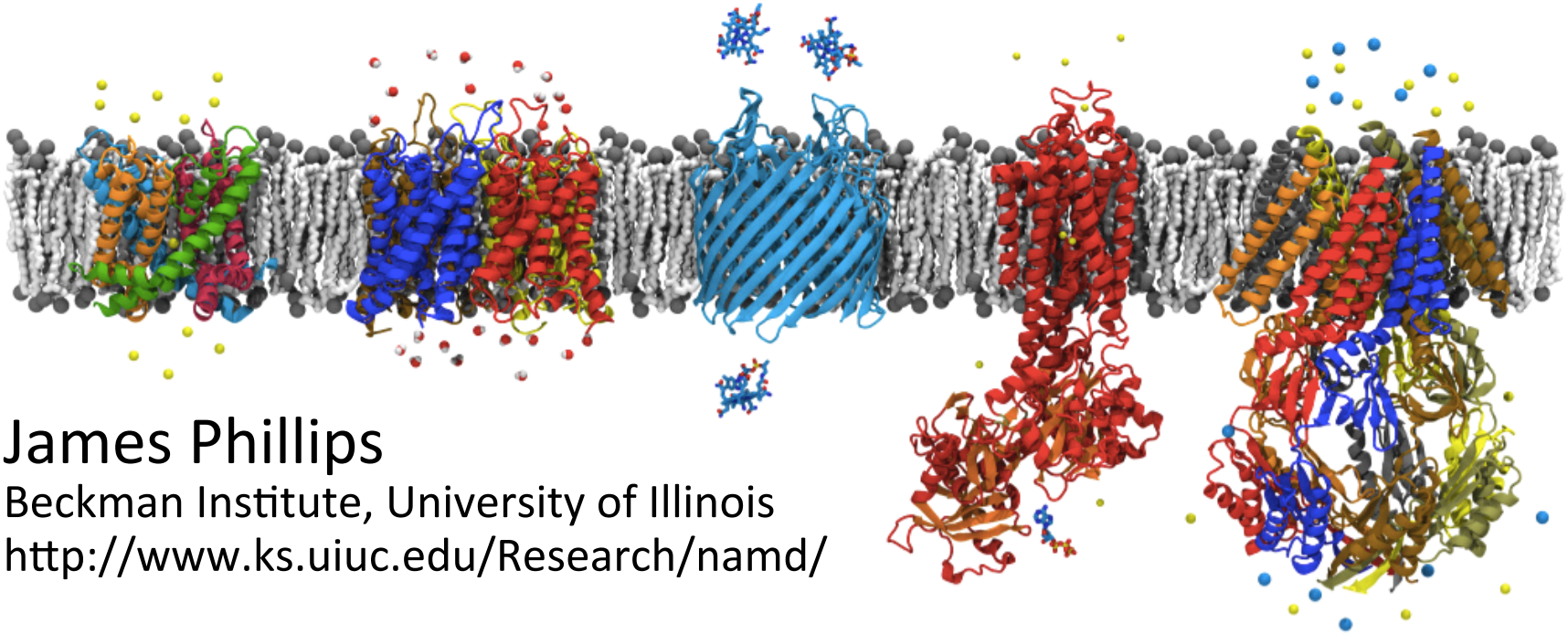
# Experiences with Charm++ and NAMD on Knights Landing Supercomputers

*15th Annual Workshop on Charm++ and its Applications*



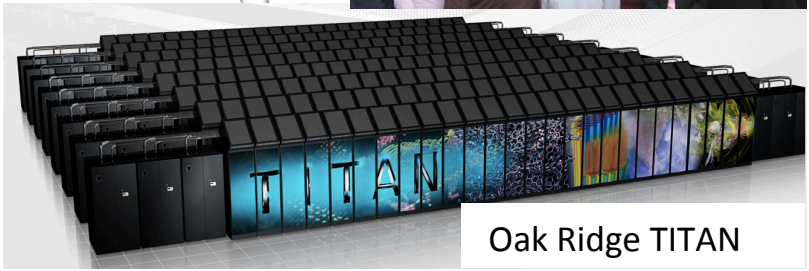
James Phillips
Beckman Institute, University of Illinois
http://www.ks.uiuc.edu/Research/namd/

# NAMD Mission Statement:

## *Practical Supercomputing for Biomedical Research*

- 88,000 users can't all be computer experts.
  - 18% are NIH-funded; many in other countries.
  - 26,000 have downloaded more than one version.
  - 6,000 citations of NAMD reference papers.
  - 1,000 users per month download latest release.
- One program available on all platforms.
  - Desktops and laptops – setup and testing
  - Linux clusters – affordable local workhorses
  - Supercomputers – "most used code" at XSEDE TACC
  - Petascale – "widest-used application" on Blue Waters
  - GPUs – from desktop to supercomputer
- User knowledge is preserved across platforms.
  - No change in input or output files.
  - Run any simulation on **any number of cores.**
- Available free of charge to all.

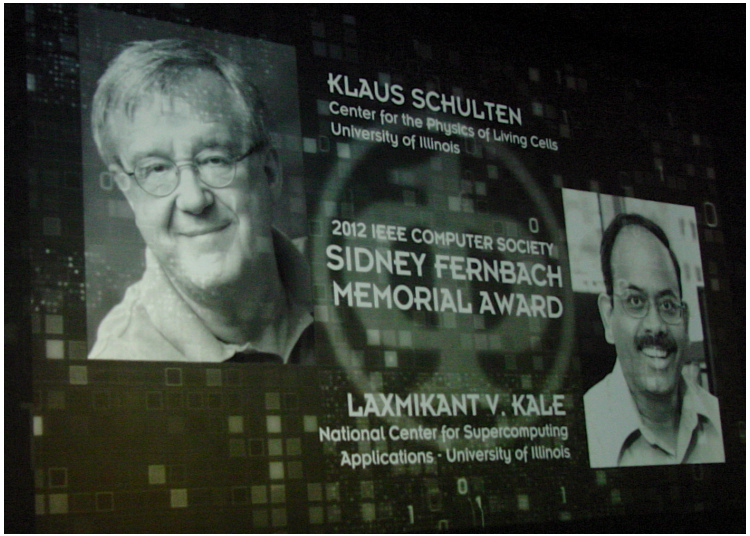

Hands-On Workshops




Oak Ridge TITAN

# Computing research drives NAMD (and vice-versa)

- Parallel Programming Lab – (co-PI Kale)
  - Charm++ parallel runtime system
  - Gordon Bell Prize 2002
  - IEEE Fernbach Award 2012
  - 16 publications SC 2012-16
  - 6+ codes on Blue Waters
- Support from Intel, NVIDIA, IBM, Cray
- 20 years of co-design for NAMD
  - Performance, portability, productivity
  - SC12: Customized Cray network layer
  - SC14: Cray network topology optimization
  - Parallelization of Collective Variables module

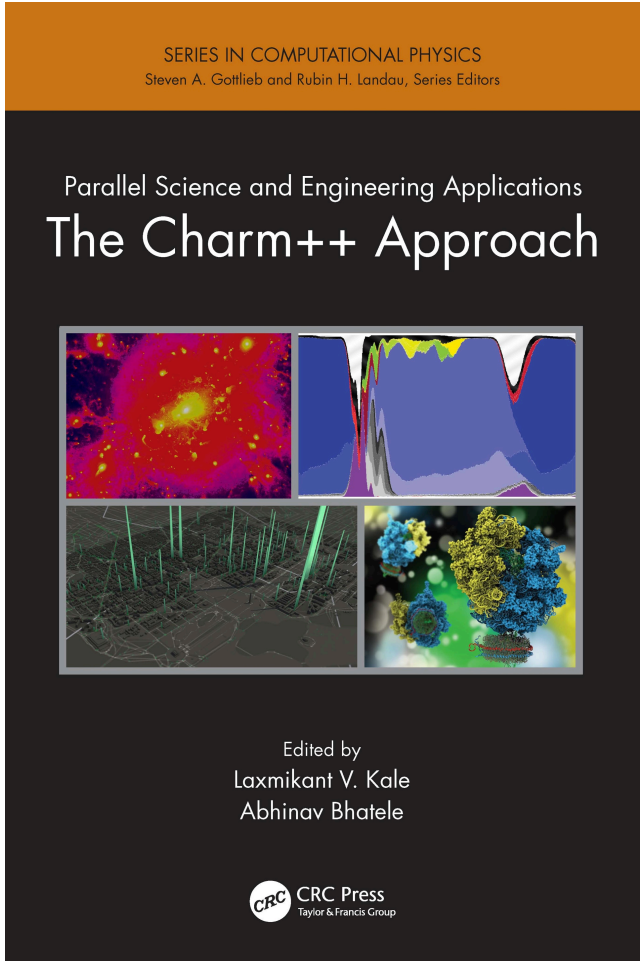



*"For outstanding contributions to the development of widely used parallel software for large biomolecular systems simulation"*

# Charm++ Used by NAMD

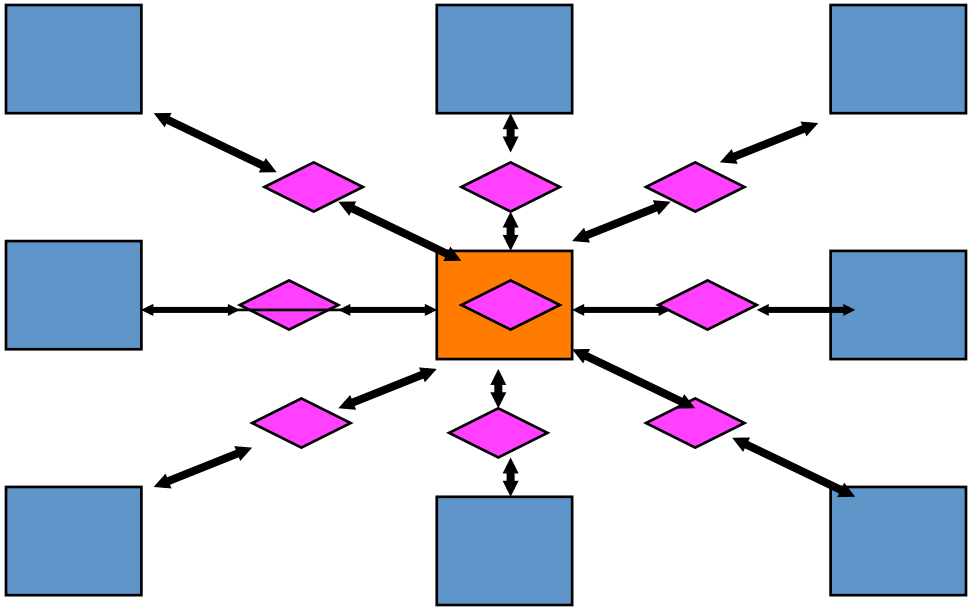
- Parallel C++ with *data driven* objects.
- Asynchronous method invocation.
- Prioritized scheduling of messages/execution.
- Measurement-based load balancing.
- Portable messaging layer.

