Porting LAMMPS to the Titan Supercomputer

W. Michael Brown
National Center for Computational Sciences
Oak Ridge National Laboratory

Titan Users and Developers Workshop (West Coast)
January 31, 2013

LAMMPS

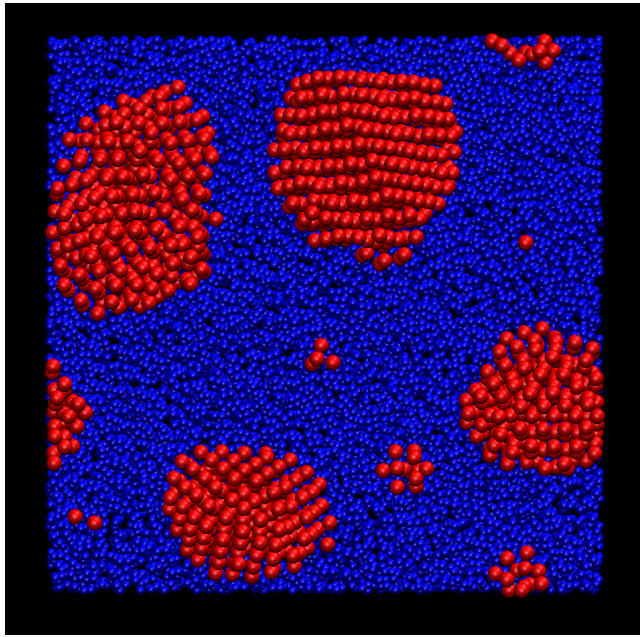
- Large-scale Atomic/Molecular Massively Parallel Simulator
  - http://lammps.sandia.gov
- General Purpose Classical Molecular Dynamics Software
  - Simulations for biology, materials science, granular, mesoscale, etc.
- Open source (GPL), with a user community
  - 1700 Citations
  - 100K downloads, 25K mail list messages
  - 100 contributors to code base
- Flexible and extensible:
  - 80% of code-base is add-ons by developers and users
  - styles: atom, pair, fix, compute, etc

Force Fields

- Biomolecules: CHARMM, AMBER, OPLS, COMPASS (class 2), Gromacs, long-range Coulombics via PPPM, point dipoles, ...
- Polymers: all-atom, united-atom, coarse-grain (bead-spring FENE), bond-breaking, bond-forming, ...
- Materials: EAM/MEAM for metals, Buckingham, Morse, Yukawa, Tersoff, COMB, AI-REBO, ReaxFF, GAP, …
- Mesoscale: granular, DPD, SPH, PD, colloidal, aspherical, triangulated, …

Hybrid Models

- Water/proteins on metal/silica surface
- Metal-semiconductor interface
- Metal islands on amorphous (LJ) substrate



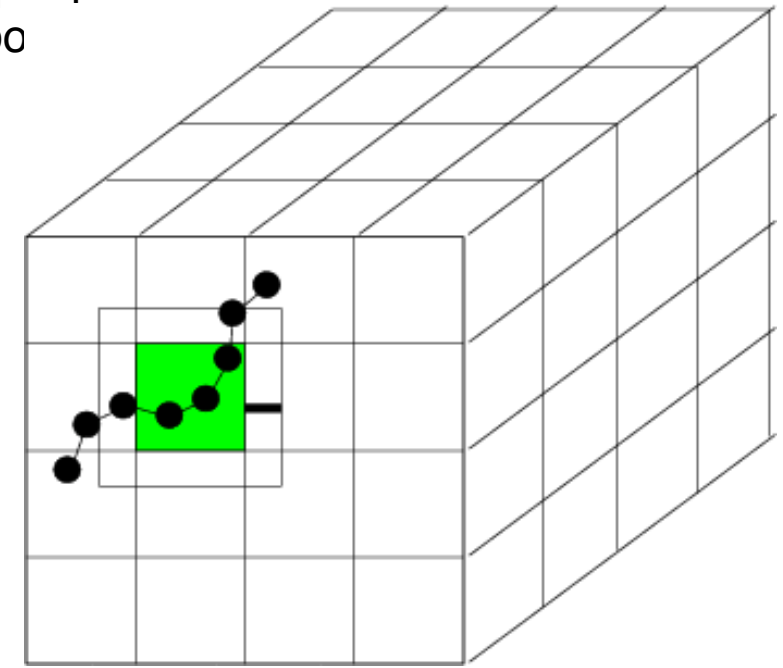
- Specify 2 (or more) pair potentials:
  - A-A, B-B, A-B, etc
- Hybrid in two ways:
  - potentials (pair, bond, etc)
  - atom style (bio, metal, etc)

Spatial Decomposition

- Physical domain divided into 3d boxes, one per processor
- Each proc computes forces on atoms in its bo
  - using info from nearby procs
- Atoms "carry along" molecular topology
  - as they migrate to new procs

- Communication via
  - nearest-neighbor 6-way stencil

- Advantages:
  - communication scales
    - sub-linear as (N/P)2/3
    - (for large problems)
  - memory is optimal N/P



Developing a Strategy for Porting LAMMPS

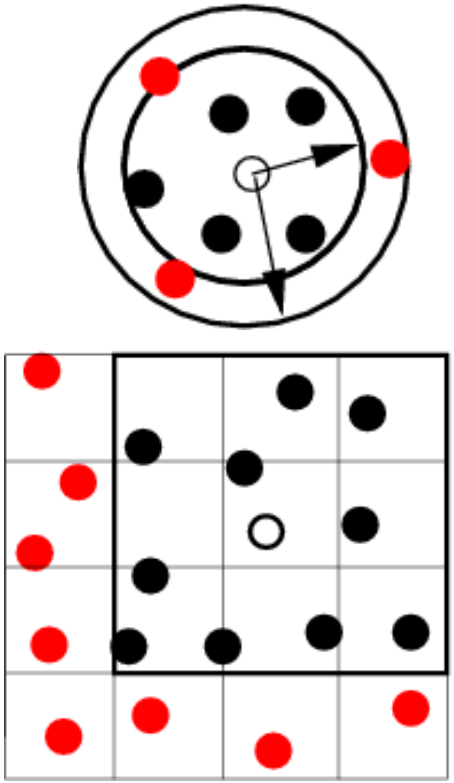
- The primary developer of LAMMPS is Steve Plimpton at SNL

  (307) Debugging is at least twice as hard as writing the program in the first place. So if your code is as clever as you can possibly make it, then by definition you're not smart enough to debug it.
  -- *Brian Kernighan*

- Goal is to get efficient acceleration without modifying the LAMMPS core
  - Acceleration will be maintained with the main LAMMPS package
  - Compatibility with all of LAMMPS features

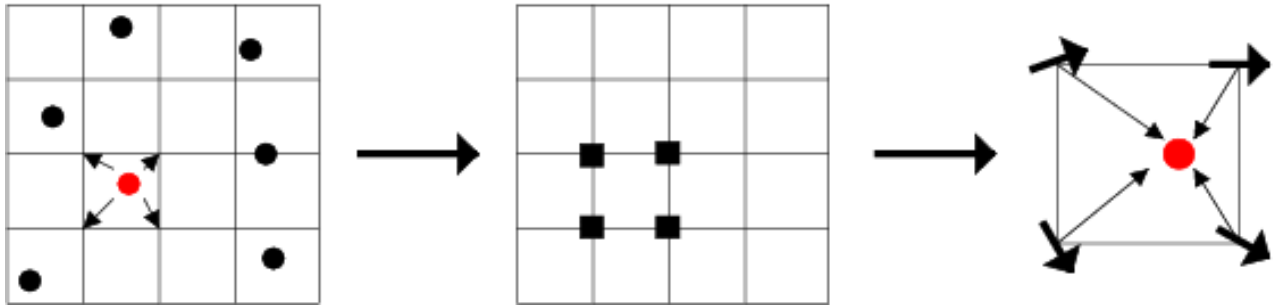
Classical Molecular Dynamics

- Time evolution of a system of particles
- Calculate the force on a particle due to other particles in the system as the gradient of the potential energy
  - 2-body and many-body potentials
  - Short-range forces are calculated using only particles within a spherical cutoff
  - For some models, additional long range calculations must be performed
  - For molecular models, additional forces due to bonds
- Apply constraints, thermostats, barostats
- Do statistics calculations, I/O
- Time integration to find new position of particles at next time step



