by structural similarity

Chemoreceptor trimer-of-dimers analysis with Bendix plugin in VMD
Display of Computed Properties on Structures

Per-residue solvent-accessible surface area of Ubiquitin

PME electrostatic potential contour for a helicase on a volumetric slice plane
CheA kinase PCA: first principal component porcupine plot
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
10 Years of GPU Computing in VMD

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

Blast from the past:
CUmDA starting with version 0.7 !!!
Quad core Intel QX6700, three NVIDIA GeForce 8800GTX GPUs, RHEL4 Linux

**Interactive Remote** Visualization and Analysis

- Enabled by hardware H.264/H.265 video encode/decode
- Enable visualization and analyses not possible with conventional workstations
- Access data located anywhere in the world
  - Same VMD session available to any device
- Linux prototype in-development using NVIDIA Video Codec SDK, easy-to-use *NvPipe* wrapper library
NVIDIA Video CODEC SDK and NvPipe

- GPUs (Kepler-on) include NVENC and NVDEC video codec acceleration hardware
- Independent of GPU compute hardware
- Hardware-accelerated codecs can overlap with interactive rendering, and computation
- NvPipe provides an easy to use API for interactive video streaming, abstracting many low level codec details, ideal for basic remote visualization implementations: https://github.com/NVIDIA/NvPipe
NvPipe
https://github.com/NVIDIA/NvPipe

• Simplified API for producing a basic encoder/decoder system.
• Roughly 100 lines of code for basic encode/decode “Hello World” loops with minimal error handling logic
• Encode/decode ends up being simpler than your networking code 😊
• Encode loop structure:
  – User selects encoder type, e.g. NVPIPE_H264_NV, and target encoder bitrate parameter
  – User provides uncompressed RGB or RGBA image buffer, image dimensions, and size of the output memory buffer
  – NvPipe compresses the frame using the NVENC hardware encoder, and returns the number of bytes of output written to the output buffer
• Symmetric decode loop structure:
  – Provide decoder with compressed buffer, buffer size in bytes, and image dimensions as input
  – Decoder produces uncompressed output image
• Optionally supports FFMPEG back-ends (but I haven’t tried those 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
OpenGL: GLX vs. EGL

- Viz Application (user)
  - OpenGL
  - X server (root)
  - GLX
  - Driver
  - GPU

- Viz Application (user)
  - OpenGL
  - EGL
  - Driver
  - GPU
Swine Flu A/H1N1 neuraminidase bound to Tamiflu: VMD EGL rendering demonstrating full support for all VMD shaders and OpenGL features, multisample antialiasing, ray cast spheres, 3-D texture mapping, ...
Benefits of EGL Platform Interfaces

• Enumerate and select among available platforms, potentially supporting multiple vendors in the same host/node
  – Allows specific target implementation to be bound, e.g. GPU, CPU-integrated GPU, software rasterizer

• **EGL interfaces make it EASY to bind a GPU to a thread with optimal CPU affinity with respect to NUMA topology, NVLink GPU topology**
  – High-perf. multi-GPU image compositing, video streaming
  – EGL plays nicely with MPI, CUDA/OpenCL, OptiX, NVENC, etc
  – NVIDIA EGL supports multiple GPU indexing schemes, e.g. **PCIe ordering**
  – **Exploit NVLink interconnect topology on IBM OpenPOWER platforms, e.g. “Minsky”, upcoming DOE/ORNL “Summit” system**
VMD EGL Performance on Amazon EC2 Cloud

<table>
<thead>
<tr>
<th>MPI Ranks</th>
<th>EC2 “G2.8xlarge” GPU Instances</th>
<th>HIV-1 movie rendering time (sec), (I/O %) 3840x2160 resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>626s (10% I/O)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>347s (19% I/O)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>221s (31% I/O)</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>141s (46% I/O)</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>107s (64% I/O)</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>90s (76% I/O)</td>
</tr>
</tbody>
</table>