**Complete info at charmplusplus.org and charm.cs.illinois.edu**



# NAMD Hybrid Decomposition

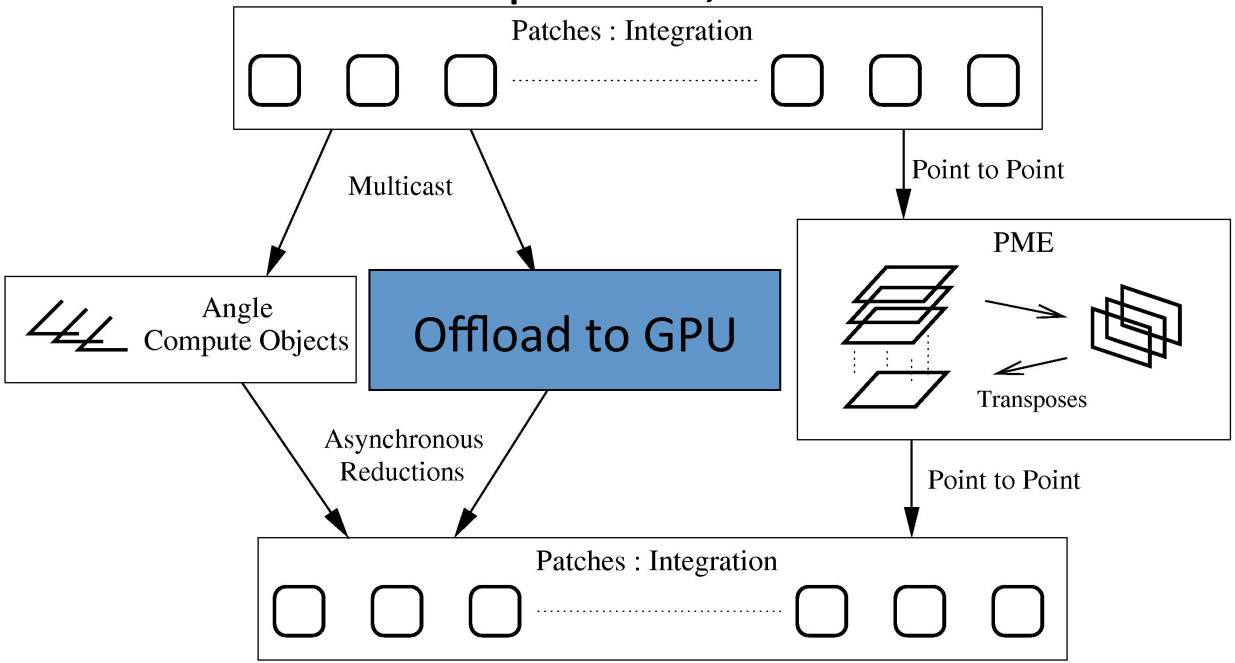
Kale *et al., J. Comp. Phys.* 151:283-312, 1999.



- Spatially decompose data and communication.
- Separate but related work decomposition.
- "Compute objects" facilitate iterative, measurement-based load balancing system.

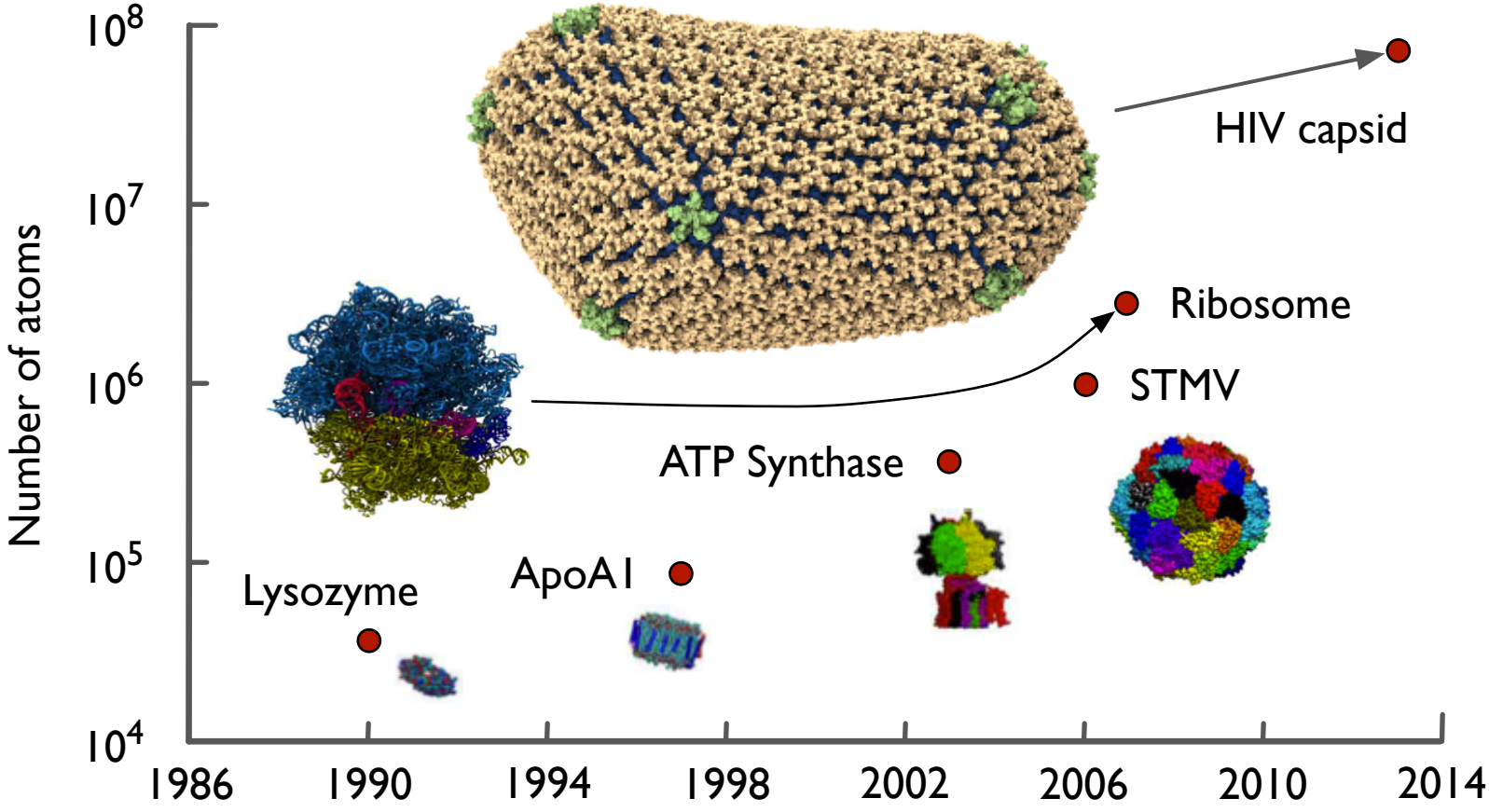
# NAMD Overlapping Execution

Phillips *et al., SC2002*.