PPPM (Particle-Mesh Ewald)

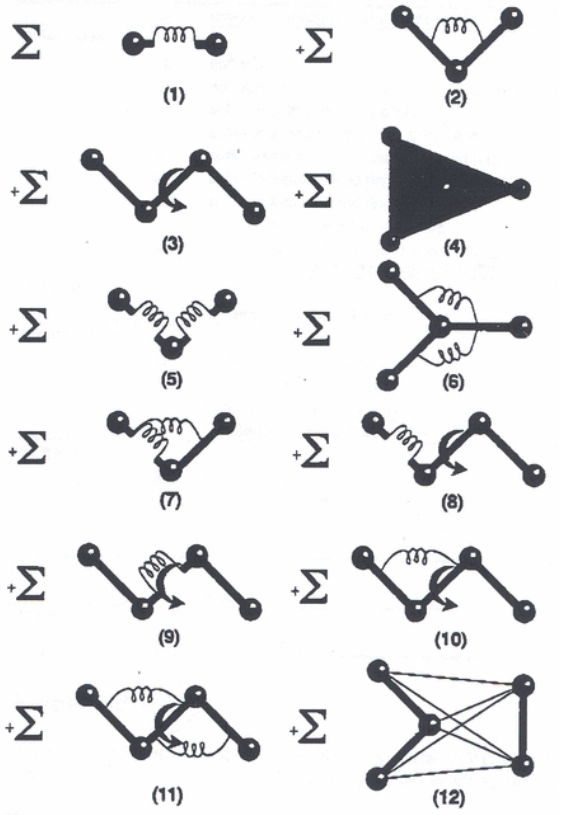
- Long range coulombics needed for many systems (charged polymers (polyelectrolytes), Organic & biological molecules, Ionic solids, oxides
- *Hockney & Eastwood, Comp Sim Using Particles (1988).*
- *Darden, et al, J Chem Phys, 98, p 10089 (1993).*
- Same as Ewald, except integral evaluated via:
  - interpolate atomic charge to 3d mesh
  - solve Poisson's equation on mesh (FFTs)
  - interpolate E-fields back to atoms



Molecular Force Fields (Class 1 & 2)

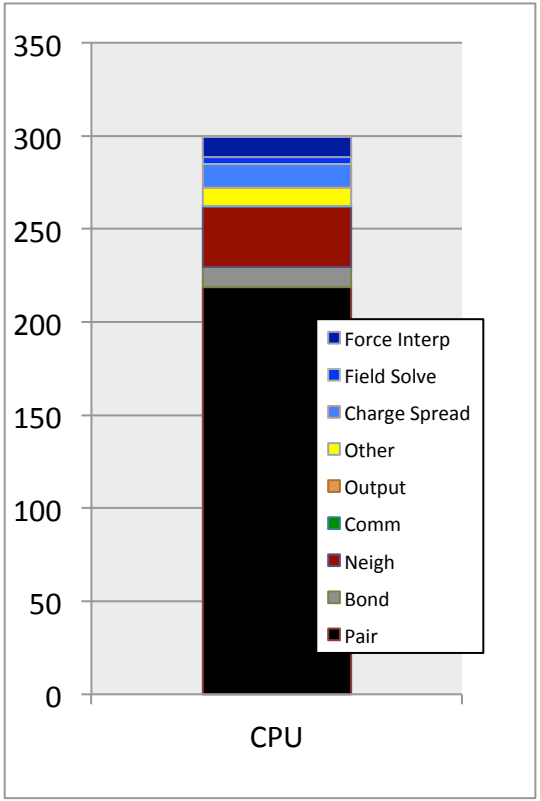
$$E(r) = K(r - r_0)^2$$
$$E(\theta) = K(\theta - \theta_0)^2$$
$$E(\phi) = K(1 + d\cos(n\phi))$$
$$E(\psi) = K(\psi - \psi_0)^2$$

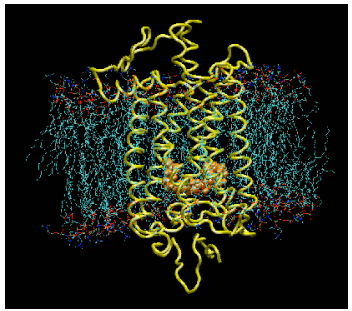
- Higher-order cross terms
- Originated by
  - BioSym → MSI → Accelrys
  - www.accelrys.com
- Parameters available in their
  - commercial MD code



Rhodopsin Simulation

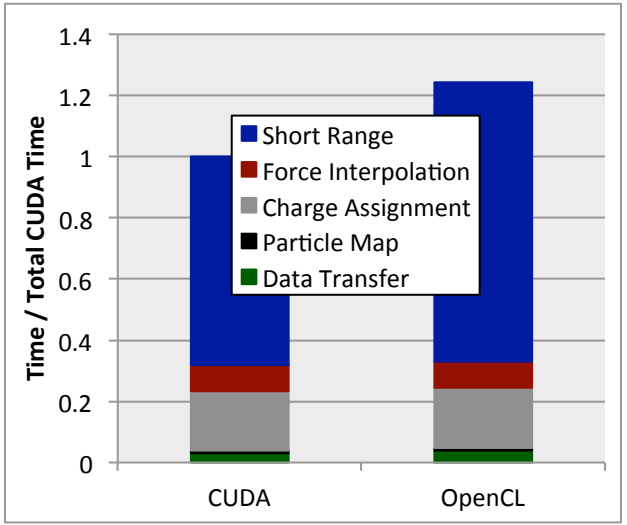
```
# Rhodopsin model

units           real
neigh_modify    delay 5 every 1

atom_style      full
bond_style      harmonic
angle_style     charmm
dihedral_style  charmm
improper_style  harmonic
pair_style      lj/charmm/coul/long 8.0 10.0
pair_modify     mix arithmetic
kspace_style    pppm 1e-4

read_data       data.rhodo

fix             1 all shake 0.0001 5 0 m 1.0 a 232
fix             2 all npt temp 300.0 300.0 100.0 &
                z 0.0 0.0 1000.0 mtk no pchain 0
tchain 1

special_bonds   charmm

thermo          50
thermo_style    multi
timestep        2.0

run             100
```





Geryon Library

- Allows same code to compile with CUDA Runtime, CUDA Driver, and OpenCL APIs
- Simple classes allow for more succinct code than CUDA-Runtime
  - Change from one API to another by simply changing the namespace
  - Use multiple APIs in the same code
  - Lightweight (only include files – no build required)
  - Manage device query and selection
  - Simple vector and matrix containers
  - Simple routines for data copy and type casting
  - Simple routines for data I/O
  - Simple classes for managing device timing
  - Simple classes for managing kernel compilation and execution



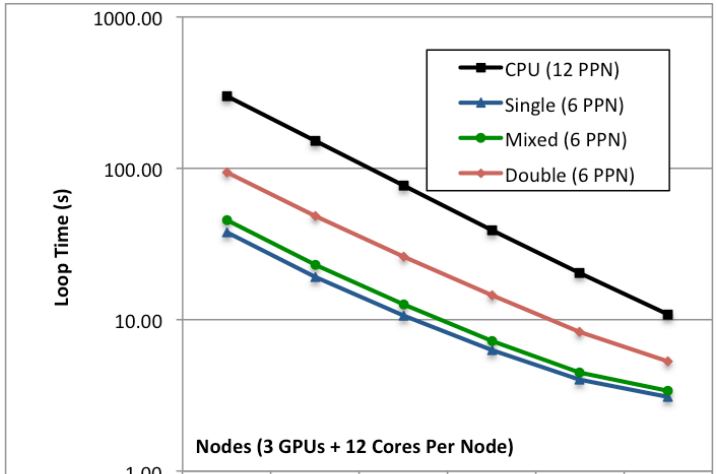
http://users.nccs.gov/~wb8/geryon/index.htm

