Immersive Out-of-Core Visualization of Large-Size and Long-Timescale Molecular Dynamics Trajectories

J. Stone, K. Vandivort, K. Schulten

Theoretical and Computational Biophysics Group

Beckman Institute for Advanced Science and Technology

University of Illinois at Urbana-Champaign

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

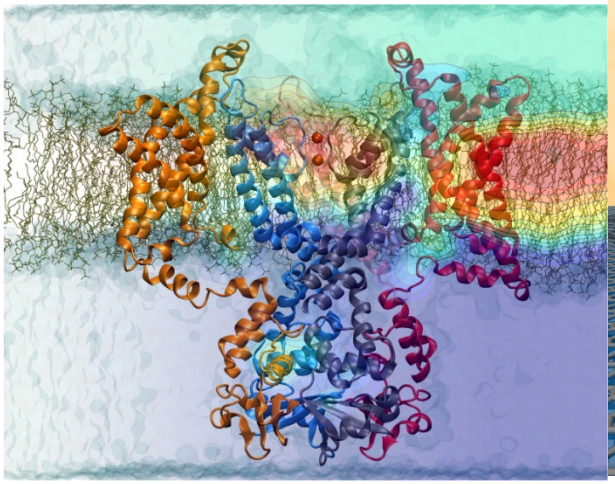
7th International Symposium on Visual Computing

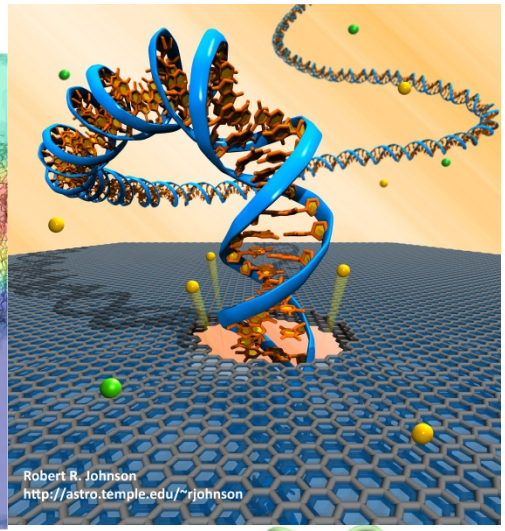
Special Track: Immersive Visualization

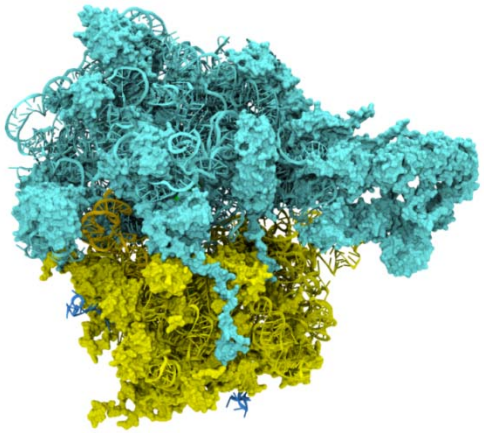
Las Vegas, NV, September 26, 2011

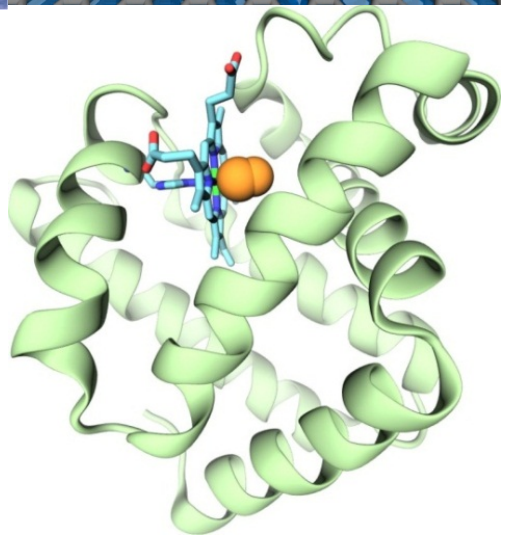
Visualizing Biomolecules

- Simplified structure representations
- Coloring by structural properties, volumetric fields, similarity to related structures, …
- High quality shading
- Depth cueing, ambient occlusion lighting
- Stereoscopic display
- Motion, animation of molecular dynamics