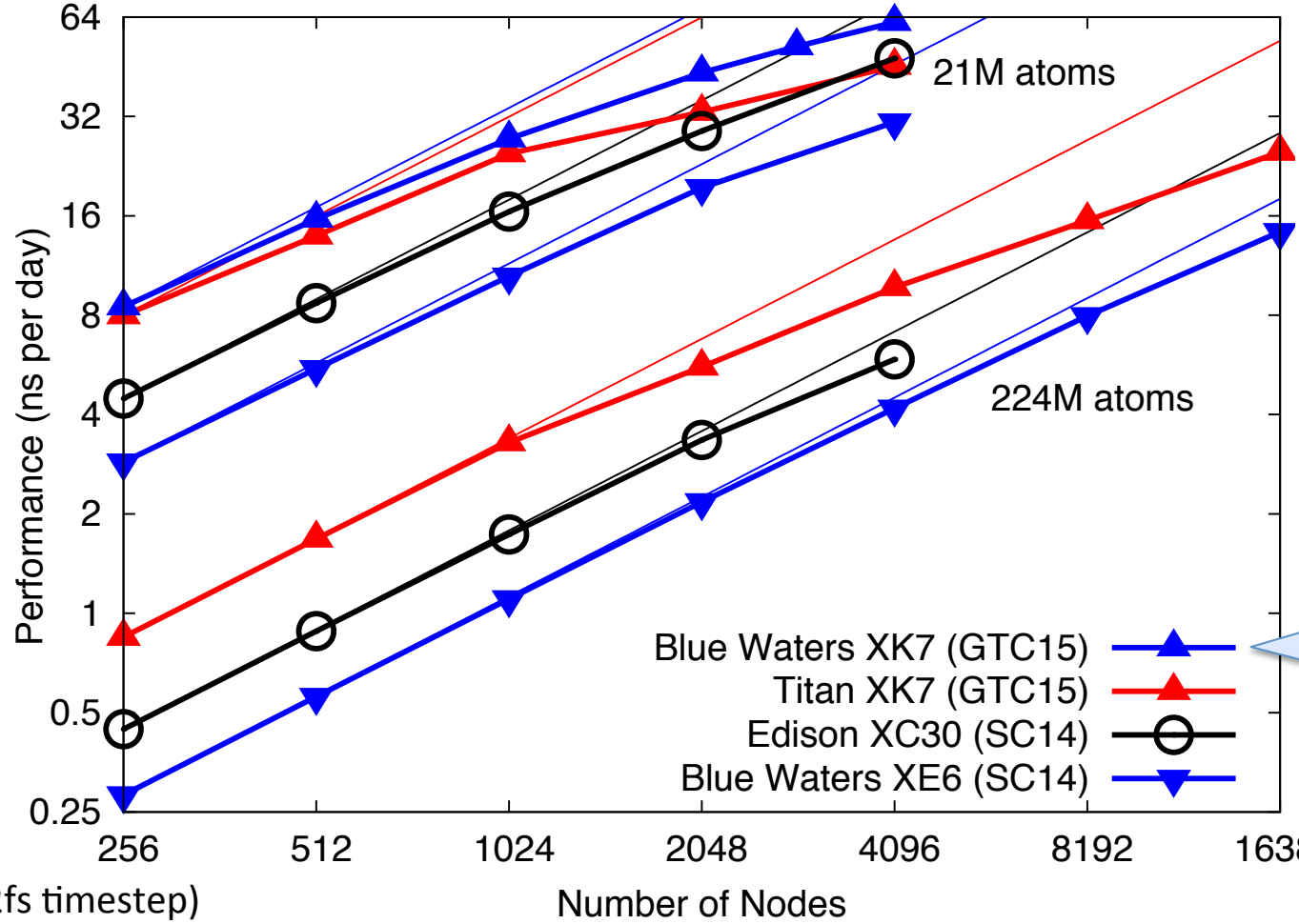


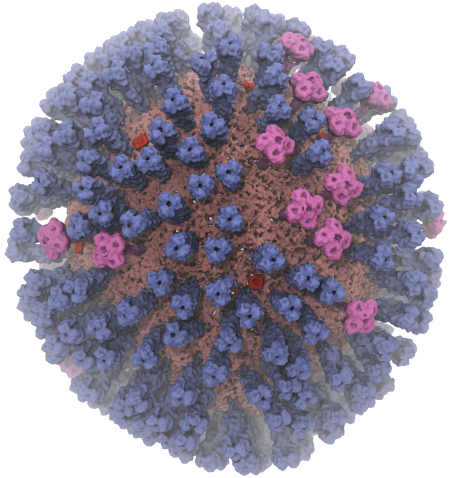

Objects are assigned to processors and queued as data arrives.

# A brief history of NAMD (and VMD)



# NAMD Runs Large Petascale Simulations Well

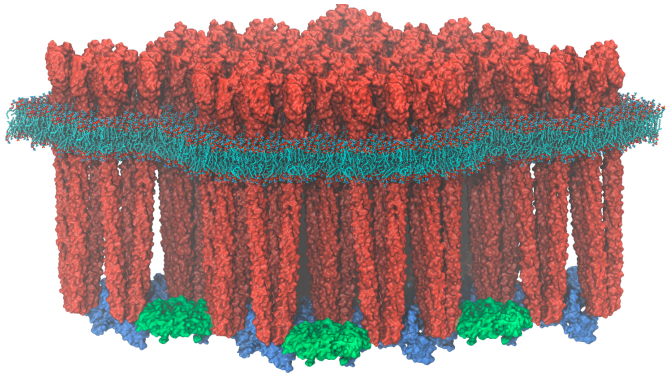


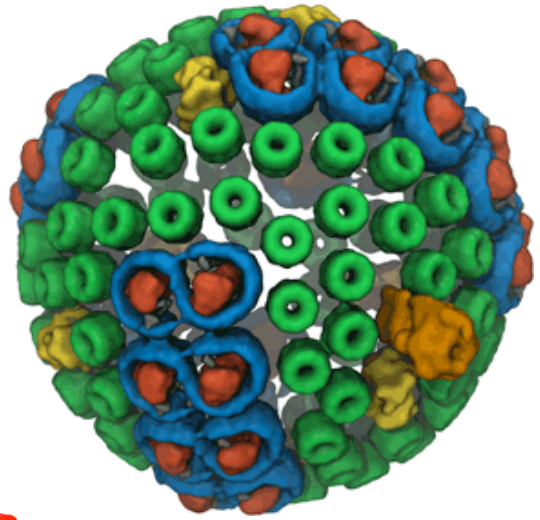
Influenza, 210M atoms
Amaro Lab, UCSD

Topology-aware scheduler

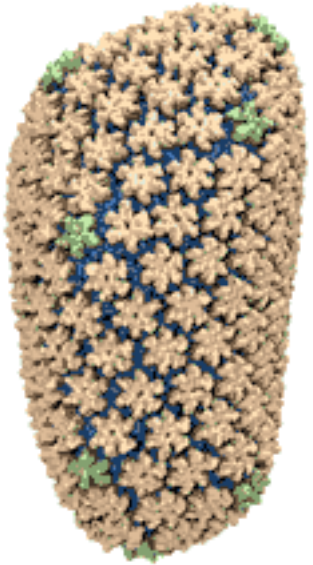
# A Sampling of Petascale Projects Using NAMD



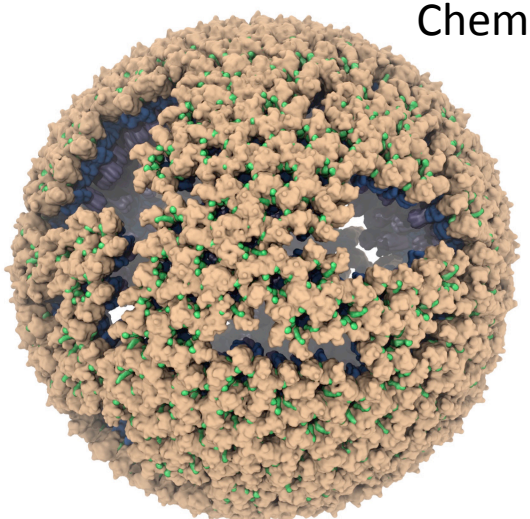
Chemosensory Array



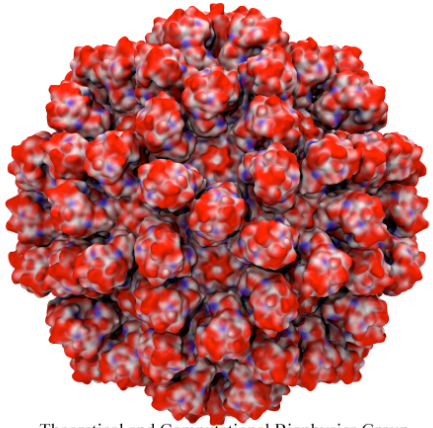
Chromatophore



HIV



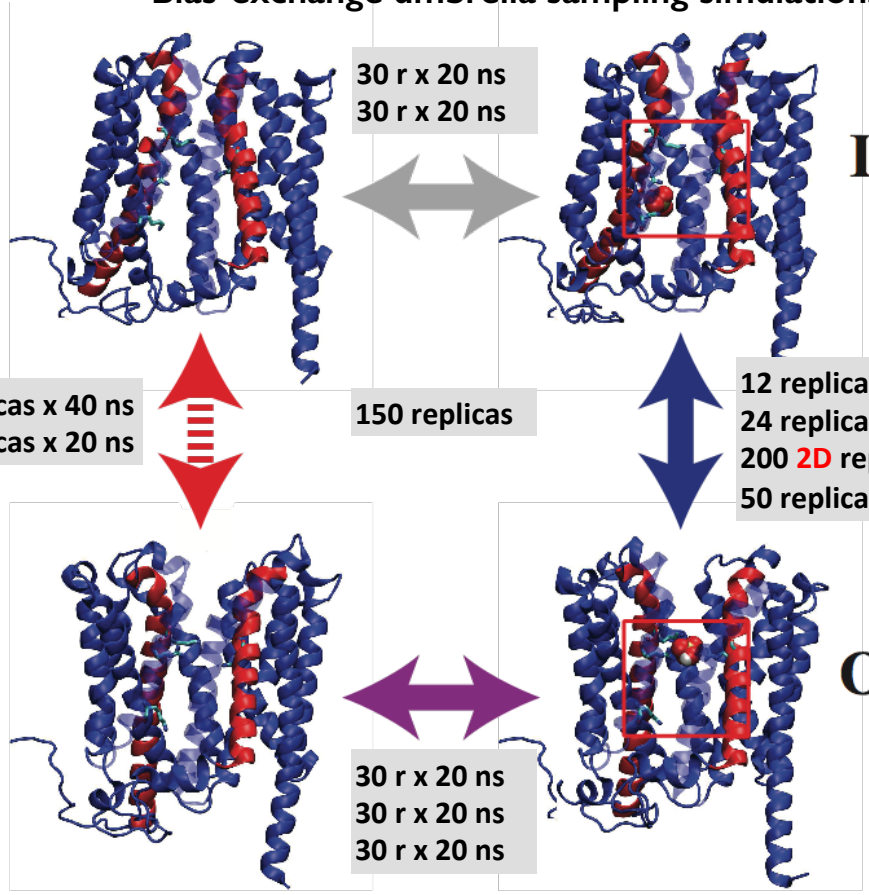
Rous Sarcoma Virus

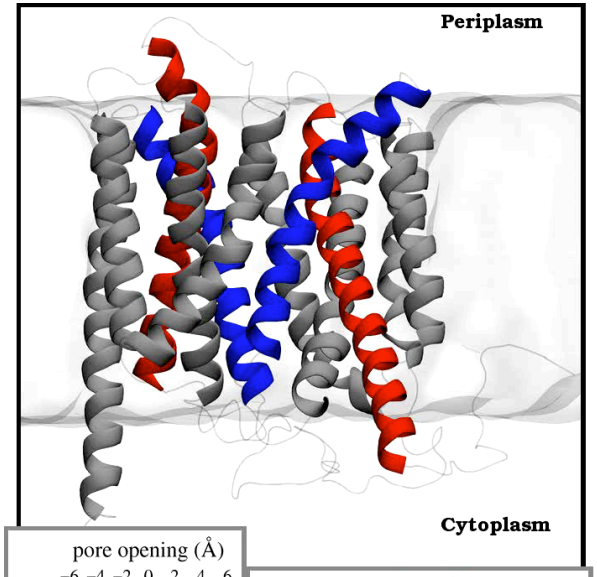


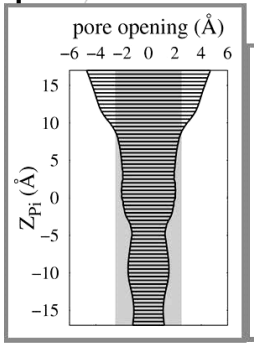
Rabbit Hemorrhagic Disease

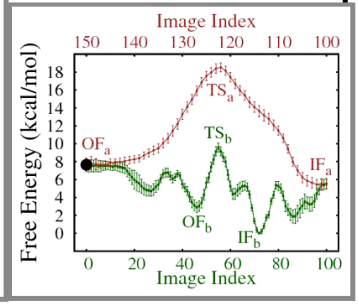
# New multi-copy methodologies enable study of millisecond processes