Accelerating Force Calculation

- Add a new “pair_style” in the same manner that new force models are added (e.g. lj/cut/gpu vs lj/cut)
- Copy the atom data, neighbor list from the host
- Assign each particle to a thread, thread accumulates the force for all of its neighbors
- Repack the neighbor list on the GPU for contiguous memory access by threads
  - On CPU, only need to calculate the force once for both particles in the pair
    - For shared memory parallelism, must handle memory collisions
    - Can avoid this by calculating the force twice, once for each particle in the pair
      - Doubles the amount of computation for force
      - Doubles the size of the neighbor list

Accelerating Force Calculation

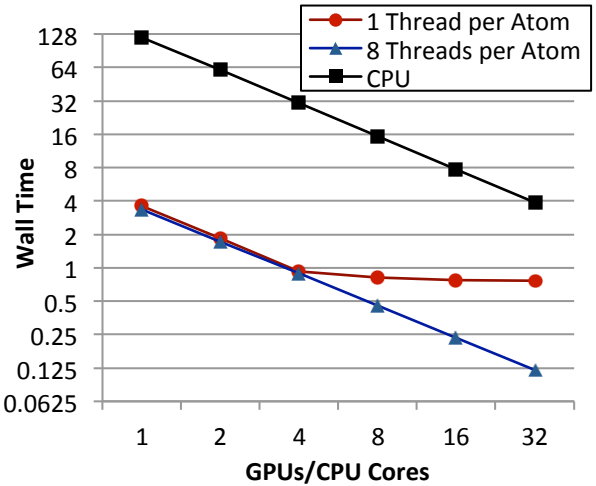
- Random memory access is expensive!
  - Store atom type with the position in a 4-vector
    - Better data alignment, reduces the number of random access fetches
    - Periodically sort data by location in simulation box in order to improve data locality
- Mixed precision – perform computations in single precision with all accumulation and answer storage in double precision
- 50x M2070 speedup versus single Istanbul core



	1	2	4	8	16	32
CPU (12 PPN)	297.80	151.53	76.74	38.71	20.44	10.83
Single (6 PPN)	37.44	19.16	10.66	6.31	4.02	3.08
Mixed (6 PPN)	45.13	23.07	12.65	7.25	4.46	3.36
Double (6 PPN)	94.04	48.33	25.97	14.40	8.24	5.30

Accelerating Force Calculation

- 512 Compute Cores on Fermi
- Want many more threads than this to hide latencies
- With thread per atom decomposition, need lots of atoms per GPU
- Increase parallelism with multiple threads per atom
  - Tradeoff between increased parallelism and increased computation/synchronization
    - For NVIDIA and AMD, accelerators can avoid synchronization for *n* threads if *n* is a power of 2 and *n<C.*
    - *n* depends on the model and the number of neighbors – LAMMPS will guess this for you
    - Need to change neighbor list storage in order to get contiguous memory access for an arbitrary number of threads
      - Can increase the time for neighbor stuff, but generally a win

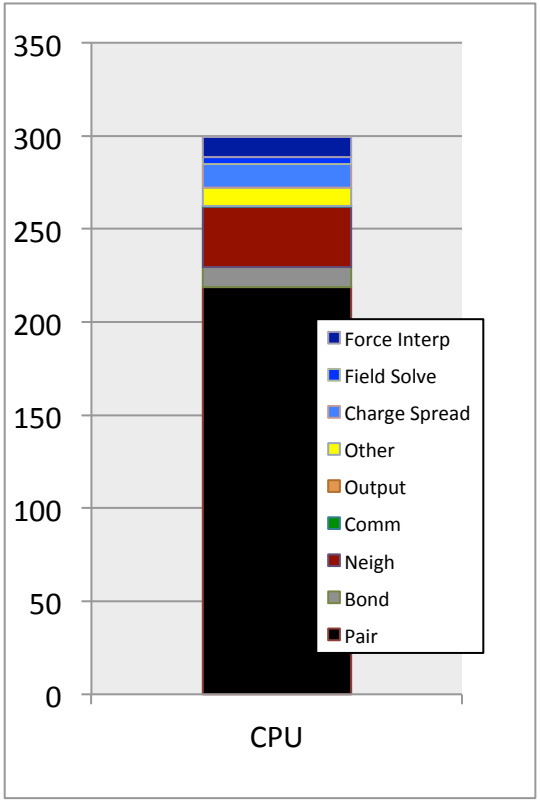


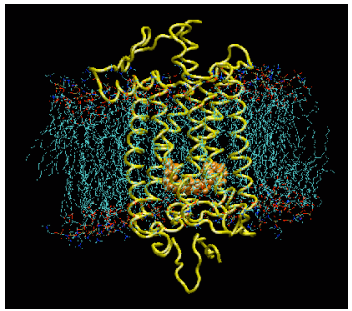
Accelerated Non-bonded Short-Range Potentials

- Single, mixed, and double precision support for:
  - lj/cut
  - lj96/cut
  - lj/expand
  - lj/cut/coul/cut
  - lj/cut/coul/long
  - lj/charmm/coul/long
  - lj/class2
  - lj/class2/coul/long
  - morse
  - cg/cmm
  - cg/cmm/coul/long
  - coul/long
  - gayberne
  - resquared
  - gauss
  - morse
  - buck/cut
  - buck/coul/cut
  - buck/coul/long
  - eam
  - eam/fs
  - eam/alloy
  - table
  - yukawa
  - born
  - born/coul/long
  - born/coul/wolf
  - colloid
  - coul/dsf
  - coul/long
  - dipole/cut
  - dipole/sf
  - lj/cut/coul/debye
  - lj/cut/coul/dsf
  - lj/expand

Rhodopsin Simulation

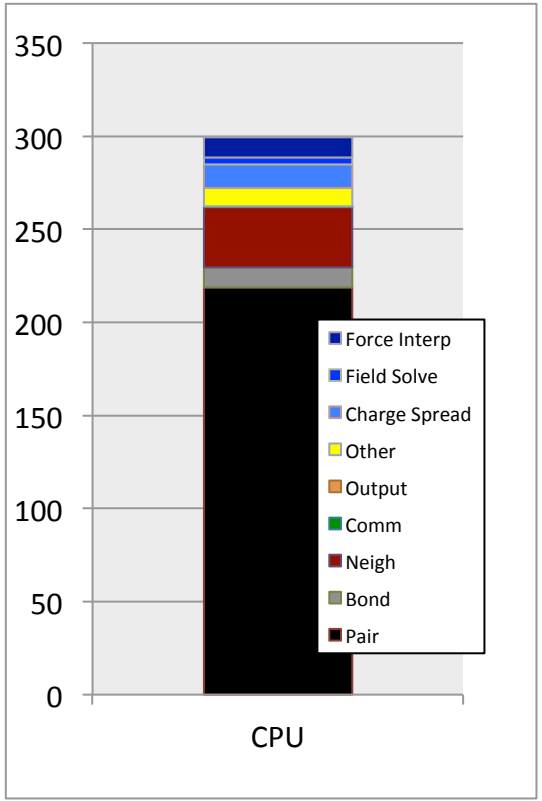
```
# Rhodopsin model
newton off    # For GPU, we double the compute work
              # to avoid memory collisions

units           real
neigh_modify    delay 5 every 1

atom_style      full
bond_style      harmonic
angle_style     charmm
dihedral_style  charmm
improper_style  harmonic
pair_style      lj/charmm/coul/long/gpu 8.0 10.0
pair_modify     mix arithmetic
kspace_style    pppm 1e-4

read_data       data.rhodo

fix             1 all shake 0.0001 5 0 m 1.0 a 232
fix             2 all npt temp 300.0 300.0 100.0 &
                z 0.0 0.0 1000.0 mtk no pchain 0
tchain 1

special_bonds   charmm

thermo          50
thermo_style    multi
timestep        2.0

run             100
```