Performance at 32 nodes reaches ~48 frames per second

64M atom HIV-1 capsid simulation rendered via EGL

Close-up view of HIV-1 hexamer rendered via EGL
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
High Fidelity Ray Tracing

- **Interactive RT** on laptops, desk, cloud
- Large-scale parallel rendering: in situ or post hoc visualization tasks
- AO, DoF, instancing, …. 
- Stereoscopic panorama and full-dome projections
- Omnidirectional VR: YouTube, HMDs

- Built-in ray tracing engines:
  - **Tachyon**: cross-platform RT
  - **NVIDIA OptiX**: GPU-accelerated and remote RT on VCA clusters
  - **Intel OSPRay**: CPU x86/Phi-optimized parallel rendering w/ MPI

VMD/OptiX all-atom Chromatophore
Lighting Comparison, STMV Capsid

Two lights, no shadows

Ambient occlusion + two lights, 144 AO rays/hit
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


Goal: Intuitive interactive viz. in crowded molecular complexes

Results from 64 M atom, 1 \( \mu \)s sim!

Close-up view of chloride ions permeating through HIV-1 capsid hexameric centers
HIV-1 Parallel HD Movie Rendering on Blue Waters Cray XE6/XK7

New VMD TachyonL-OptiX on XK7 vs. Tachyon on XE6:
K20X GPUs yield up to twelve times geom+ray tracing speedup

<table>
<thead>
<tr>
<th>Ray Tracer Version</th>
<th>Node Type and Count</th>
<th>Script Load</th>
<th>State Load</th>
<th>Geometry + Ray Tracing</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>New TachyonL-OptiX</td>
<td>64 XK7 Tesla K20X GPUs</td>
<td>2 s</td>
<td>39 s</td>
<td>435 s</td>
<td>476 s</td>
</tr>
<tr>
<td>New TachyonL-OptiX</td>
<td>128 XK7 Tesla K20X GPUs</td>
<td>3 s</td>
<td>62 s</td>
<td>230 s</td>
<td>295 s</td>
</tr>
<tr>
<td>TachyonL-OptiX [1]</td>
<td>64 XK7 Tesla K20X GPUs</td>
<td>2 s</td>
<td>38 s</td>
<td>655 s</td>
<td>695 s</td>
</tr>
<tr>
<td>TachyonL-OptiX [1]</td>
<td>128 XK7 Tesla K20X GPUs</td>
<td>4 s</td>
<td>74 s</td>
<td>331 s</td>
<td>410 s</td>
</tr>
<tr>
<td>TachyonL-OptiX [1]</td>
<td>256 XK7 Tesla K20X GPUs</td>
<td>7 s</td>
<td>110 s</td>
<td>171 s</td>
<td>288 s</td>
</tr>
<tr>
<td>Tachyon [1]</td>
<td>256 XE6 CPUs</td>
<td>7 s</td>
<td>160 s</td>
<td>1,374 s</td>
<td>1,541 s</td>
</tr>
<tr>
<td>Tachyon [1]</td>
<td>512 XE6 CPUs</td>
<td>13 s</td>
<td>211 s</td>
<td>808 s</td>
<td>1,032 s</td>
</tr>
</tbody>
</table>

20 M atom chromatophore patch
VMD Shading Comparison: EF-Tu

Outline Shader

“Goodsell” Shader

Glossy Shader

Ambient Occlusion, Shadowing
Diverse Shading and Lighting Approaches

Decrystallization: Interactive Ray Tracing w/ Ambient Occlusion Lighting, Depth of Field Focal Blur

Myoglobin
Exportin Cse1p

Nanopore
Faraday Discussions
Volume: 169
Molecular Simulations and Visualization

Atomic structure of the AIDS pathogen’s protein coat
PAGE 640

THE FIRST LIGHT
In pursuit of the most distant galaxies
PAGE 564

CROSSING THE BORDERS
International collaboration makes the most impact
PAGE 571

A SITTING TARGET
An indirect hit on ‘undruggable’ KEAP1 protein
PAGE 577 & 588
Preparation, Visualization, Analysis of All-Atom Cell-Scale Simulations