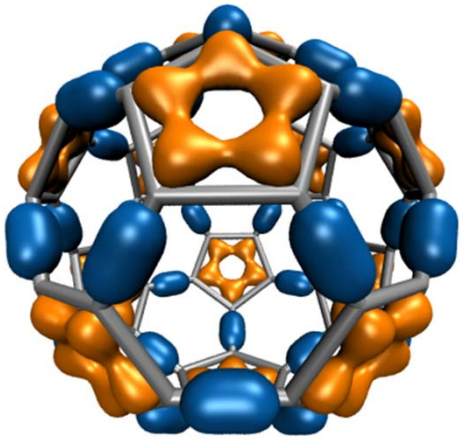


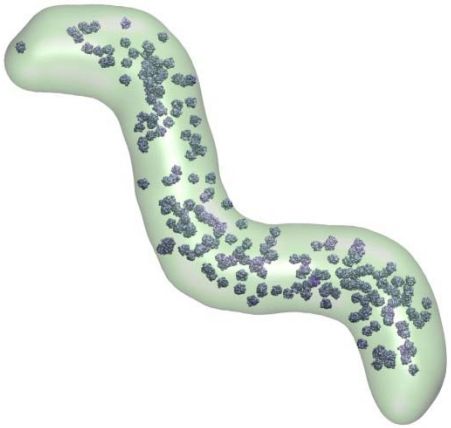


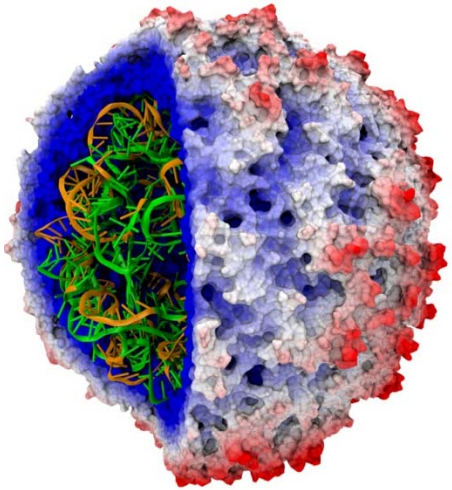


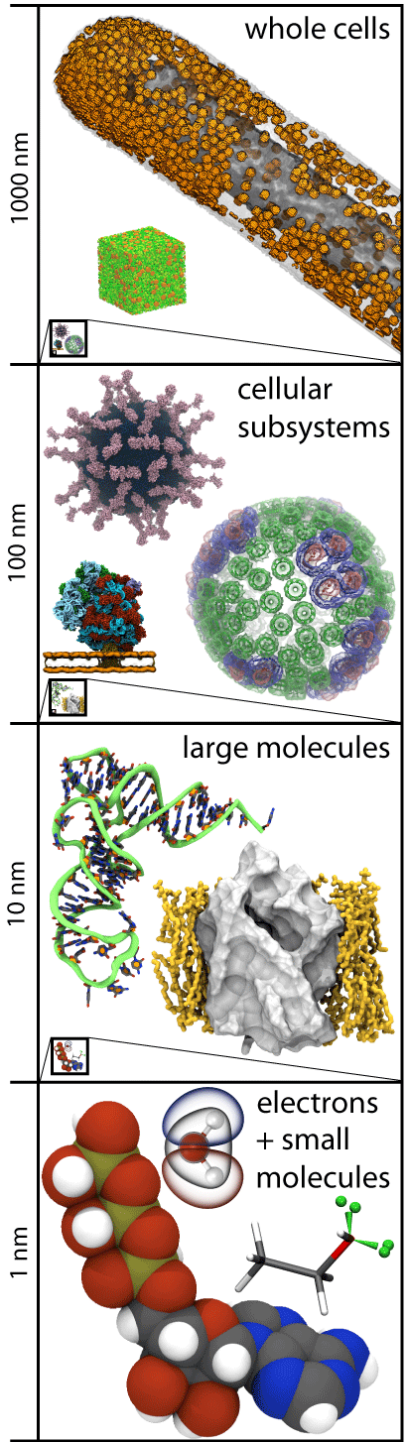
VMD – “Visual Molecular Dynamics”

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



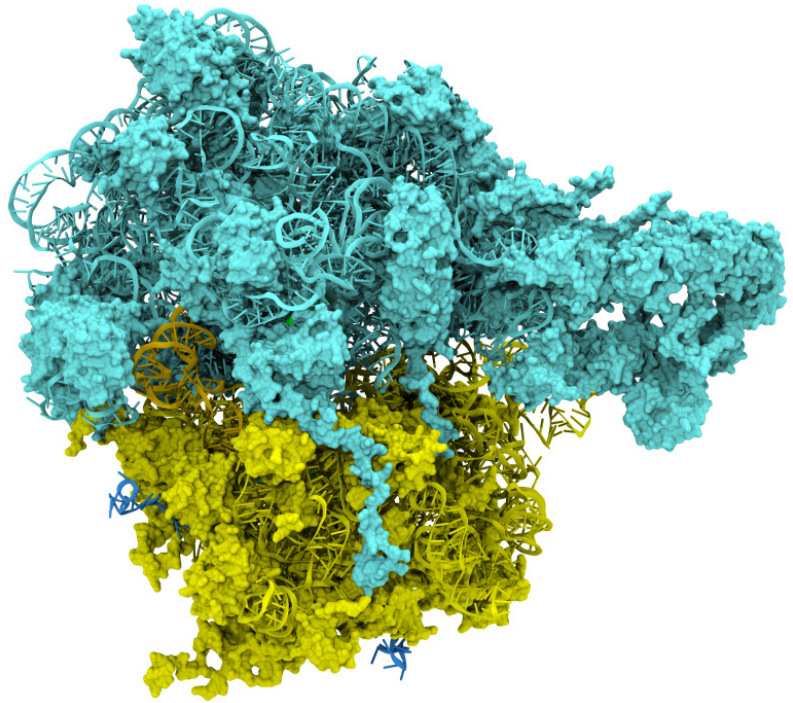






Goal: A Computational Microscope

- Study the molecular machines in living cells
- Health-relevant biomolecules are often large **multi-million atom complexes**
- Computer simulations on large parallel computers enable views of **dynamics** inaccessible to experiment
- Simulation trajectories (output) are many **terabytes** in size, far too large to load in memory, users juggle subsets of data…
- Out-of-core techniques can address size limitations, but achieving interactive performance is difficult
- By optimizing file formats, data structures, selection traversal, OpenGL rendering, and by using SSDs for fast I/O, out-of-core immersive visualization becomes feasible