**Bias-exchange umbrella sampling simulations of GlpT membrane transporters**



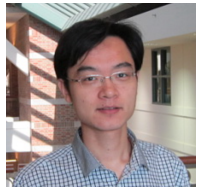



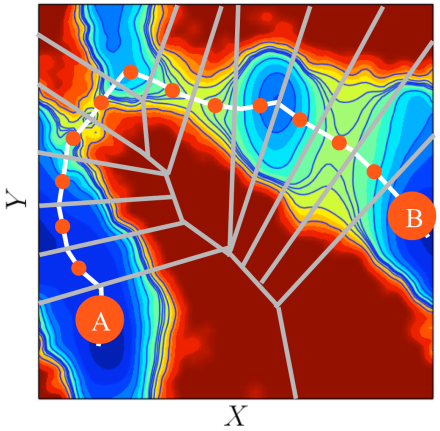

M. Moradi, G. Enkavi, and E. Tajkhorshid, *Nature Communications* **6**, 8393 (2015)

# Coming Soon: Milestoning

***Portable innovation*** implemented in Tcl and Colvars scripts by graduate student

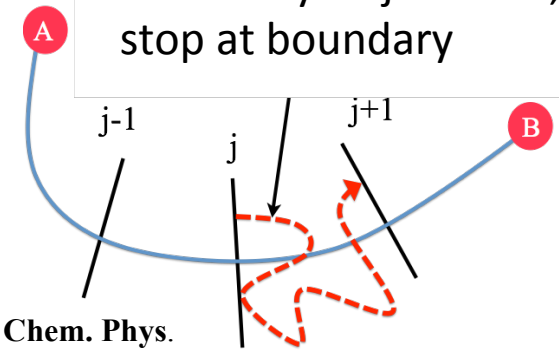


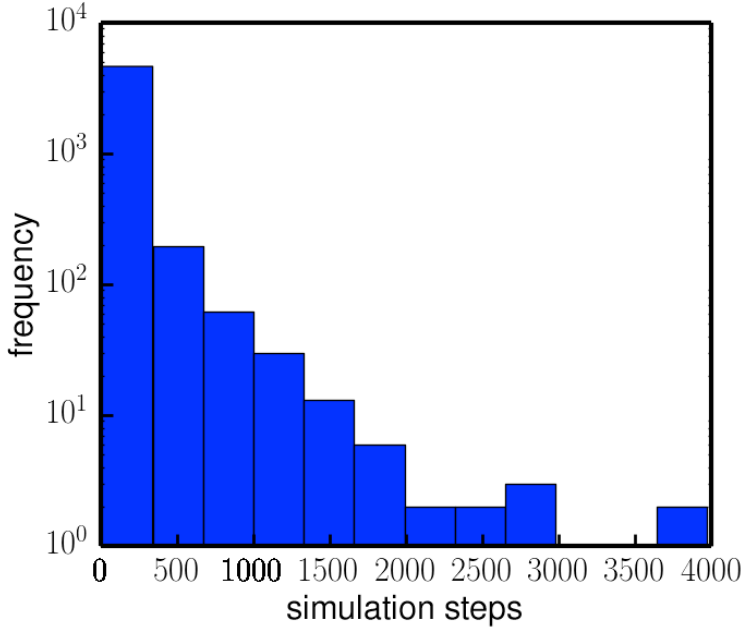
Wen Ma




Use string method to identify low-energy transition path and partition space into Voronoi polygons

Run many trajectories, stop at boundary



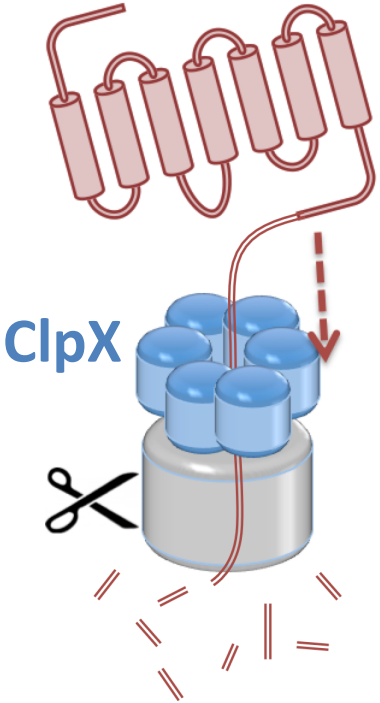




NAMD 2.11 work queue efficiently handles randomly varying run lengths across multiple replicas in same run

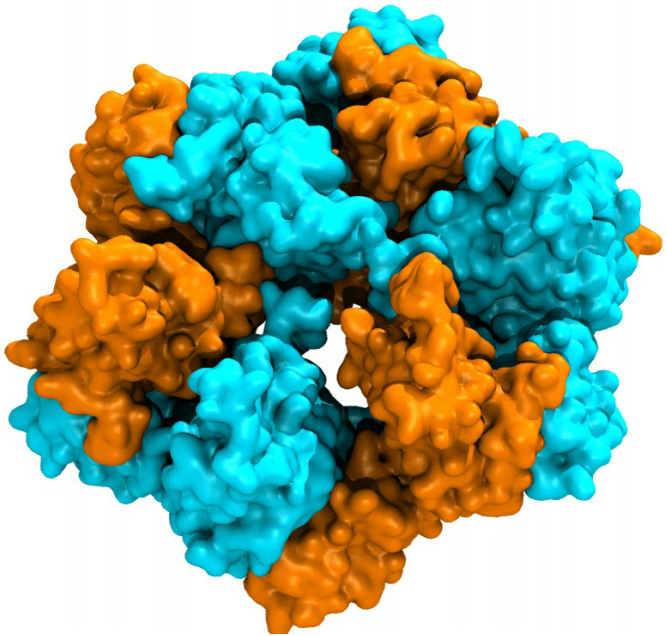

Faradjian and Elber, 2004. **J. Chem. Phys**.
Bello-Rivas and Elber, 2015, **J. Chem. Phys**

# Milestoning Applied to Molecular Motors

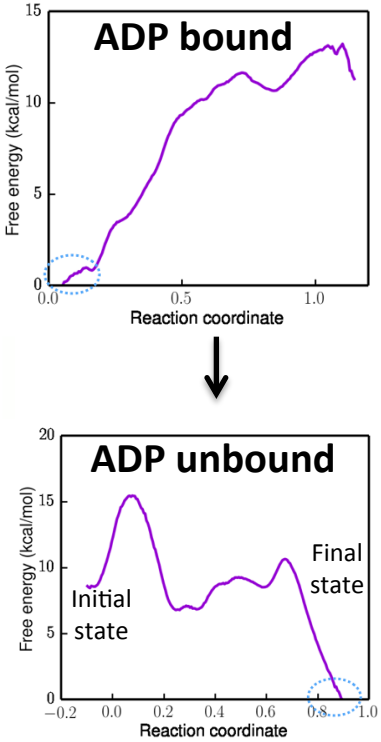
*TACC Stampede KNL Early Science Project*




Experimental collaborator:
A. Martin, UC Berkeley



ClpX powerstroke transition
Predicted time scale: 0.5 ms




**ADP release** shifts global minimum, leading to motor action

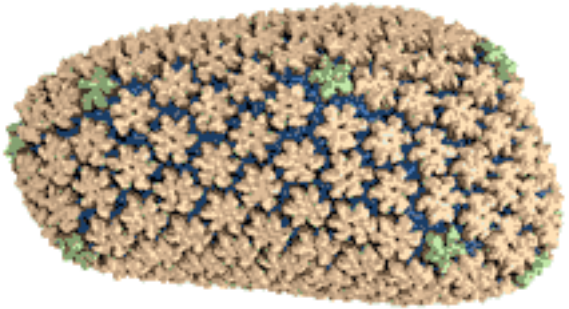
Ma and Schulten, JACS (2015); Singharoy and Schulten, submitted