- Interactive rasterization w/ OpenGL/EGL now, Vulkan in future releases of VMD
- Interactive ray tracing on CPUs and GPUs
- Support for large host memory (TB), up to 2 billion atoms per “molecule” now
- Parallel analysis, visualization w/ MPI


Interactive Ray Tracing of Cells

- High resolution cellular tomograms
- Multi-billion voxels
- Even isosurface or lattice site graphical representations involve ~100M geometric primitives
- 24GB Quadro M6000s used for interactive RT of cellular tomograms of this size
- Latest Quadro GP100 GPUs benefit from OptiX 4.1 support for NVLink and distribution of scene data across multiple GPUs

Proto-Cell Rendered with VMD+OptiX

- 113M particles
- 1,397 copies of 14 different membrane proteins
- Preparing for simulations on pre-exascale computers
Stereoscopic Panorama Ray Tracing

- Render 360° images and movies for VR headsets such as Oculus Rift, Google Cardboard
- Ray trace panoramic stereo spheremaps or cubemaps for very high-frame-rate display via OpenGL texturing onto simple geometry
- Stereo requires spherical camera projections poorly suited to rasterization
- Benefits from OptiX multi-GPU rendering and load balancing, remote visualization
VMD VR Demos

VMD VR ray tracing:
Google Cardboard [1]
Demo w/ Indiana U., SC’15 [2]

Prototype of VR user interaction with VMD models in room-scale VR with NVIDIA @ SC’16


Progressive Ray Tracing Engine
Ray tracing loop runs continuously in new thread
Decodes H.264 video stream from remote VCA GPU cluster

Omnistereo Image Stream

Camera + Scene

HMD Display Loop
HMD loop runs in main VMD application thread at max OpenGL draw rate
View-dependent stereo reprojection for current HMD head pose
HMD distortion correction

15Mbps Internet Link
Remote VCA GPU Cluster
Ray tracing runs continuously, streams H.264 video to VMD client
VMD TachyonL-OptiX: Multi-GPU on NVIDIA VCA Cluster

Scene Data Replicated, Image Space + Sample Space
Parallel Decomposition onto GPUs

VCA 0:
8 M6000 GPUs

VCA N:
8 M6000 GPUs
# Remote Omnidirectional Stereoscopic RT Performance @ 3072x1536 w/ 2-subframes

<table>
<thead>
<tr>
<th>Scene</th>
<th>Per-subframe samples AA : AO (AO per-hit)</th>
<th>RT update rate (FPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STMV shadows</td>
<td>1:0</td>
<td>22.2 18.1 10.3</td>
</tr>
<tr>
<td></td>
<td>2:0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4:0</td>
<td></td>
</tr>
<tr>
<td>STMV Shadows+AO</td>
<td>1:1</td>
<td>18.2 16.1 12.4</td>
</tr>
<tr>
<td></td>
<td>1:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1:4</td>
<td></td>
</tr>
<tr>
<td>STMV Shadows+AO+DoF</td>
<td>1:1</td>
<td>16.1 11.1 8.5</td>
</tr>
<tr>
<td></td>
<td>2:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2:2</td>
<td></td>
</tr>
<tr>
<td>HIV-1 Shadows</td>
<td>1:0</td>
<td>20.1 18.1 10.2</td>
</tr>
<tr>
<td></td>
<td>2:0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4:0</td>
<td></td>
</tr>
<tr>
<td>HIV-1 Shadows+AO</td>
<td>1:1</td>
<td>17.4 12.2 8.1</td>
</tr>
<tr>
<td></td>
<td>1:2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1:4</td>
<td></td>
</tr>
</tbody>
</table>
Planetarium Dome Master Projections
NSF CADENS “Birth of Planet Earth”
Cinematic Pipeline