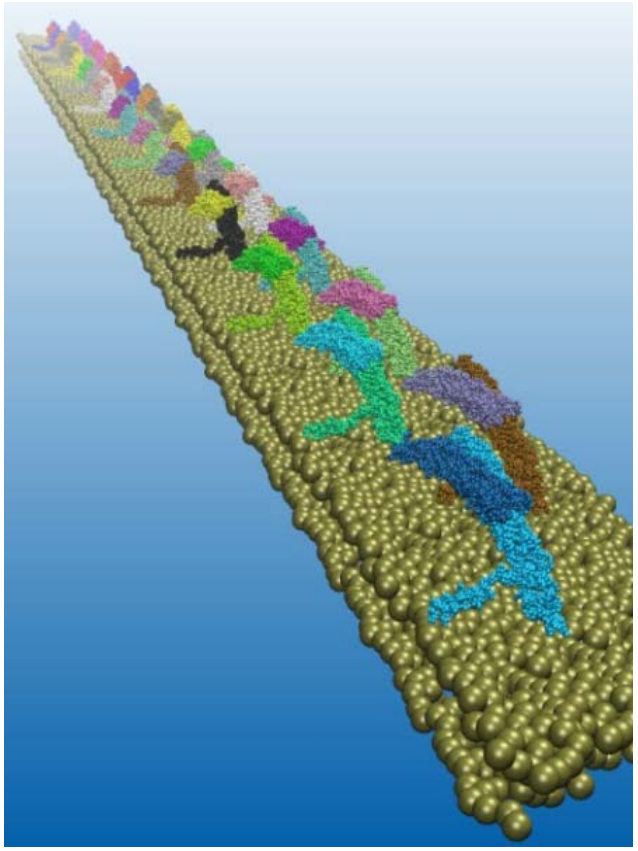
Ribosome: synthesizes proteins from genetic information, target for antibiotics

Data Challenges for Immersive Visualization of Dynamics of Large Structures

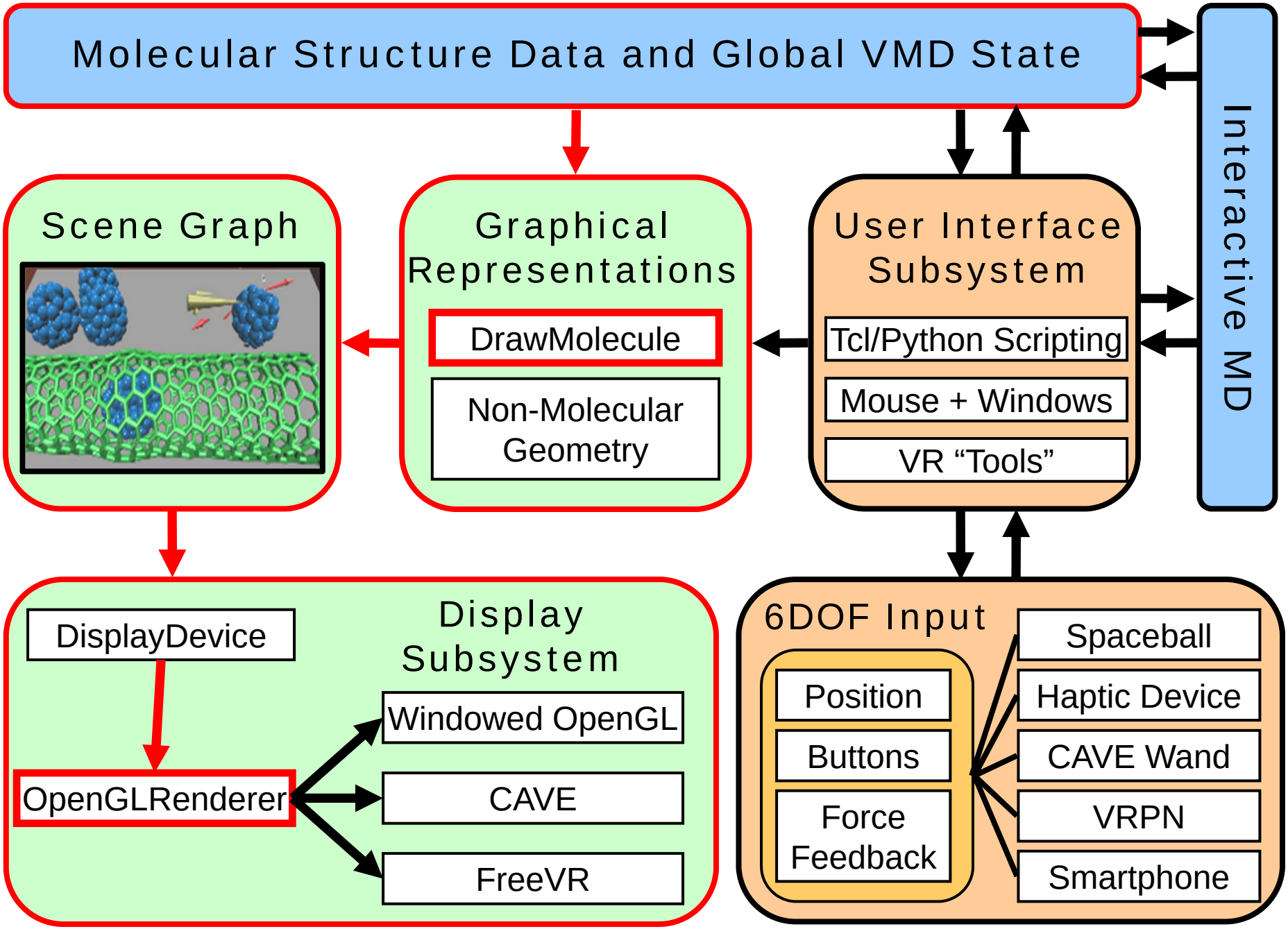
- Molecular dynamics trajectories store (at least) 12 bytes per atom, per timestep, for thousands to millions of timesteps
- 100M atom simulation stores **1.2GB per timestep!**
- Host CPU memory bandwidth is on the order ~10GB/sec, even rendering straight from RAM we **cannot** afford to traverse every atom during rendering
- Aggregate host memory bandwidth for all CPUs and PCIe controllers is less than ~20GB/sec
- Even with multithreading for I/O, computing scene graph, rendering to multiple GPUs, we must minimize data accesses, and **eliminate data copies** wherever possible