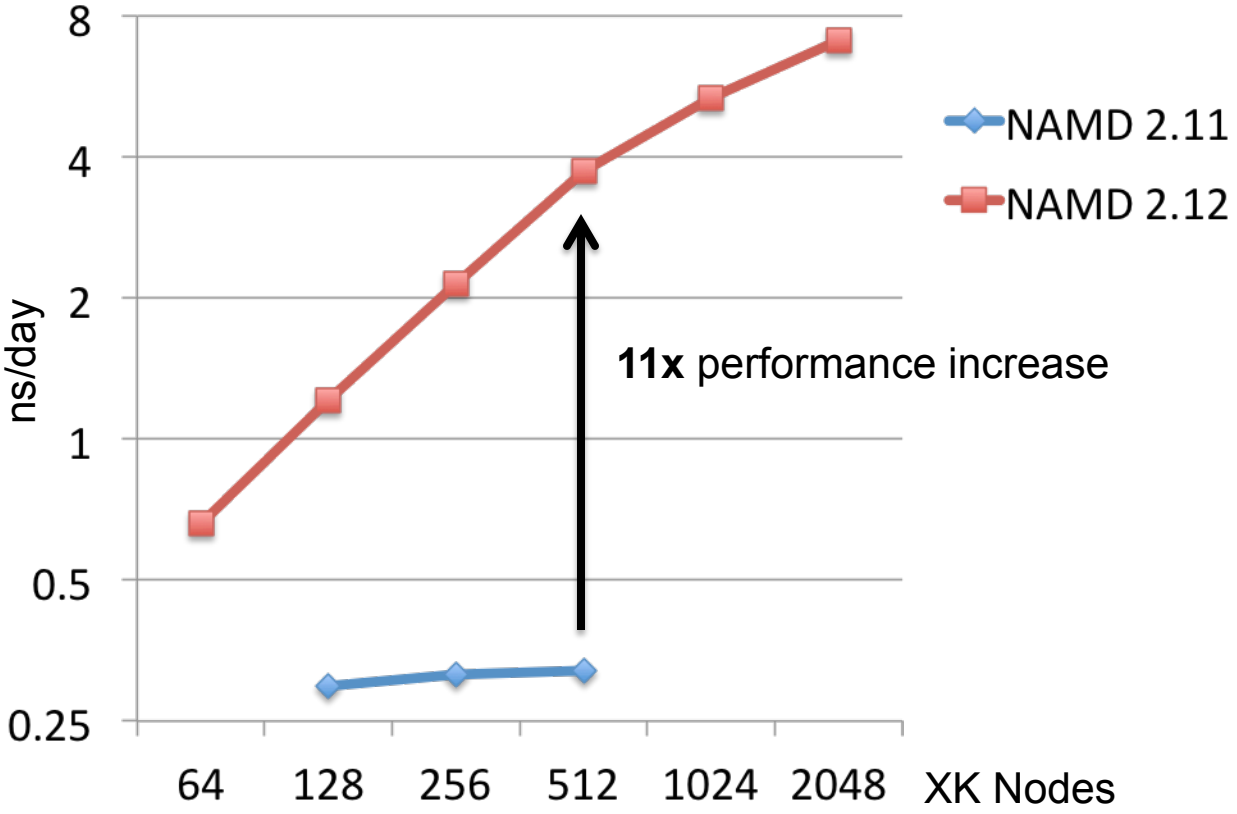
# NAMD 2.12 Release

- Final release December 22, 2017
- Capabilities:
  - **New QM/MM interface to ORCA, MOPAC, etc.**
  - Alchemical free energy calculation enhancements for constant pH
  - Efficiently reload molecular structure at runtime for constant pH
  - Grid force switching and scaling for MDFF and membrane sculpting
  - Python scripting interface for advanced analysis and feedback
- Performance:
  - **New GPU kernels up to three times as fast** (esp. implicit solvent)
  - Improved vectorization and new KNL processor kernels
  - Improved scaling for large implicit solvent simulations
  - Improved scaling with many collective variables
  - Improved GPU-accelerated replica exchange
  - Enhanced support for replica MDFF on cloud platforms

# NAMD 2.12 Large Implicit Solvent Models

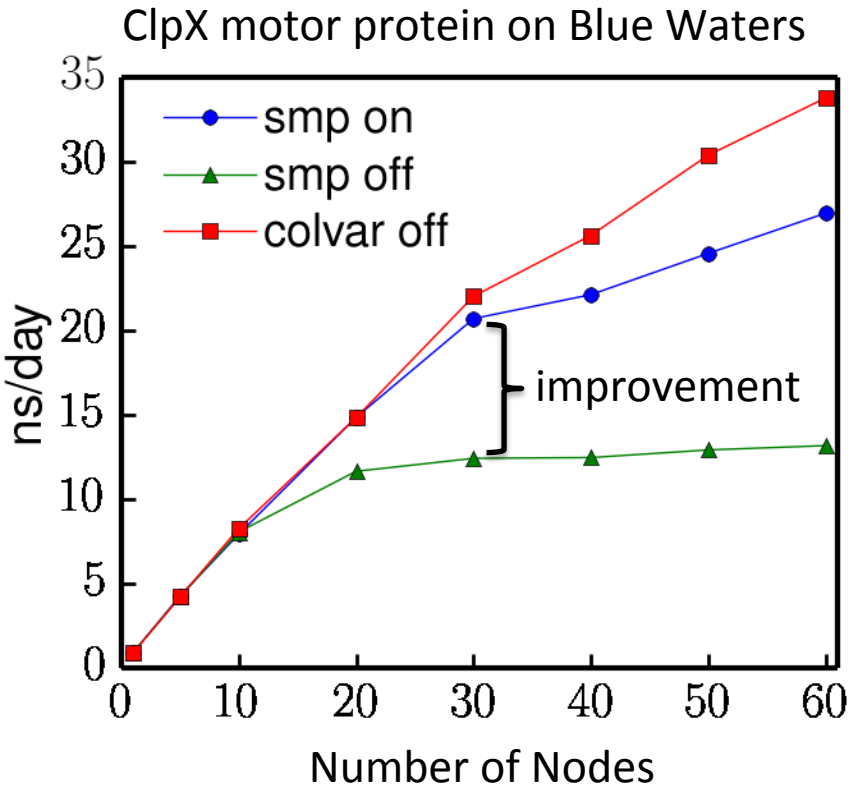
NAMD 2.12 (Dec 2016) provides order-of-magnitude performance increase for 5.7M-atom implicit solvent HIV capsid simulation on GPU-accelerated XK nodes.



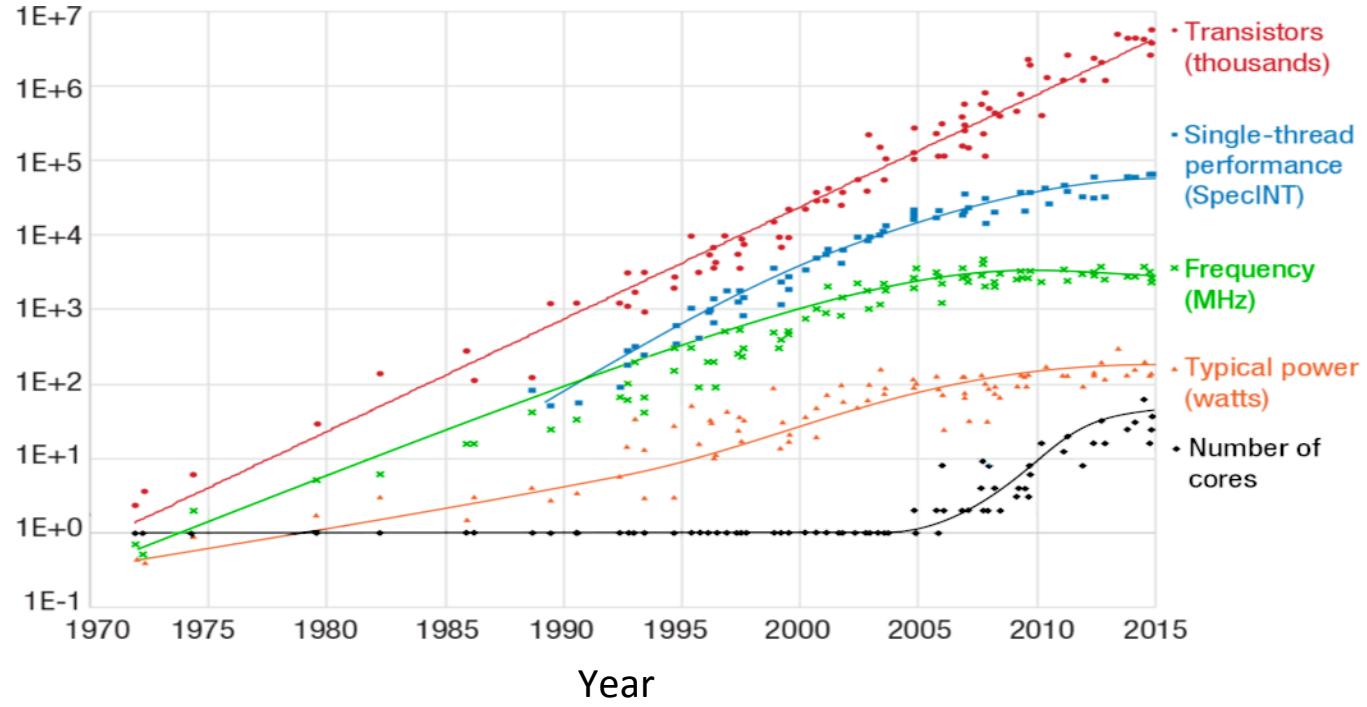


# Collective variables parallelization

- Colvars (Fiorin, Henin) provides flexible, hierarchical steering and free energy analysis methods
- Parallel bottleneck as complexity of user-defined variables increases (e.g., multiple RMSDs)
- Charm++ “smp” shared memory build restores scalability via CkLoop parallelization
- Released in NAMD 2.12



# Hardware trends challenge software developers




Source: Kirk M. Bresniker, Sharad Singhal, R. Stanley Williams, "Adapting to Thrive in a New Economy of Memory Abundance", Computer vol. 48 no. 12, p. 44-53, Dec., 2015