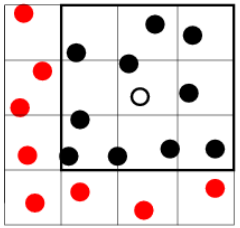




Neighbor List Builds

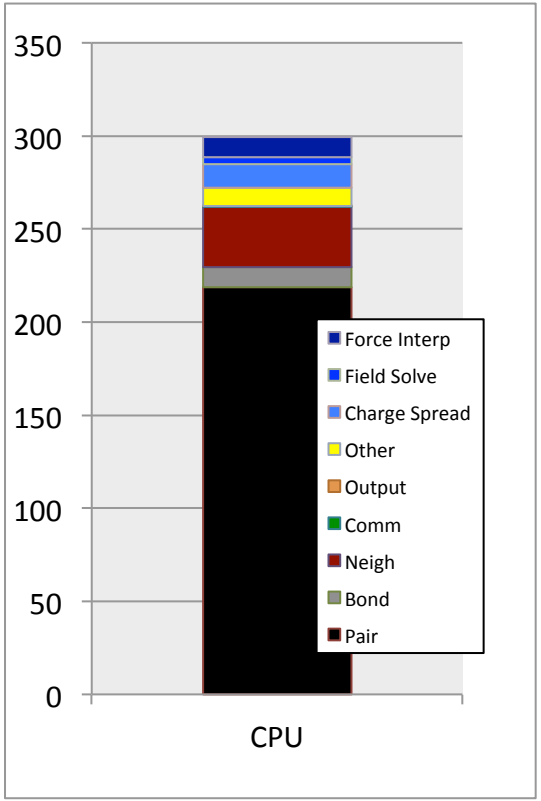
- CPU time for neighbor list builds can now become dominant with GPU acceleration
- Build neighbor lists on the GPU
  - In LAMMPS, neighbor lists are requested by the "pair_style" – just don't request a neighbor list and you are free to do it on the GPU
  - This is optional for several reasons
- First calculate cell list on GPU
  - Requires binning
  - Atomic operations or sort
    - LAMMPS uses radix sort to get determinism or optionally performs the binning only on the CPU
- Calculate verlet list from cell list
- Copy and transpose data for 1-2, 1-3, 1-4 interactions
- 8.4x M2070 Speedup vs Single Core Istanbul (Memory)





Particle-Particle Particle-Mesh

- Most of the time spent in charge spreading and force interpolation
  - Field solve is 3D FFT
    - Communication intensive
    - Can run with an arbitrary FFT library, including cuFFT
      - Generally, not enough work to be worth it
- Charge spreading
- Can't use the CPU algorithm (efficiently)
  - First, map the particles to mesh points
    - Requires atomic operations to count number of atoms at each point
    - Problem, since the particles are sorted spatially
      - Reindex the particles accessed by each thread to reduce the probability of memory collisions
        - Reduces kernel time by up to 75%



Particle-Particle Particle-Mesh

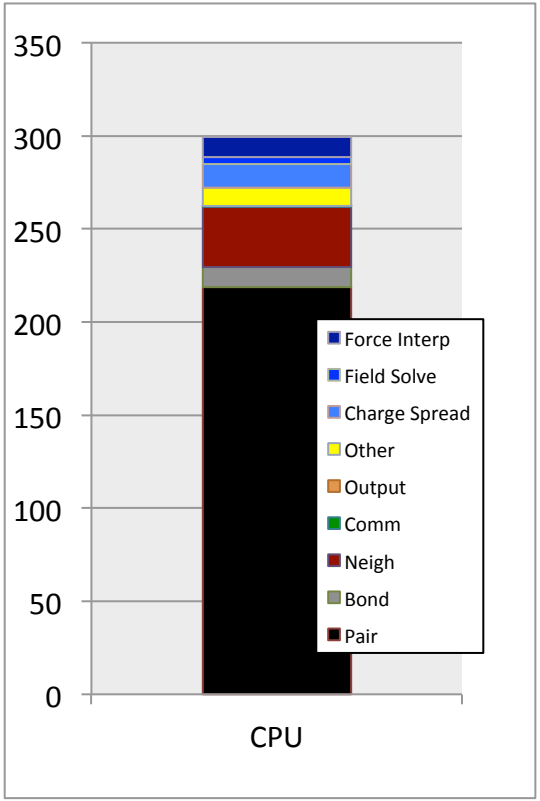
- Second, spread the charges on the mesh using splines
  - Difficult stencil problem
    - Can't tile due to shared/local memory limitations
    - Use a pencil decomposition (on the GPU within each subdomain assigned to the process)
      - Avoid atomic operations for charge spreading (still slower even with hardware support for floating point atomics)
      - Contiguous memory access for particle positions and charges mapped to mesh
      - Tradeoff is that a significant amount of recomputation is required compared to the naïve and CPU algorithms
    - Assign multiple pencils to each multiprocessor to improve strong scaling performance
      - This requires an increase in the number of global memory fetches, so there are limits
- 13.7X M2070 speedup versus single core Istanbul (small number of mesh points)

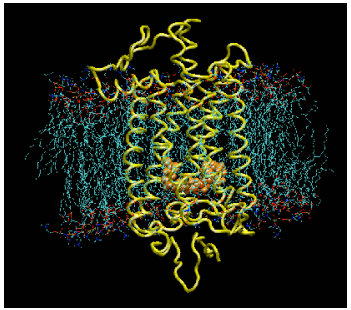
Particle-Particle Particle-Mesh

- Force Interpolation
  - Use same algorithm as CPU
    - Thread per atom
    - No memory collisions, etc.
- 27.8x Speedup versus 1 core Istanbul

Rhodopsin Simulation

```
# Rhodopsin model
newton off

units           real
neigh_modify    delay 5 every 1

atom_style      full
bond_style      harmonic
angle_style     charmm
dihedral_style  charmm
improper_style  harmonic
pair_style      lj/charmm/coul/long/gpu 8.0 10.0
pair_modify     mix arithmetic
kspace_style    pppm/gpu 1e-4

read_data       data.rhodo

fix             1 all shake 0.0001 5 0 m 1.0 a 232
fix             2 all npt temp 300.0 300.0 100.0 &
                z 0.0 0.0 1000.0 mtk no pchain 0
tchain 1

special_bonds   charmm

thermo          50
thermo_style    multi
timestep        2.0

run             100
```