Challenges for Immersive Visualization of Dynamics of Large Structures

- Graphical representations re-generated for each simulation timestep:
  - Dependent on user-defined atom selections
- Although visualizations often focus on interesting regions of substructure, fast display updates require rapid traversal of molecular data structures
- Optimized per-frame atom selection traversal:
  - Increased performance of per-frame updates by ~10x for 116M atom BAR case with 200,000 selected atoms
- New GLSL point sprite sphere shader:
  - Reduce host-GPU bandwidth for displayed geometry
  - Over 20x faster than old GLSL spheres drawn using display lists — drawing time is now inconsequential
- Optimized all graphical representation generation routines for large atom counts, sparse selections

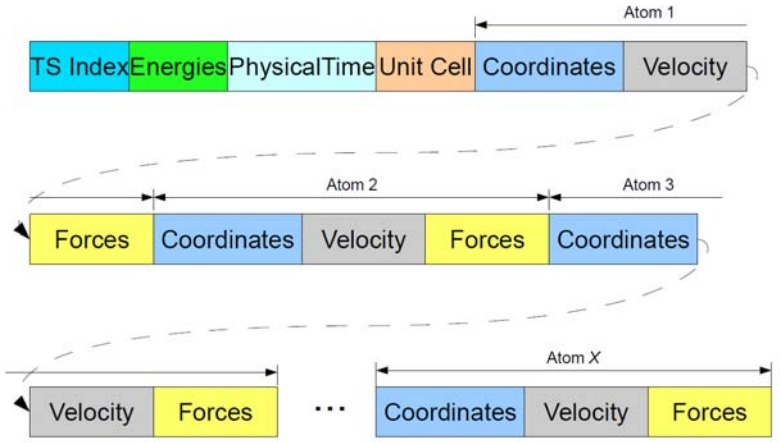


116M atom BAR domain test case:
200,000 selected atoms,
stereo trajectory animation 70 FPS,
static scene in stereo 116 FPS

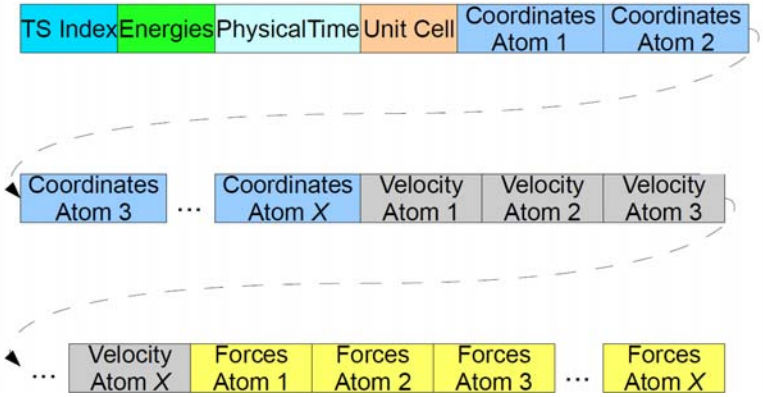

Molecular Structure Data and Global VMD State
Interactive MD
Scene Graph
Graphical Representations
DrawMolecule
Non-Molecular Geometry
User Interface Subsystem
Tcl/Python Scripting
Mouse + Windows
VR "Tools"
DisplayDevice
Display Subsystem
OpenGLRenderer
Windowed OpenGL
CAVE
FreeVR
6DOF Input
Position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