**Moore's Law has stayed alive, transistor count keeps climbing (and likely will for next ~5 years)**

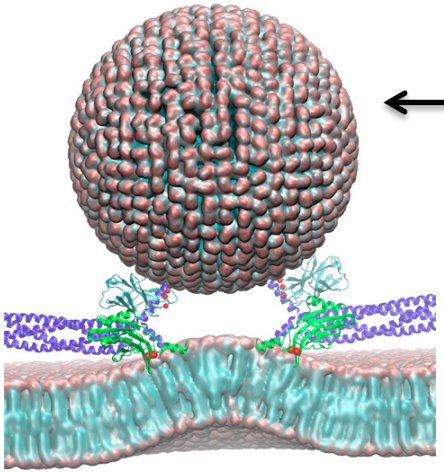
**But single thread performance from frequency has stalled**

**Due to power limits**

**Instead, core counts have been increasing**

# New Platforms Require Multi-Year Preparation

**Fall 2016: Argonne "Theta" and NERSC "Cori" Intel Xeon Phi KNL**
Argonne Early Science: Membrane Transporters (with Benoit Roux)
Technical Assistance: Brian Radak, Argonne
User Benefits: KNL port, multi-copy enhanced sampling, constant pH



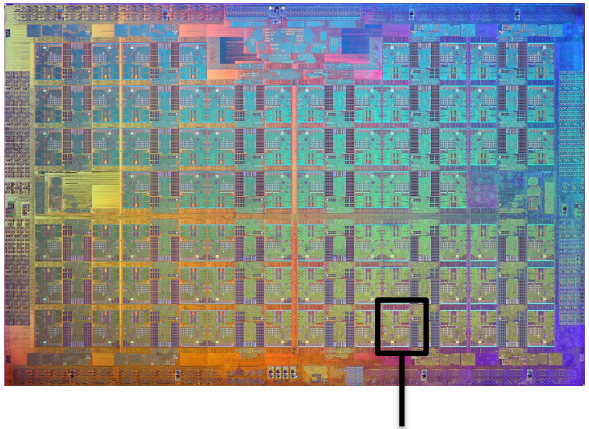
Synaptic vesicle and presynaptic membrane

**2018: Oak Ridge "Summit" 200PF Power9 + Volta GPU**
Early Science: "Molecular Machinery of the Brain"
Performance Target: 200 ns/day for 200M atoms
Technical Assistance: Antti-Pekka Hynninen, Oak Ridge/NVIDIA
User Benefit: GPU performance in NAMD 2.11, 2.12

**2019: Argonne "Aurora" 200PF Intel Xeon Phi KNH**
Early Science: Membrane Transporters
PIs Roux, Tajkhorshid, Kale, Phillips

# Intel Xeon Phi KNL processor port

- Intel's alternative to GPU computing:
  - 64-72 low-power/low-clock CPU cores
  - 4 threads per core – 256-way parallelism
  - 16-wide (single precision) vector instructions
- Three installations:
  - Argonne Theta, NERSC Cori: Cray network
  - TACC Stampede 2: Intel Omni-Path network
- Challenges addressed:
  - Greater use of Charm++ shared-memory parallelism
  - New vectorizable kernels developed with Intel assistance
  - New Charm++ network layer for Omni-Path in progress



# AVX-512 Optimizations

- New kernels, optimizations guided by Intel
  - icpc -DNAMD_KNL -xMIC-AVX512
  - __assume_aligned(…,64);
  - #pragma simd assert reduction(+:…)
  - Single-precision calculation, double accumlation
  - Linear electrostatic interpolation (similar to CUDA)
  - Explicit vdW (switched Lennard-Jones) calculation
  - Fall back to old kernels for exclusions, alchemy, etc.

# AVX-512 Gather Optimization

```
 float p_j_x, p_j_y, p_j_z, x2, y2, z2, r2;
#pragma vector aligned
#pragma ivdep
  for ( g = 0 ; g < list_size; ++g ) {
    int gi=list[g];  // indices must be 32-bit int to enable gather instructions
    p_j_x = p_j[ gi ].position.x;    p_j_y = p_j[ gi ].position.y;    p_j_z = p_j[ gi ].position.z;

    x2 = p_i_x - p_j_x;    r2 = x2 * x2;
    y2 = p_i_y - p_j_y;    r2 += y2 * y2;
    z2 = p_i_z - p_j_z;    r2 += z2 * z2;

    if ( r2 <= cutoff2 ) {  // cache gathered data in compact arrays
      *nli = gi;  ++nli;      *r2i = r2;  ++r2i;
      *xli = x2;  ++xli;      *yli = y2;  ++yli;      *zli = z2;  ++zli;
    }
  }
```

# KNL Memory Modes

- 16 GB MCDRAM high-bandwidth memory
  - also at least 96 GB of regular DRAM
- Flat mode: exposed as NUMA domain 1
  - numactl --membind=1 or --preferred=1
- Cache mode: used as direct-mapped cache
  - Performs similar to flat mode most of the time
  - Potential for thrashing when addresses randomly conflict
- Hybrid mode: 4GB or 8GB used as cache
- When in doubt, “cache-quadrant” mode
  - If less than 16GB required, “flat-quadrant” + “numactl –m 1”
  - No observed benefit from SNC (sub-NUMA cluster) modes

# Charm++ Build Options

- Choose network layer:
  - multicore (smp but only a single process, no network)
  - netlrts (supports multi-copy) or net (deprecated)
  - gni-crayx[ce] (Cray Gemini or Aries network)
  - verbs (supports multi-copy) or net-ibverbs (deprecated)
  - mpi (fall back to MPI library, use for Omni-Path)
- Choose smp or (default) non-smp:
  - smp uses one core per process for communication
- Optional compiler options:
  - iccstatic uses Intel compiler, links Intel-provided libraries statically.
  - Also: --no-build-shared --with-production

# General NAMD/Charm++ Tips

- DO NOT use the MPI network layer (except on OmniPath for now)
  - Low-level verbs, gni, pami layers exist because they are faster
  - Leverage MPI startup via “charmrun ++mpiexec”
  - See also ++scalable-start, ++remote-shell, ++runscript
- DO use SMP builds for larger simulations
  - Reduced memory usage and often faster
  - Trade-off: communication thread not available for work
  - Major direction of future optimization and tuning
- DO set processor affinity explicitly
  - For example: ++ppn 7 +commap 0,8 +pemap 1-7,9-15
  - Cray by default tends to lock all threads onto same core
- DO save one core for OS to improve scaling
  - Cray “aprun –r 1” reserves and forces OS to run on last core

**ALCF Theta Build and Run Options**

- 64-core processors, Cray Aries network
- build charm++ gni-crayxc persistent smp -xMIC-AVX512
- aprun -n $((7*$nodes)) -N 7 -d 17 -j 2 -r 1
- +ppn 16 +pemap 0-55+64 +commap 56-62
  - or +ppn 8 +pemap 0-55 +commap 56-62

**TACC Stampede KNL Build and Run Options**

- 68-core processors, Intel Omni-Path network
- build mpi-linux-x86_64 smp icc -xMIC-AVX512
- sbatch --ntasks=$((13*$nodes))
- +ppn 8 +pemap 0-51+68 +commap 53-65
  - or +ppn 4 +pemap 0-51 +commap 53-65

# KNL Run Option Reasoning

- Leave core free to isolate OS noise
- Pairs of cores on a “tile” share 1MB L2 cache
  - Do not split tile between PEs of different nodes
  - OK to split tile between comm threads
- Use 1 or 2 hyperthreads for PE cores
  - Dedicate core to each comm thread
- Need several comm threads per host
  - Fewer for Cray Aries and than for Intel Omni-Path
  - Multiple copies of static data reduce memory contention
- Different configurations fit 64-core vs 68-core models

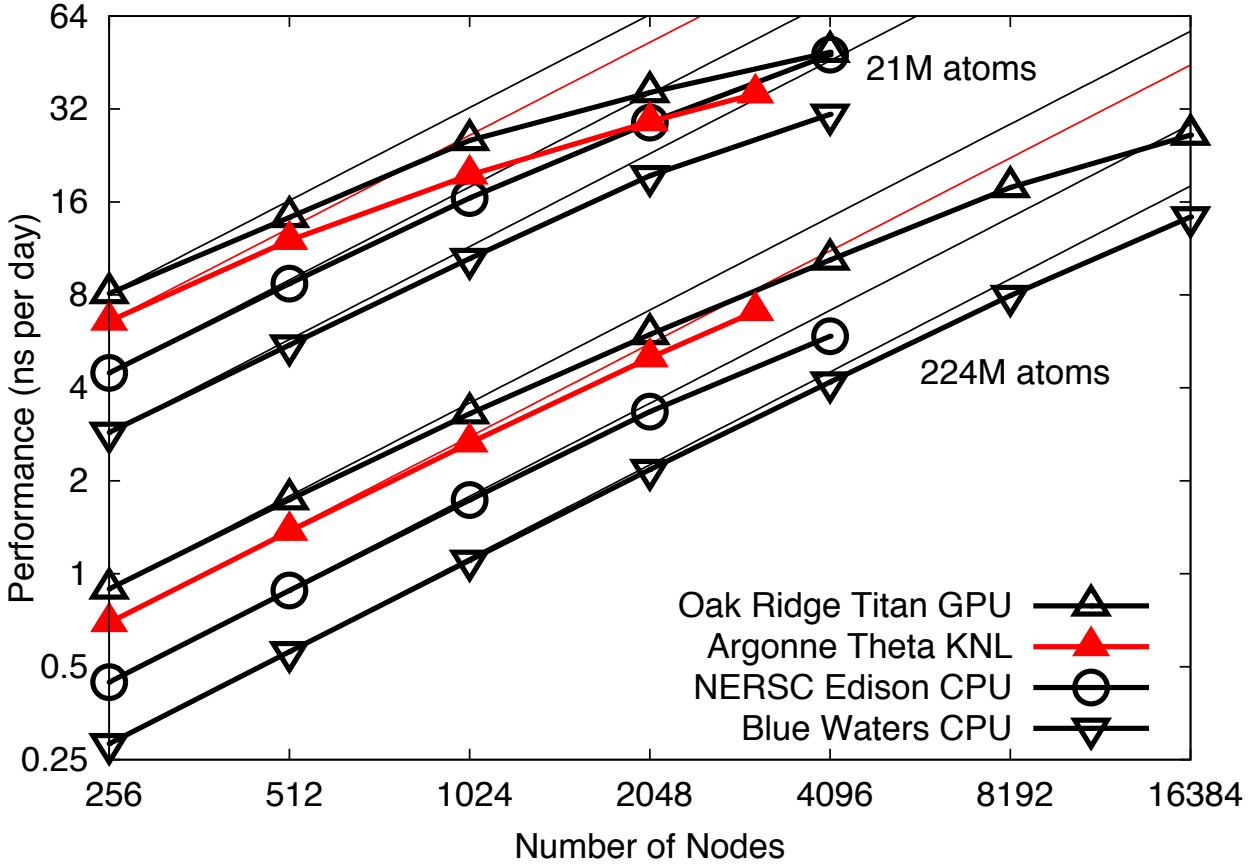
## ALCF Theta Run Option Math

- 64 cores, reserve one for OS (-r 1), leaves 63
- 63 = 9*7 = 9*(6+1) = 54 PE + 9 comm  +ppn 12 +pemap 0-53+64 +commap 54-62
- 63 = 7*9 = 7*(8+1) = 56 PE + 7 comm  +ppn 16 +pemap 0-55+64 +commap 56-62
- 60 = 4*15 = 4*(14+1) = 56 PE + 4 comm  +ppn 28 +pemap 0-63:16.14+64 +commap 14-62:16