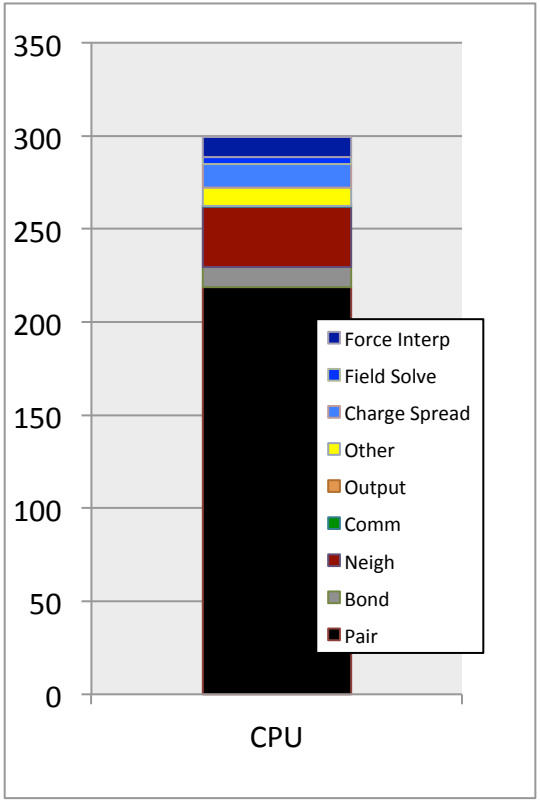


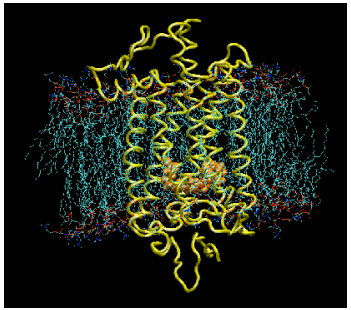
CPU/GPU Concurrency

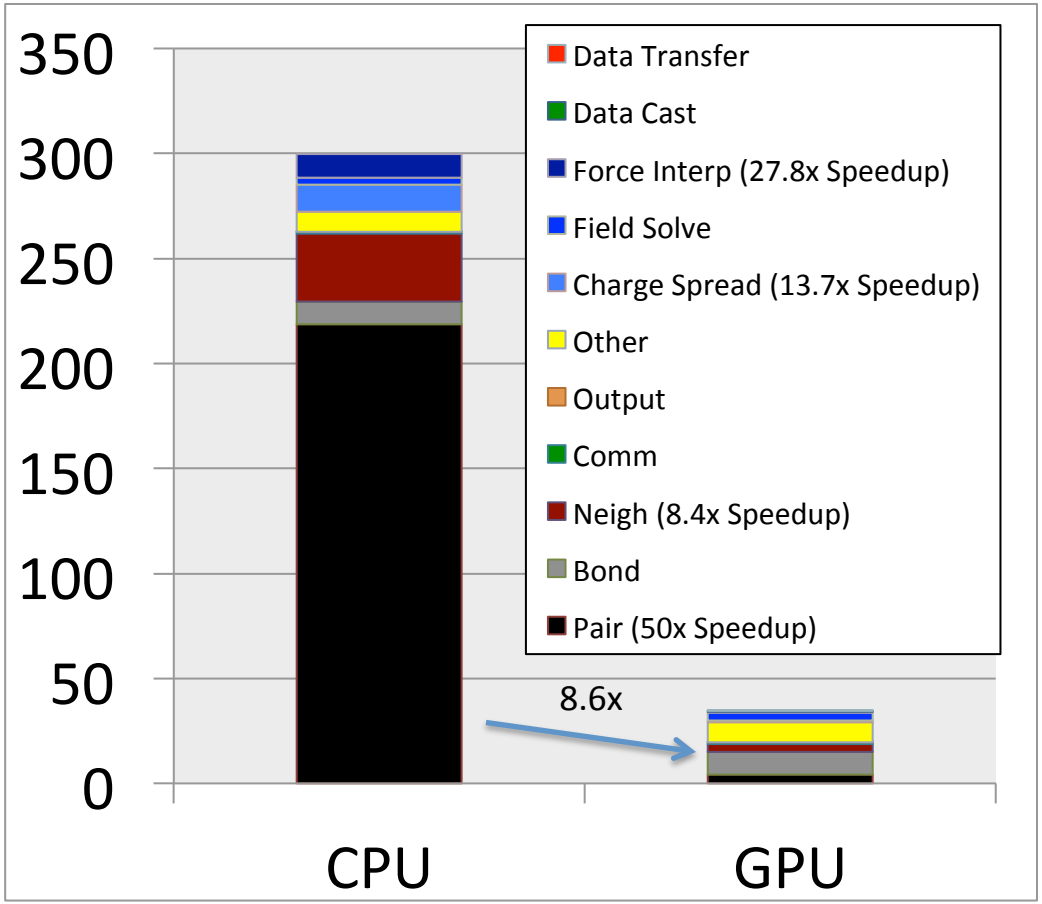
- Can compute short-range, bond, angle, dihedral, improper, k-space contributions to the forces, energies, and virials independently of each other.
- Run the calculations not ported to the accelerator simultaneously
  - In some cases, there is no possible gain from porting additional routines
- Can also divide the force calculation between the CPU and the accelerator with dynamic load balancing
- Can run hybrid models with one model calculated on the CPU and the other on the accelerator
- Add a new “fix” to LAMMPS with a hook to block and move accelerator data to the host before proceeding with time integration, etc.

Rhodopsin Simulation

```
# Rhodopsin model
newton off
fix gpu force/neigh 0 0 1

units           real
neigh_modify    delay 5 every 1

atom_style      full
bond_style      harmonic
angle_style     charmm
dihedral_style  charmm
improper_style  harmonic
pair_style      lj/charmm/coul/long/gpu 8.0 10.0
pair_modify     mix arithmetic
kspace_style    pppm/gpu 1e-4

read_data       data.rhodo

fix             1 all shake 0.0001 5 0 m 1.0 a 232
fix             2 all npt temp 300.0 300.0 100.0 &
                z 0.0 0.0 1000.0 mtk no pchain 0
tchain 1

special_bonds   charmm

thermo          50
thermo_style    multi
timestep        2.0

run             100
```