I/O Challenges for Out-of-Core Visualization

- Existing molecular dynamics trajectory file formats:
  - Not optimized for peak I/O performance
  - Sometimes haphazardly organized such that data fields may have to be transposed or reorganized on-the-fly by visualization tools
- Performance of magnetic disks is inadequate for smooth trajectory animation, except large RAID arrays, which are unwieldy, loud, and expensive, limiting their applicability
- Portable I/O APIs only achieve half of peak hardware performance on high-performance I/O devices

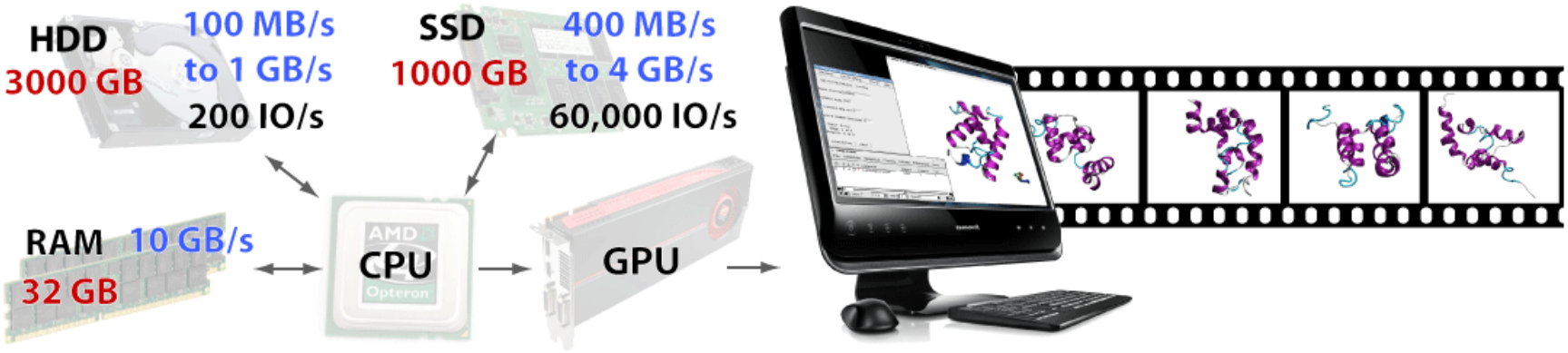


Atom data in "array of structures" (Bad)



Atom data in "structure of arrays" (Good)

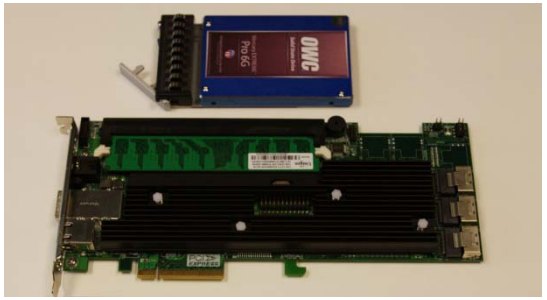
Performance of Solid State Disks vs. Magnetic Hard Drives



- SSDs offer sequential I/O rates 4x faster than high-end magnetic disks, and random I/O rates as high as 300x faster

Use of SSDs for High-Performance Molecular Dynamic Trajectory I/O

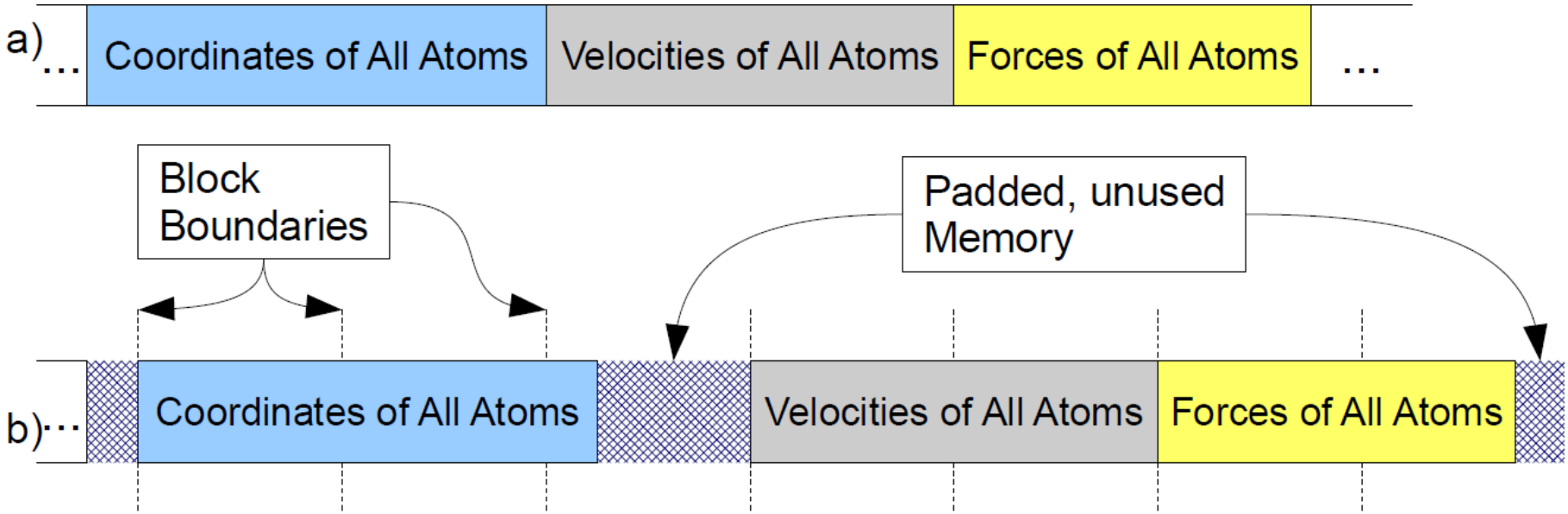
- A single SSD can achieve trajectory I/O rates that previously required a RAID array
- Well-suited for laptops
- A small SSD RAID array (~8 SSDs) can saturate a PCIe x8 RAID controller, delivering over 2GB/sec to application code, using direct I/O
- New PCIe-based SSDs achieve I/O rates similar to a RAID array, but with all components on a single PCIe card
- Using two RAIDs and doing parallel I/O with multiple threads, we have achieved I/O rates up to 4GB/sec in a test code