Visualization and Rendering

VMD

Virtual Director

Camera Path

Geometry

Houdini

Compositing

Nuke

Cam Posns

Images
Integration of VMD with Virtual Director
NSF CADENS “Birth of Planet Earth” Test Frames
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

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

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

A mass-weighted force is then applied to each atom

$$f_{i}^{EM} = -\nabla U_{EM}(\mathbf{R}) = -w_i \frac{\partial V_{EM}(\mathbf{r}_i)}{\partial r_i}$$
Structural Route to the all-atom HIV-1 Capsid


Crystal structures of separated hexamer and pentamer

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

Hexamer of hexamers HIV capsid substructure Molecular Dynamics Flexible Fitting (MDFF) simulation. All-atom structure fitting into cryo-EM density map.
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
Padding optimizes global memory performance, guaranteeing coalesced global memory accesses.

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.

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

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

Threads producing results that are used.

Inactive threads, region of discarded output.
Parallel MDFF Cross Correlation Analysis on Cray XK7

Rabbit Hemorrhagic Disease Virus (RHDV)

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traj. frames</td>
<td>10,000</td>
</tr>
<tr>
<td>Structure component selections</td>
<td>720</td>
</tr>
<tr>
<td>Single-node XK7 (projected)</td>
<td>336 hours (14 days)</td>
</tr>
<tr>
<td>128-node XK7</td>
<td>3.2 hours 105x speedup</td>
</tr>
<tr>
<td>2048-node XK7</td>
<td>19.5 minutes 1035x speedup</td>
</tr>
</tbody>
</table>

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

## VMD Tesla P100 Cross Correlation Performance

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

Latest results on Pascal P100, first runs on Volta V100

<table>
<thead>
<tr>
<th>Hardware platform</th>
<th>Runtime, Speedup vs. Chimera, VMD+GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimera Xeon E5-2687W (2 socket) [1]</td>
<td>15.860s, 1x</td>
</tr>
<tr>
<td>VMD-CUDA IBM Power8 + 1x Tesla K40 [2]</td>
<td>0.488s, 32x 0.9x</td>
</tr>
<tr>
<td>VMD-CUDA Intel Xeon E5-2687W + 1x Quadro K6000 [1,2]</td>
<td>0.458s, 35x 1.0x</td>
</tr>
<tr>
<td>VMD-CUDA Intel Xeon E5-2698v3 + 1x Tesla P100</td>
<td>0.090s, 176x 5.1x</td>
</tr>
<tr>
<td>VMD-CUDA IBM Power8 “Minsky” + 1x Tesla P100</td>
<td>0.080s, 198x 5.7x</td>
</tr>
<tr>
<td><strong>VMD-CUDA Intel Xeon E5-2697Av4 + 1x Tesla V100</strong></td>
<td><strong>0.050s, 317x 9.2x</strong></td>
</tr>
</tbody>
</table>


Visualization of Molecular Orbitals

- 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
- **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**