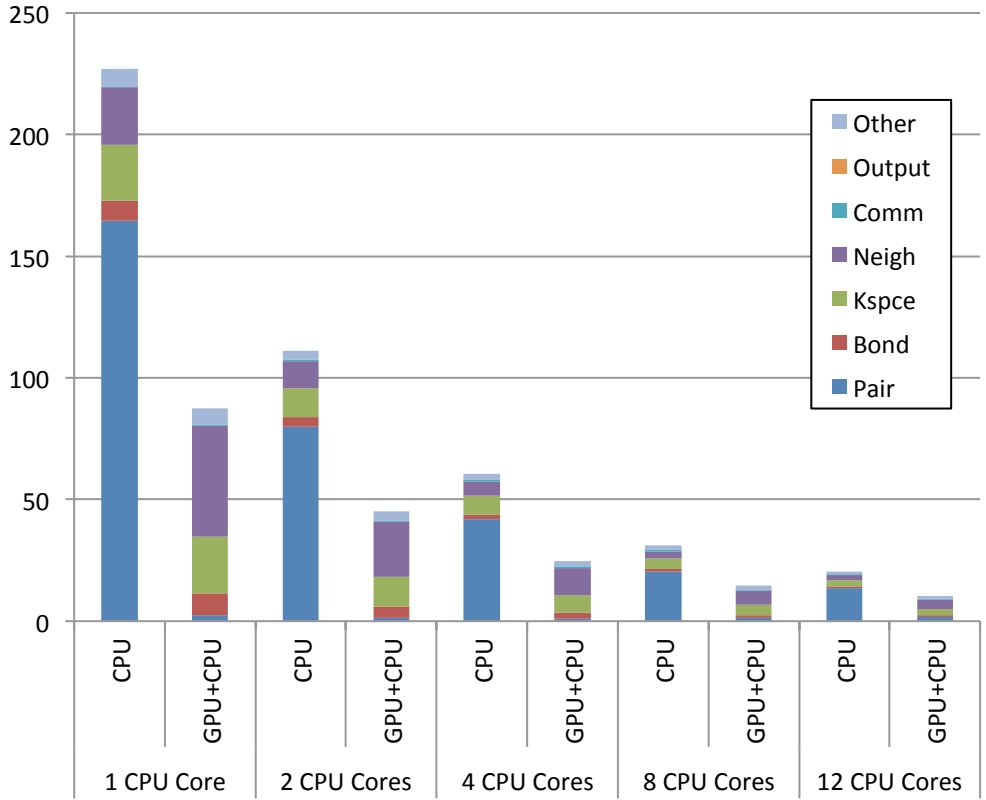




But we have 16 cores on the Interlagos chip and LAMMPS scales well…

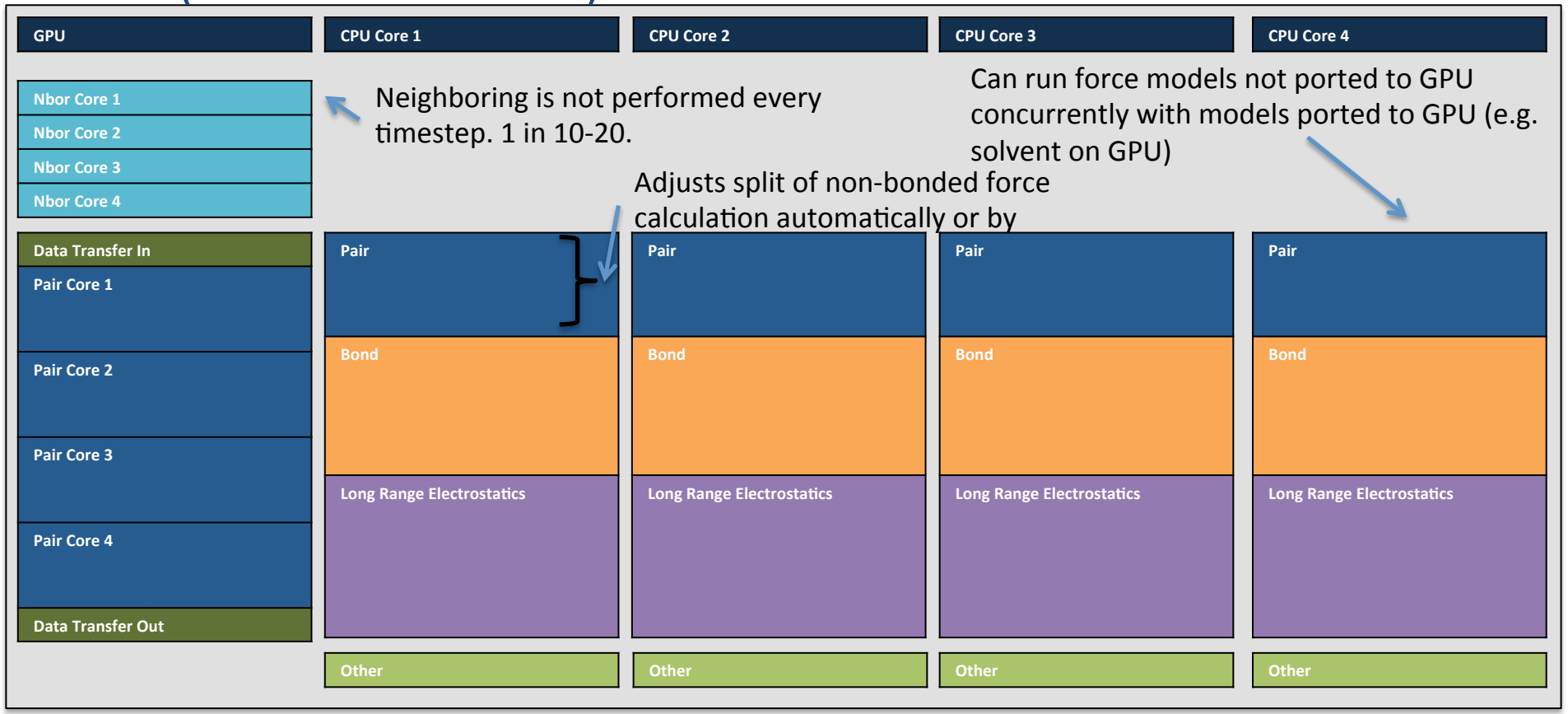
Need to parallelize code not ported for the accelerator on the CPU cores:

- Example: Rhodopsin Benchmark on a box with 12 CPU cores and 1 GPU
  - Here, only the non-bonded "pair" force calculation is accelerated for an example
  - Speedup on 1 CPU core with a GPU is only 2.6x with this approach (M2070 vs Istanbul)
  - Speedup vs 12 CPU cores is still > 2x because we use all 12 cores in addition to the GPU
    - MPI Processes Share the GPU

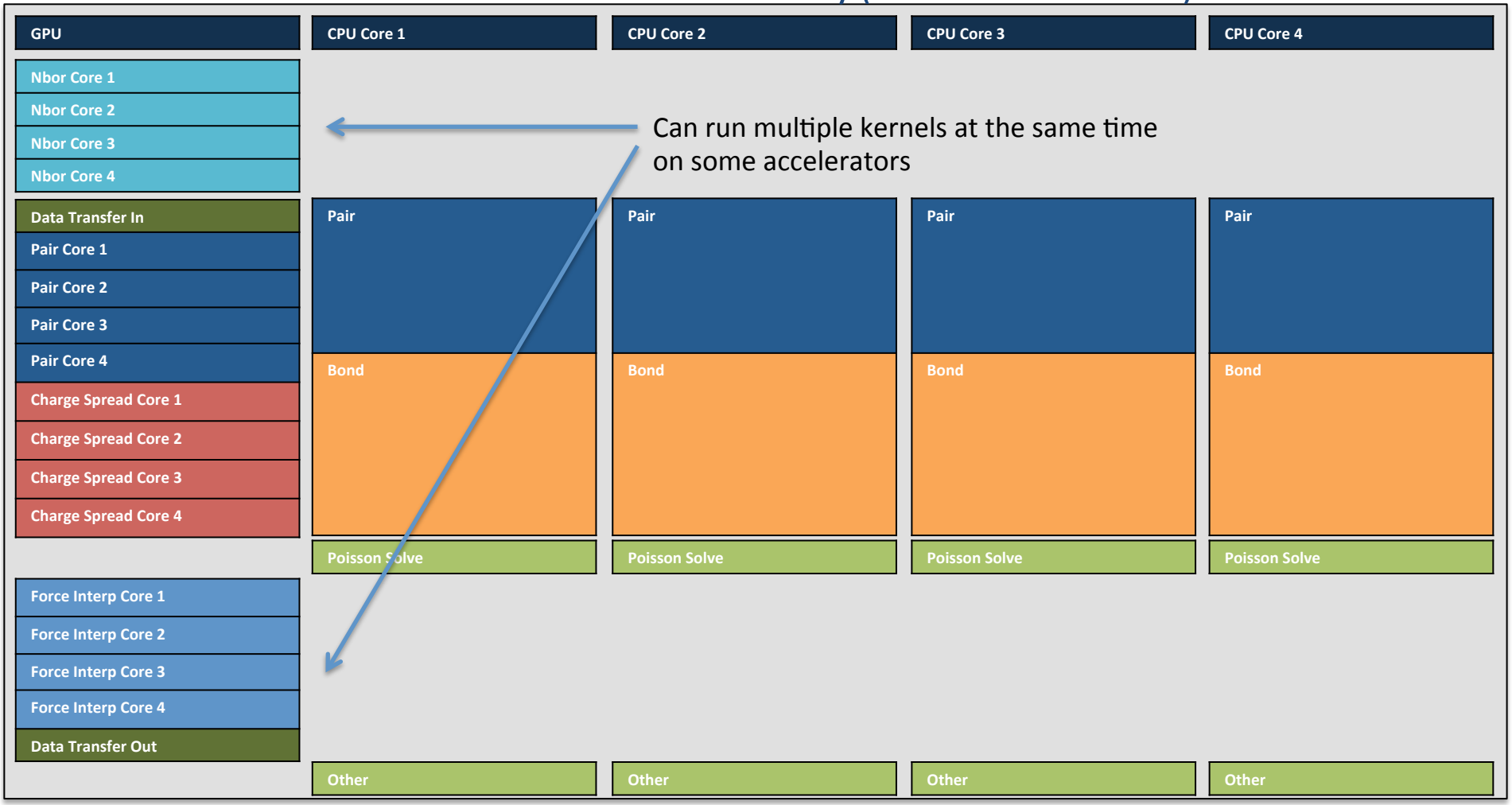


Host-Accelerator Concurrency

(- PPPM Acceleration) *Not to scale*



Host-Accelerator Concurrency (+PPPM Acceleration)



Minimizing the Code Burden

- Focus porting efforts on the routines that dominate the computational time and have high potential for speedup with accelerators
- Use concurrent CPU/GPU calculations where possible and overlap host-accelerator data transfer with computation
- Parallelize routines that are not ported on the CPU
- Allow a legacy code to take advantage of accelerators without rewriting the entire thing
- In some cases, there is no advantage to porting
  - Multi-Core CPUs can be competitive with GPU accelerators for some routines (latency-bound, thread divergence, etc.), especially at lower particle counts
  - If concurrent CPU/GPU calculations are used effectively, there can be no advantage to porting certain routines
  - For some simulations, the upper bound for performance improvements due to porting additional routines and removing data transfer is < 5%.