Buffered vs. Direct Operating System I/O APIs

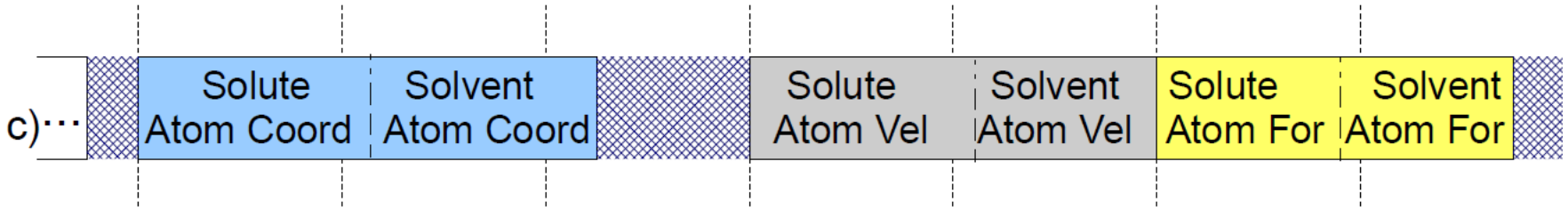
- Standard cross-platform C/C++ I/O APIs use “buffered” I/O:
  - OS reads disk blocks into kernel buffers, **then copies** into user destination buffers
  - Performance often half of what state-of-the-art storage hardware is capable of
  - During heavy I/O, aggressive kernel buffer allocation can cause paging of application data — a disaster for interactive rendering performance…
- Direct I/O benefits and complexities:
  - Direct I/O APIs read disk blocks straight to the user process destination buffer — a **zero copy approach** that conserves memory bandwidth and yields peak performance
  - Non-portable: different among Linux, MacOS X, and Windows, and minor differences between various Unix implementations
  - I/O size must be a multiple of the OS disk block- or VM page-size
  - File pointers and target memory buffers must always be aligned to block or page boundaries
  - Requires changes to both on-disk file formats and to application code

Trajectory File Format Changes for Direct I/O



Random Access I/O for Selective Loading of Trajectory Data

- Typical molecular simulations include many components that may not need to be displayed in typical cases (e.g. bulk solvent)
- SSDs provide very high random access I/O rates, allowing selective reads of only the atom data required for the current view, as determined by user's selections
- By skipping reads of just bulk solvent, we can often gain at least 2x performance, sometimes much more…



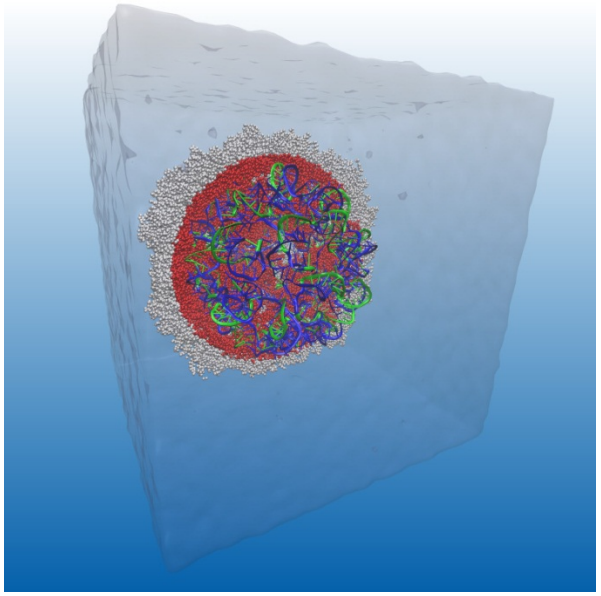
Single SSD Direct I/O Performance Results