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++;
    }
}
```
MO GPU Parallel Decomposition

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

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

Each thread computes one MO lattice point.

Padding optimizes global memory performance, guaranteeing coalesced global memory accesses

Grid of thread blocks

Threads producing results that are discarded

Lattice computed using multiple GPUs

Threads producing results that are used

GPU 0
GPU 1
GPU 2
### VMD Tesla P100 Performance for C\textsubscript{60} Molecular Orbitals, 516x519x507 grid

<table>
<thead>
<tr>
<th>Hardware platform</th>
<th>Runtime,</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Power8 (ORNL ‘crest’) + 1x Tesla K40 [1]</td>
<td>3.49s,</td>
<td>1.0x</td>
</tr>
<tr>
<td>Intel Xeon E5-2698v3 + 1x Tesla P100</td>
<td>1.35s,</td>
<td>2.5x</td>
</tr>
<tr>
<td>IBM Power8 “Minsky” + 1x Tesla P100</td>
<td>1.09s,</td>
<td>3.3x</td>
</tr>
<tr>
<td>IBM Power8 (ORNL ‘crest’) + 4x Tesla K40 [1]</td>
<td>0.91s,</td>
<td>3.8x</td>
</tr>
<tr>
<td>Intel Xeon E5-2698v3 + 4x Tesla P100</td>
<td>0.37s,</td>
<td>9.4x</td>
</tr>
<tr>
<td>IBM Power8 “Minsky” + 4x Tesla P100</td>
<td>0.30s,</td>
<td>11.6x</td>
</tr>
<tr>
<td>Intel Xeon E5-2697Av4 + 1x Tesla V100</td>
<td>0.610s,</td>
<td>5.7x</td>
</tr>
<tr>
<td>Intel Xeon E5-2697Av4 + 2x Tesla V100</td>
<td>0.294s,</td>
<td>11.8x</td>
</tr>
<tr>
<td>Intel Xeon E5-2697Av4 + 3x Tesla V100</td>
<td>0.220s,</td>
<td>15.9x</td>
</tr>
</tbody>
</table>

MO Kernel Structure, Opportunity for NRTC JIT…
Data-driven execution, but representative loop trip counts in (…)

Loop over atoms (1 to ~200) {
  Loop over electron shells for this atom type (1 to ~6) {
    Loop over primitive functions for this shell type (1 to ~6) {
      Small loop trip counts result in significant loop overhead. **Runtime kernel generation and NVRTC JIT compilation can achieve in a large (1.8x!) speed boost via loop unrolling, constant folding, elimination of array accesses, …**
    }
    Loop over angular momenta for this shell type (1 to ~15) {}
  }
}
Molecular Orbital Computation and Display Process
Runtime Kernel Generation, NVRTC Just-In-Time (JIT) Compilation

One-time initialization

Initialize Pool of GPU Worker Threads

Read QM simulation log file, trajectory
Preprocess MO coefficient data
dismiss duplicates, sort by type, etc…

Generate/compile basis set-specific CUDA kernel

For current frame and MO index,
retrieve MO wavefunction coefficients

Compute 3-D grid of MO wavefunction amplitudes
using basis set-specific CUDA kernel

Extract isosurface mesh from 3-D MO grid
Render the resulting surface

For each trj frame, for each MO shown

One-time initialization

Initialize Pool of GPU Worker Threads

Read QM simulation log file, trajectory
Preprocess MO coefficient data
dismiss duplicates, sort by type, etc…

Generate/compile basis set-specific CUDA kernel

For current frame and MO index,
retrieve MO wavefunction coefficients

Compute 3-D grid of MO wavefunction amplitudes
using basis set-specific CUDA kernel

Extract isosurface mesh from 3-D MO grid
Render the resulting surface

For each trj frame, for each MO shown
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;
    }
}

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

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;

1.8x Faster
MO CPU Parallel Decomposition
Early AVX-512 Kernels on KNL

MO 3-D lattice decomposes into 2-D slices

Vectors of wavefunction amplitudes are computed using hardware SIMD instructions

Each CPU thread computes 1, 4, 8, 16 MO lattice points per loop iteration: C, SSE, AVX2 or AVX-512ER

Padded + aligned array

SIMD lanes producing results that are used

Padding:
Inactive SIMD lanes or region of discarded output used to guarantee aligned vector loads+stores
AVX-512ER MO CGTO Loop

```
int maxprim = num_prim_per_shell[shell_counter];  
int shelltype = shell_types[shell_counter];  
for (prim=0; prim<maxprim; prim++) {
    float exponent = basis_array[prim_counter];  
    float contract_coeff = basis_array[prim_counter + 1];  

    // contracted_gto += contract_coeff * exp(exponent*dist2);  
    __m512 expval = _mm512_mul_ps(_mm512_set1_ps(exponent * MLOG2EF), dist2);  