Optimizations to CPU code for the XK6

- Using MPI-only on multicore nodes can impact performance at scale
  - Increased MPI communication times
    - MPI Cartesian topologies (e.g. MPI_Cart_create) are designed to allow vendors to optimize process mapping for grids
      - Typically do nothing to optimize for multicore nodes
    - Implemented a two-level domain factorization algorithm to minimize off-node communications
  - Increased K-space times for P^3M calculations
    - Many processes involved in effectively all-to-all communications for 3D FFTs
    - Use separate process partitions for the K-space calculation
      - Implemented by Yuxing Peng and Chris Knight from the Voth Group at the University of Chicago

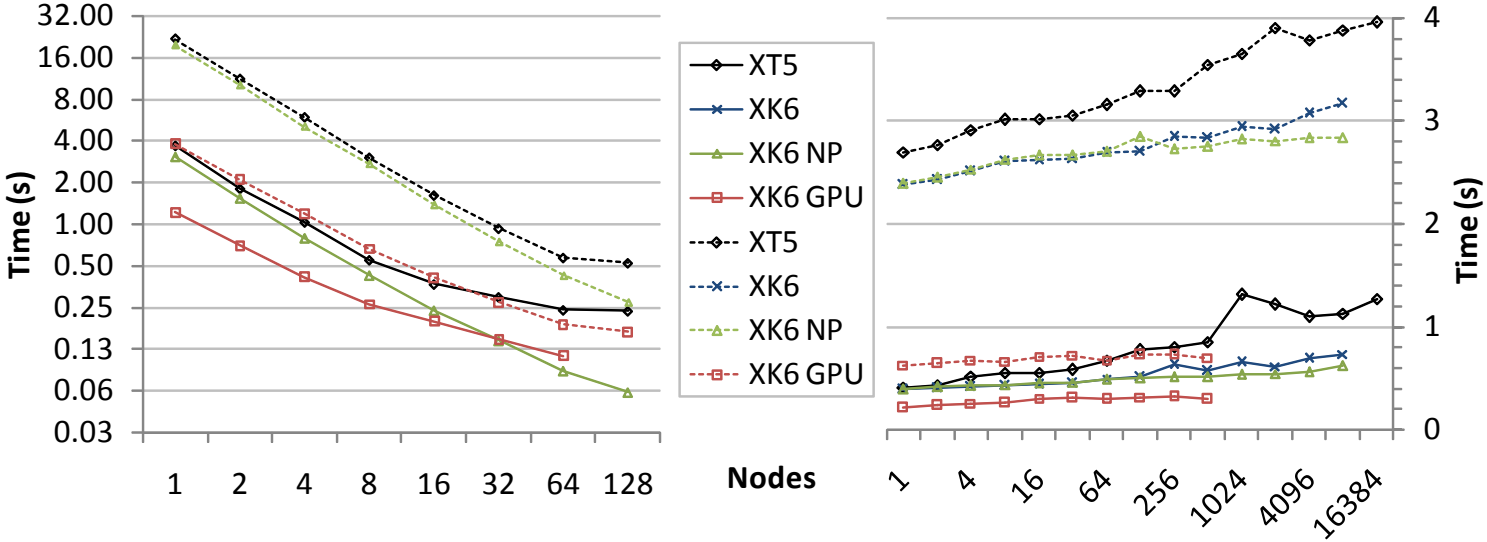
X2090 Benchmark Results

- Jaguar upgrade to Titan consisted of:
  1. Upgrade XT5 to XK6 blades (AMD Interlagos Processors and Gemini Interconnect)
  2. Install ***Titan Development Partition*** with Fermi+ GPUs on 10 cabinets
  3. Upgrade to XK7 nodes (Installation of K20X Kepler II GPUs on all all nodes)
- XK6 Upgrade and Titan Development results are presented first
- **Early results with Kepler on Titan presented at end of talk**
- Benchmarks with acceleration used mixed-precision as compared to double precision
- Left: Strong Scaling for fixed-size simulations of approximately 256K particles
- Right: Weak Scaling with ~32K particles/node
- XK6 NP results use a new MPI process to grid mapping and a separate partition for long-range electrostatics (as do the GPU results)

Brown, W. M., Nguyen, T.D., Fuentes-Cabrera, M., Fowlkes, J. D., Rack, P. D., Berger, M., Bland, A. S. A**n Evaluation of Molecular Dynamics Performance on the Hybrid Cray XK6 Supercomputer**. *Proceedings of the ICCS Workshop on Large Scale Computational Physics*. 2012. Published in *Procedia Computer Science*, 2012. 9 p. 186-195.

Atomic Fluid

- Atomic fluid - microcanonical ensemble, Lennard- Jones potential, reduced density 0.8442, neighbor skin 0.3σ, cutoffs of 2.5σ and 5.0σ

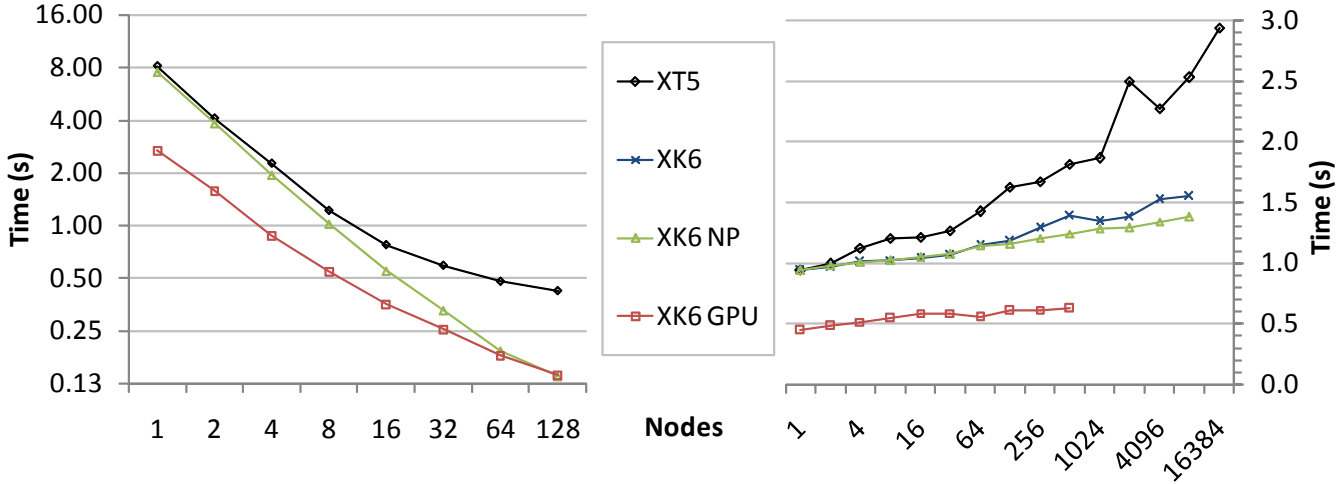


XK6 Single Node: 1.17X
XK6 GPU Single Node: 3.03X, 5.68X

XK6 Single Node: 1.04X
XK6 GPU Single Node: 1.92X, 4.33X
XK6 GPU 512 Nodes: 2.92X, 5.11X
NP gives up to 20% improvement

Bulk Copper

- Copper metallic solid in the microcanonical ensemble. The force cutoff is 4.95Å with a neighbor skin of 1.0A.
  - Requires additional ghost exchange during force calculation for electron densities