## TACC Stampede KNL Run Option Math

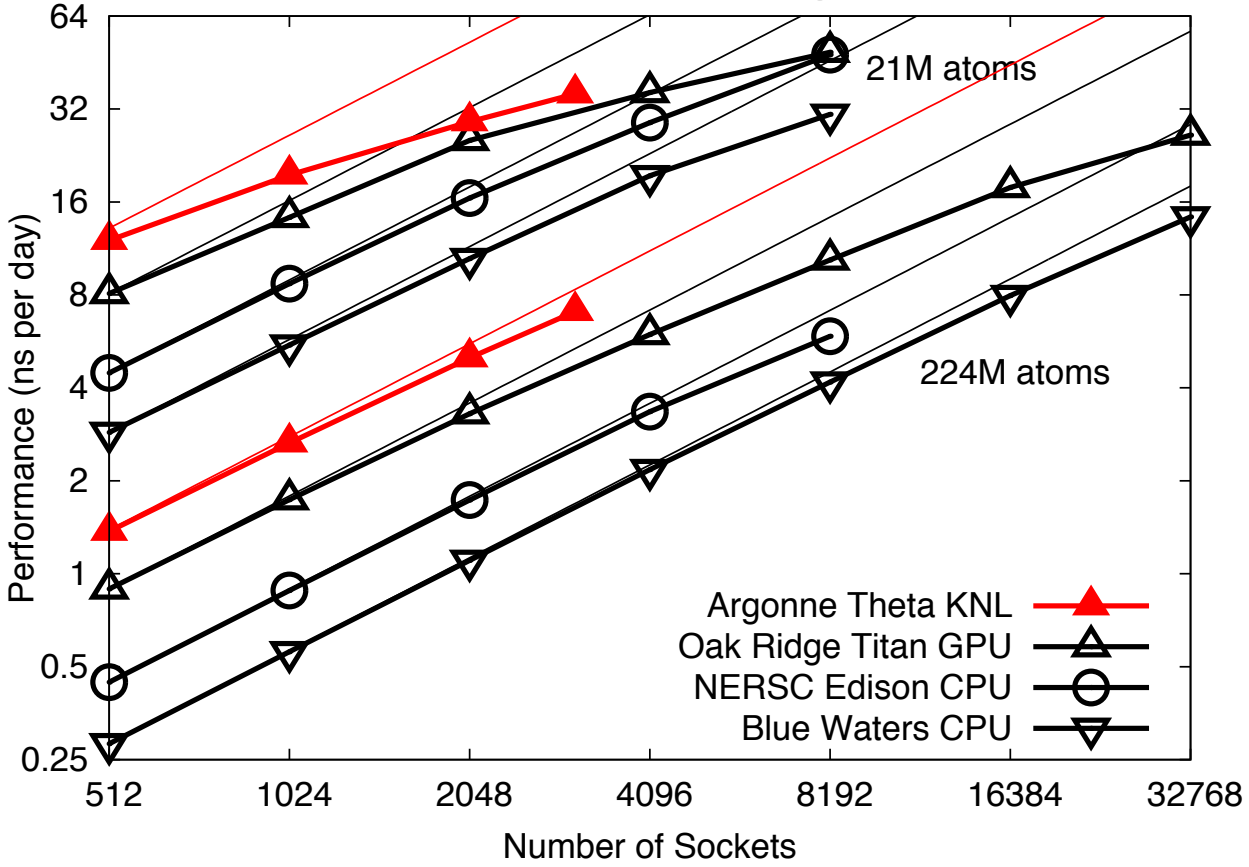
- 68 cores, reserve one for OS, leaves 67
- 65 = 13*5 = 13*(4+1) = 52 PE + 13 comm  +ppn 8 +pemap 0-51+68 +commap 53-65
- 66 = 6*11 = 6*(10+1) = 60 PE + 6 comm  +ppn 20 +pemap 0-59+68 +commap 60-65
- 68 = 4*17 = 4*(16+1) = 64 PE + 4 comm  +ppn 32 +pemap 0-63+68 +commap 64-67