    // expf() approximation required, use (base-2) AVX-512ER instructions…  
    __m512 retval = _mm512_exp2a23_ps(expval);  
    __m512 ct tmp = _mm512_mul_ps(_mm512_set1_ps(contract_coeff), retval);  
    contracted_gto = _mm512_add_ps(contracted_gto, ct tmp);  
    prim_counter += 2;
```

AVX-512ER MO Wavefunction Loop

/* multiply with the appropriate wavefunction coefficient */
__m512 ts = _mm512_set1_ps(0.0f);
switch (shelltype) {
  case S_SHELL:
    value = _mm512_add_ps(value, _mm512_mul_ps(_mm512_set1_ps(wave_f[ifunc++]), cgto));
    break;

  case P_SHELL:
    ts = _mm512_add_ps(ts, _mm512_mul_ps(_mm512_set1_ps(wave_f[ifunc++]), xdist));
    ts = _mm512_add_ps(ts, _mm512_mul_ps(_mm512_set1_ps(wave_f[ifunc++]), ydist));
    ts = _mm512_add_ps(ts, _mm512_mul_ps(_mm512_set1_ps(wave_f[ifunc++]), zdist));
    value = _mm512_add_ps(value, _mm512_mul_ps(ts, cgto));
    break;

  case D_SHELL:
    ....
AVX-512ER+FMA

MO Wavefunction Loop

/* multiply with the appropriate wavefunction coefficient */

__m512 ts = _mm512_set1_ps(0.0f);

switch (shelltype) {
    // use FMADD instructions
    case S_SHELL:
        value = _mm512_fmadd_ps(_mm512_set1_ps(wave_f[ifunc++]), cgto, value);
        break;

    case P_SHELL:
        ts = _mm512_fmadd_ps(_mm512_set1_ps(wave_f[ifunc++]), xdist, ts);
        ts = _mm512_fmadd_ps(_mm512_set1_ps(wave_f[ifunc++]), ydist, ts);
        ts = _mm512_fmadd_ps(_mm512_set1_ps(wave_f[ifunc++]), zdist, ts);
        value = _mm512_fmadd_ps(ts, cgto, value);
        break;
}

Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics
Beckman Institute, University of Illinois at Urbana-Champaign - www.ks.uiuc.edu
Performance of AVX-512ER Intrinsics vs. Autovectorization on KNL: Small 172x173x169 Grid

- Intel C++ ‘15 autovectorization (fail): 220+ sec
- Hand-coded SSE2 w/ existing thread scheme: 48.5 sec
- Hand-coded AVX-512ER w/ existing thread scheme: 6.3 sec
- Hand-coded AVX-512ER, refactoring thread pool: 0.2 sec
- Hand-coded AVX-512ER tuned thread pool: 0.131 sec
- Hand-coded AVX-512ER+FMA tweaks: 0.107 sec

Further improvement will require attention to details of cache behaviour and further tuning of low-level threading constructs for Xeon Phi/KNL
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

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

```c
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

```
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 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

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 outperform 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”
ARM + integrated GPU

X-Gene+GPU: fastest ARM platform

Note comparative performance for transfers ranging from 2KB to 16KB

- Intel (GTX TITAN)
- Intel (K20c)
- Jetson TX1
- Jetson TK1
- X-Gene
- CARMA
- KAYLA
Future Work

• New analysis features
• Support emerging platforms
• Many more CUDA GPU and AVX-512 CPU kernels…
• Interactive ray tracing of time-varying molecular geometry
• Vulkan graphics API

https://www.khronos.org/vulkan/
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
• Intel OSPRay team
• 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
“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
Related Publications

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


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


- **Unlocking the Full Potential of the Cray XK7 Accelerator.** M. D. Klein and J. E. Stone. Cray Users Group, Lugano Switzerland, May 2014.


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


Related Publications

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


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


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