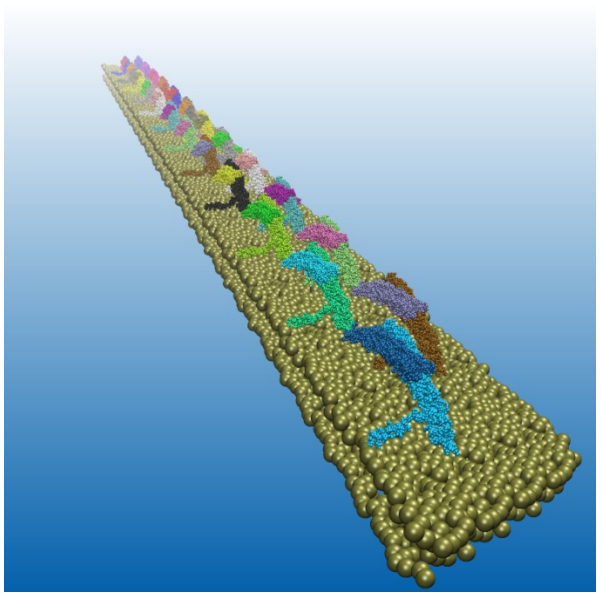
Hard-ware	Test Case	I/O Method	Atoms Loaded	Rate (TS/s)	Bandwidth (MB/s)	Speed vs. DCD		
						RAID	SSD	HD
HD	STMV	DCD, Normal	0.955M	9.3	102	0.14	0.39	1.0
	Ribosome	DCD, Normal	2.94M	3.0	105	0.12	0.38	1.0
SSD	STMV	DCD, Normal	0.955M	23.7	259	0.35	1.0	2.5
		OOC, Normal	0.955M	29.6	323	0.30	1.2	3.2
		OOC, Direct	0.955M	37.2	406	0.20	1.6	4.0
		OOC, Direct, NoSolv	0.178M	174.3	355	0.45	7.3	18.7
	Ribosome	DCD, Normal	2.94M	7.8	262	0.32	1.0	2.6
		OOC, Normal	2.94M	9.5	319	0.28	1.2	3.2
		OOC, Direct	2.94M	12.2	409	0.20	1.6	4.1
		OOC, Direct, NoSolv	1.55M	23.0	408	0.24	2.9	7.7

SSD RAID Direct I/O Performance Results

Hard-ware	Test Case	I/O Method	Atoms Loaded	Rate (TS/s)	Bandwidth (MB/s)	Speed vs. DCD		
						RAID	SSD	HD
RAID	STMV	DCD, Normal	0.955M	67	754	1.0	2.83	7.2
		OOC, Normal	0.955M	98	1,075	1.5	3.31	10.5
		OOC, Direct	0.955M	182	1,998	2.7	4.89	19.5
		OOC, Direct, NoSolv	0.178M	386	787	5.8	2.21	41.5
	Ribosome	DCD, Normal	2.94M	24.3	815	1.0	3.12	8.1
		OOC, Normal	2.94M	33.7	1,133	1.4	3.55	11.2
		OOC, Direct	2.94M	60.7	2,037	2.5	4.98	20.2
		OOC, Direct, NoSolv	1.55M	96.4	1,711	4.0	4.12	32.1
	Membrane	DCD, Normal	22.8M	3.0	781	1.0	-	-
		OOC, Normal	22.8M	4.6	1,207	1.5	-	-
		OOC, Direct	22.8M	8.0	2,087	2.6	-	-
		OOC, Direct, NoSolv	2.83M	46.5	1,508	15.5	-	-
	BAR	DCD, Normal	116M	0.6	708	1.0	-	-
		OOC, Normal	116M	0.9	1,189	1.5	-	-
		OOC, Direct	116M	1.6	2,130	2.6	-	-
		OOC, Direct, NoSolv	1.98M	76.3	1,725	127.2	-	-