# Argonne Theta KNL port

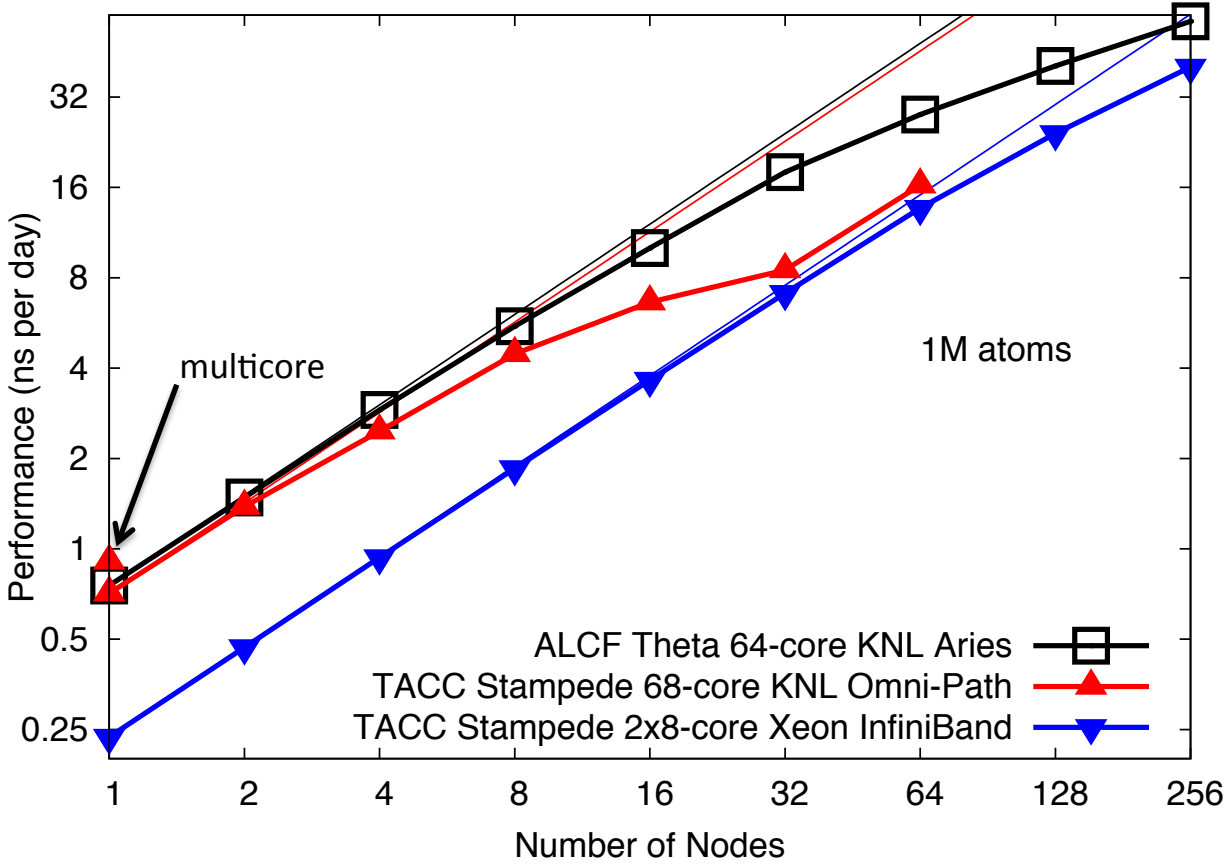


# KNL has highest performance *per-socket*

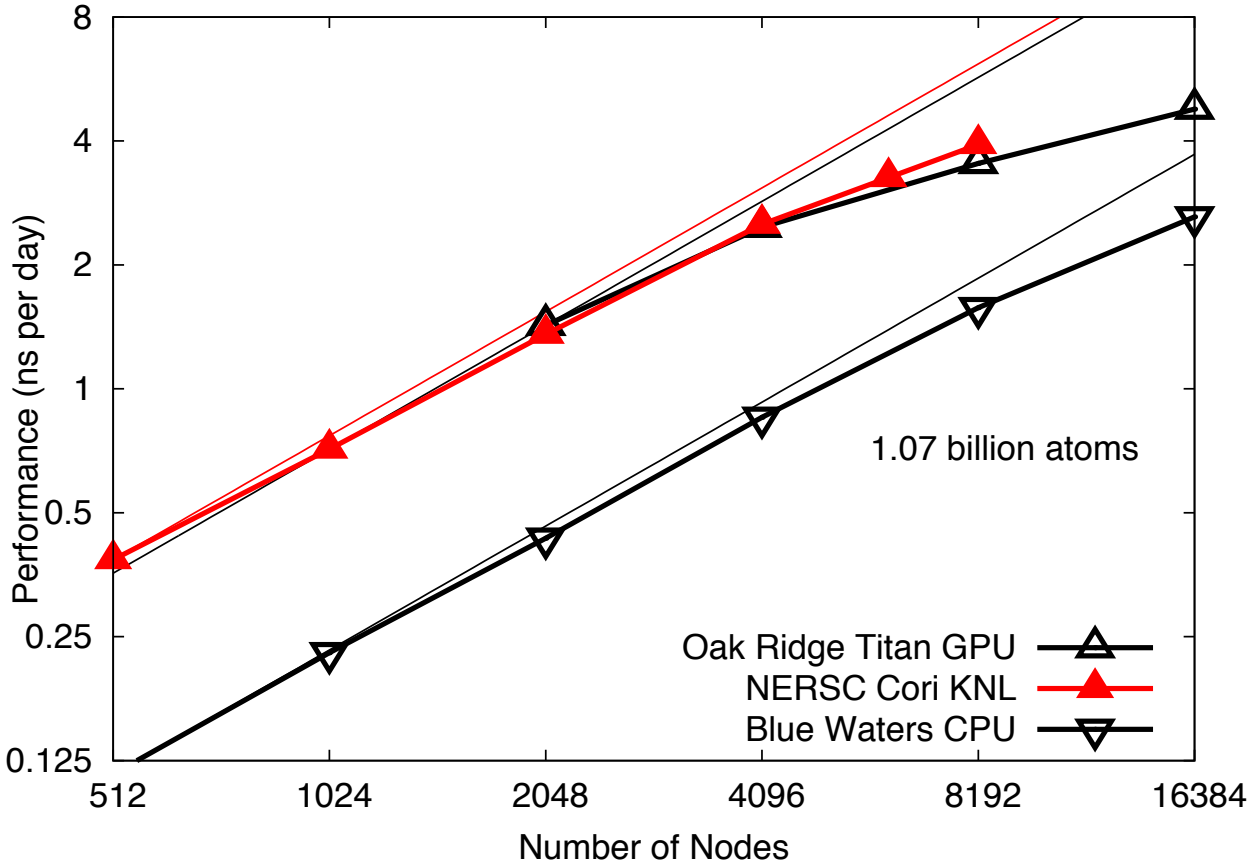
## (But GPUs on Titan are two generations old.)



# TACC Stampede KNL

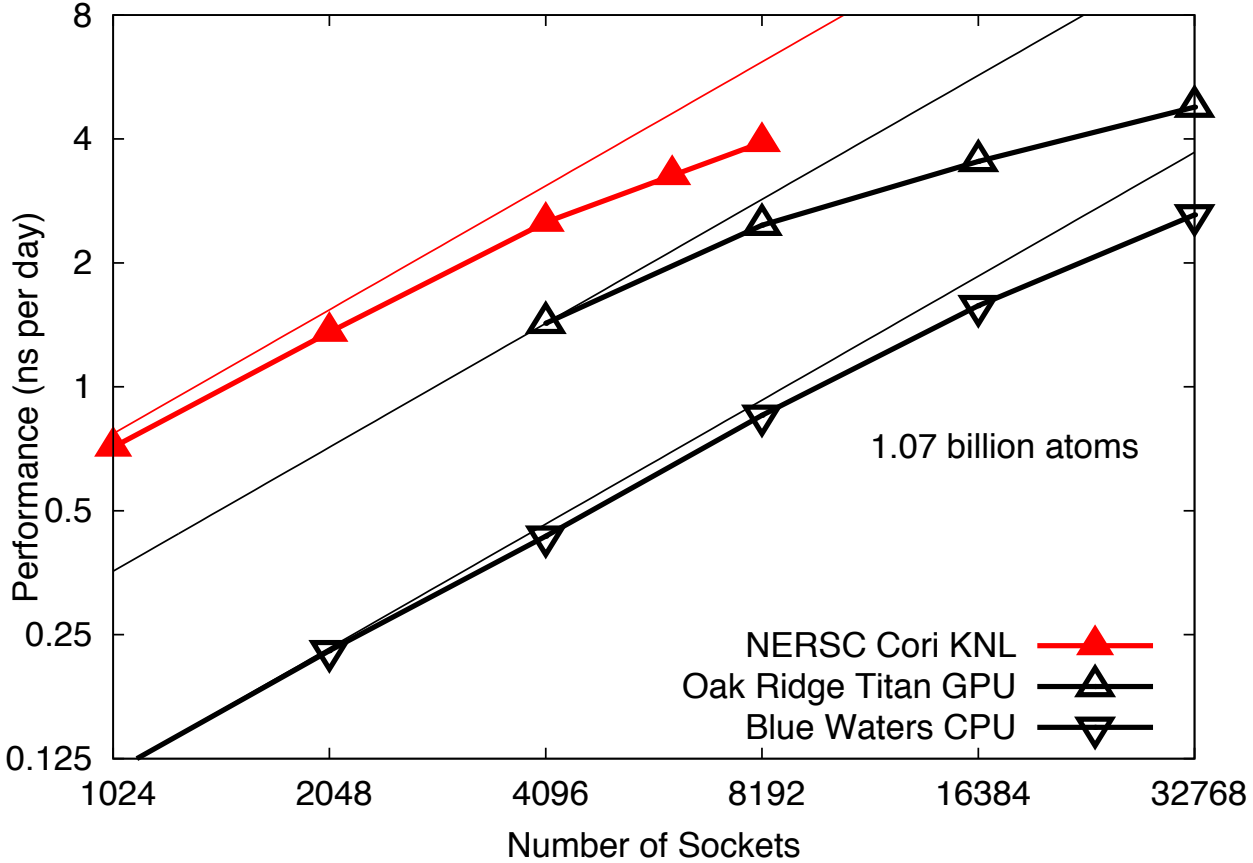


# New billion-atom benchmark on NERSC Cori KNL



# Again, NERSC Cori KNL looks better *per-socket.*

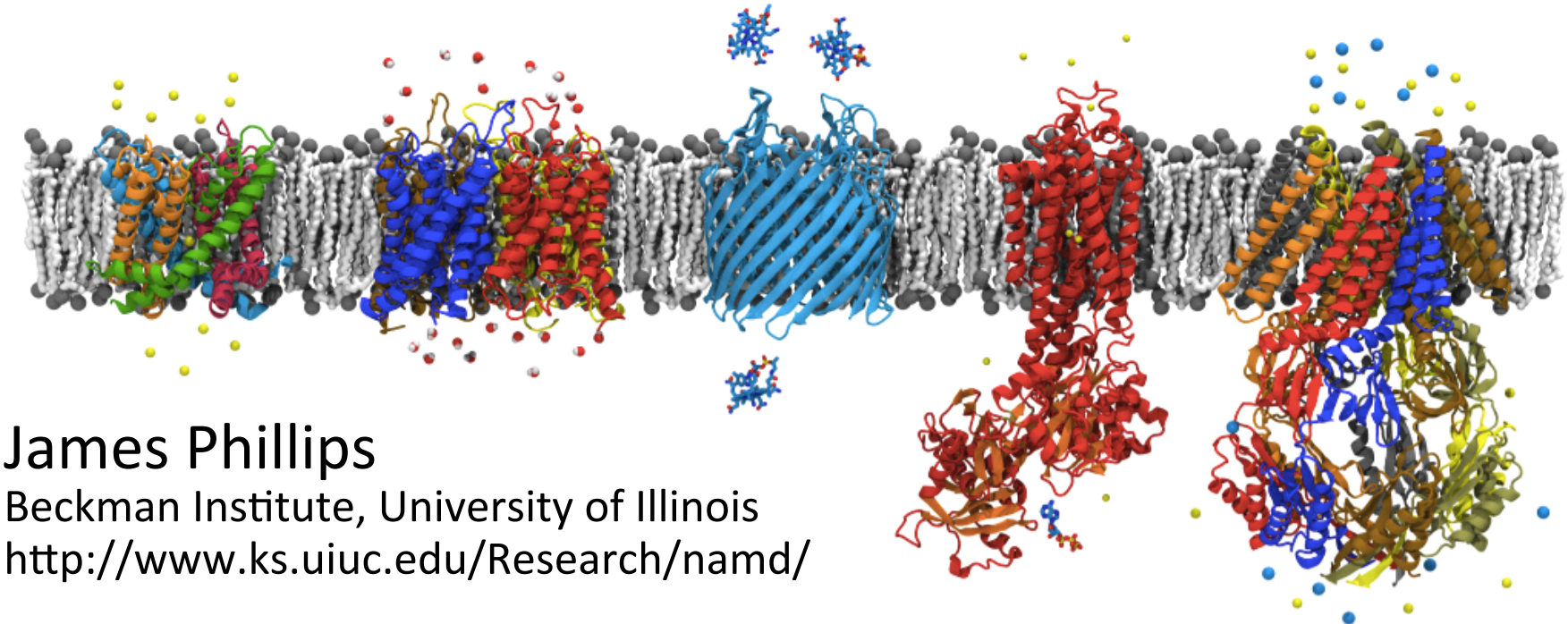
## Target platforms are 10x (2018) and 100x (2023) faster.



# Conclusions and Future Work

- AVX-512 with Intel compilers
  - Works if you know limits/tricks, watch for bugs
- MCDRAM high-bandwidth memory
  - Cache mode works, watch for thrashing at scale
- Requires Charm++ SMP build, +pemap, +commap
- Cray Aries works well with gni, 7 processes/node
- Omni-Path works OK with MPI, 13 processes/node
  - “On-loaded” architecture bottlenecks on slow cores
  - Specialized PSM2/OFI network layer might help

Thanks to: NIH, NSF, DOE, NCSA, ALCF, OLCF, TACC, PSC, SDSC, and 20+ years of NAMD and Charm++ developers and users.



James Phillips
Beckman Institute, University of Illinois
http://www.ks.uiuc.edu/Research/namd/