Immersive Visualization Performance Results

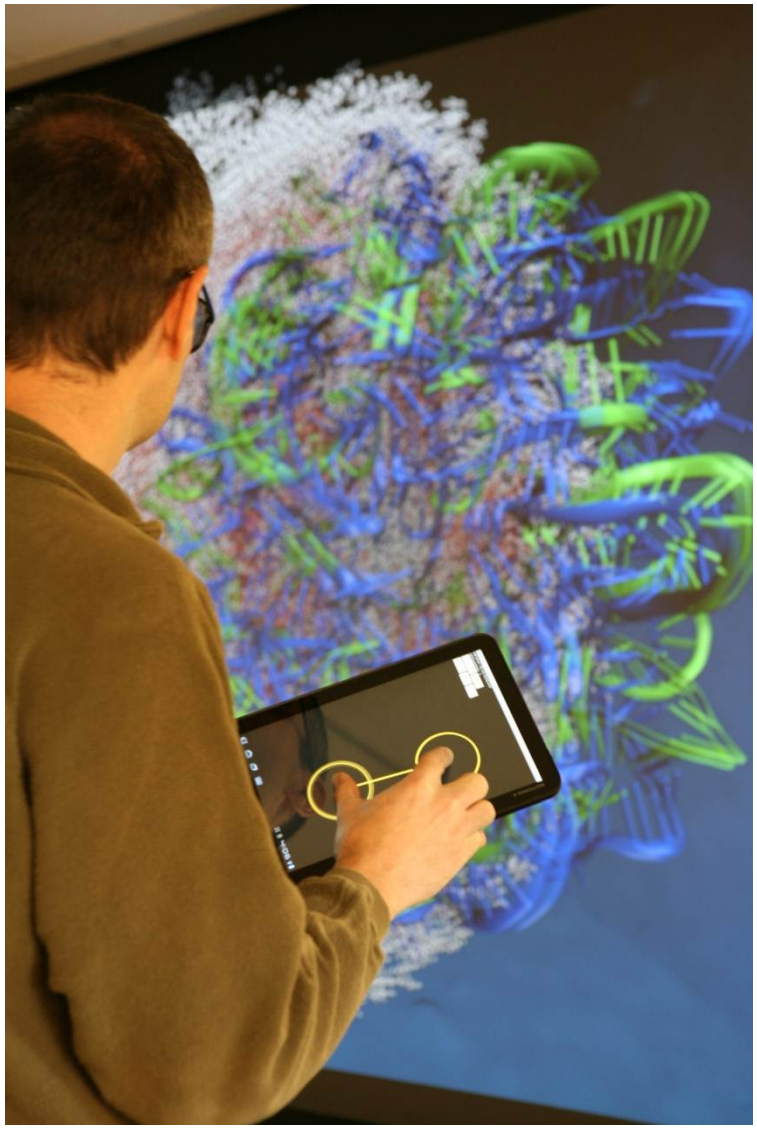




Test Case	Visualization Mode	Atoms Loaded	Display Rate (frames/s)
STMV	Stereo, static scene	0.955M	105
STMV	Stereo, in-core trajectory animation	0.955M	48
STMV	Stereo, out-of-core trajectory animation	0.955M	44
BAR domain	Stereo, static scene	116M	116
BAR domain	Stereo, in-core trajectory animation	116M	70
BAR domain	Stereo, out-of-core trajectory animation	1.98M	67

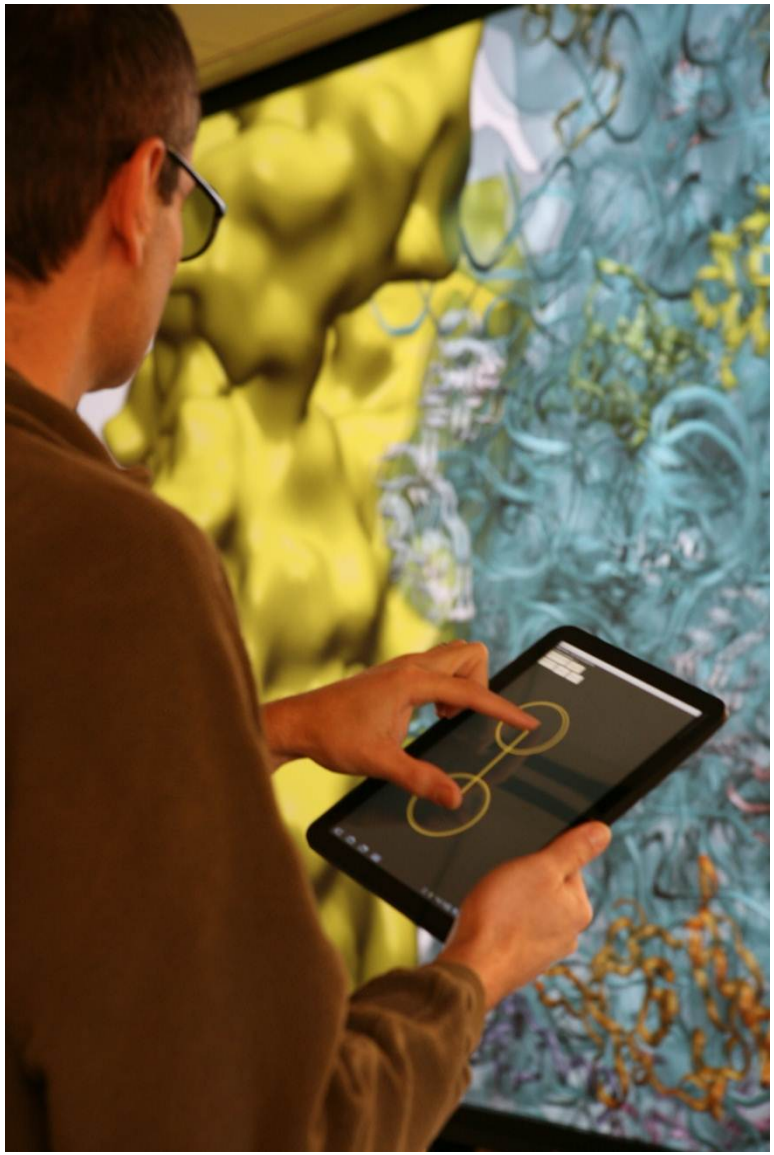
Summary

- Out-of-core performance better than 91% of in-core
- Results a combination of:
  - Improved trajectory rendering pipeline, fast GLSL shaders
  - Selective reads of atom data
  - Revised trajectory file format
  - Zero-copy direct I/O
  - SSD storage hardware



Future Work

- Extend “selective read” feature to finer granularity atom selections
- Trajectory file formats with “packed” blocks of frequently-needed data that is otherwie too sparse for the “selective read” approach to be successful
- Multi-level atom selection flag data structures
- Custom GLSL shaders for ribbon and surface representations
- Optimize data broadcasts for multi-GPU immersive systems



Acknowledgements

- Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign
- Beckman Institute
- NVIDIA CUDA Center of Excellence, University of Illinois at Urbana-Champaign
- The CUDA team at NVIDIA
- NIH support: P41-RR005969