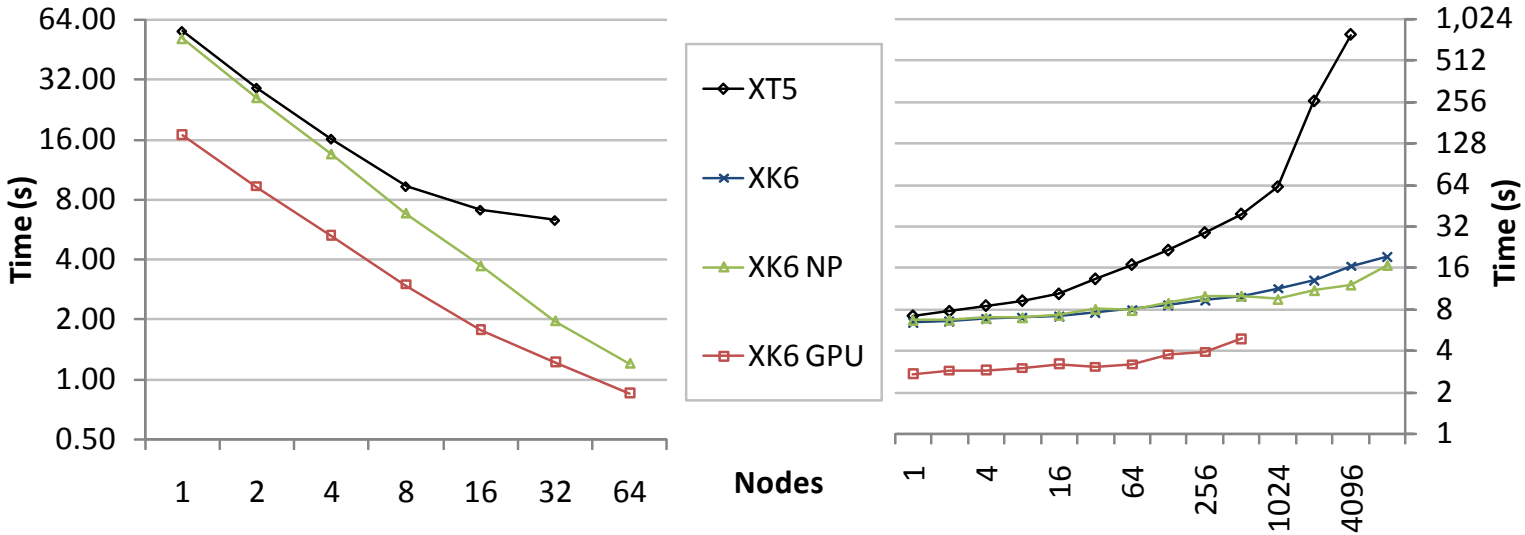


XK6 Single Node: 1.07X
XK6 GPU Single Node: 3.05X

XK6 Single Node: 1.0X
XK6 GPU Single Node: 2.12X
XK6 GPU 512 Nodes: 2.90X
NP gives up to 12% improvement

Protein

- All-atom rhodopsin protein in a solvated lipid bilayer with the CHARMM force field. Isothermal-isobaric ensemble, SHAKE constraints, 2.0fs timestep. Counter-ions, water, 8Å/ 10Å cutoff
  - Long range electrostatics are calculated with P3M.

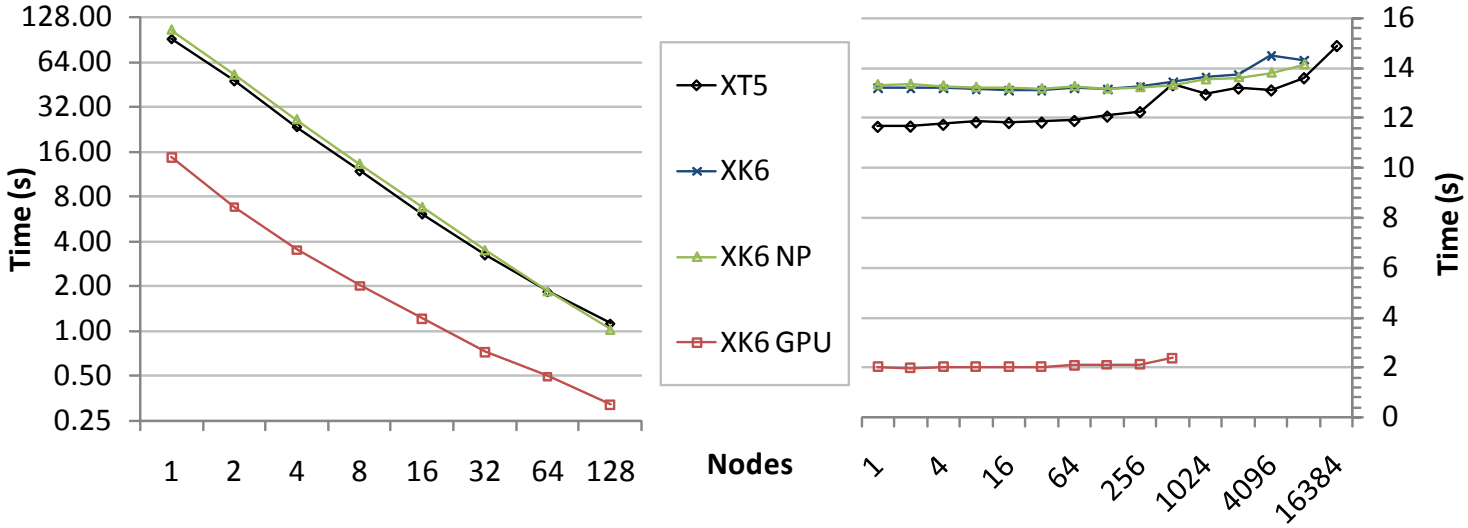


XK6 Single Node: 1.1X
XK6 GPU Single Node: 3.3X

XK6 Single Node: 1.1X
XK6 GPU Single Node: 2.6X
XK6 GPU 512 Nodes: 8X
NP gives up to 27% improvement

Liquid Crystal

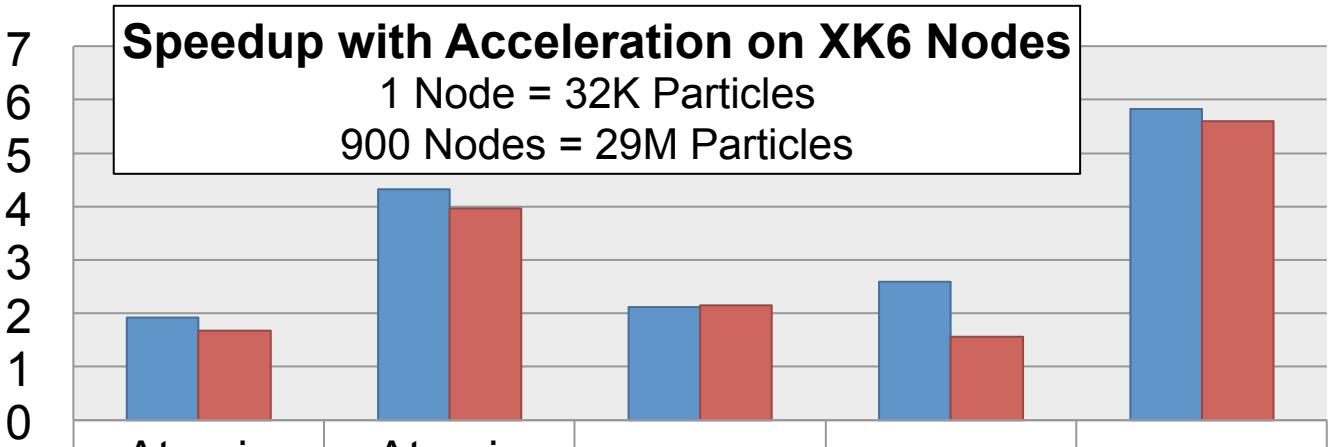
- Liquid crystal mesogens are represented with biaxial ellipsoid particles, Gay-Berne potential, isotropic phase, isothermal-isobaric ensemble, 4σ cutoff with a 0.8σ neighbor skin
  - High arithmetic intensity



XK6 Single Node: .9X
XK6 GPU Single Node: 6.23X

XK6 Single Node: .9X
XK6 GPU Single Node: 5.82X
XK6 GPU 512 Nodes: 5.65X

XK6 vs XK6+GPU Benchmarks

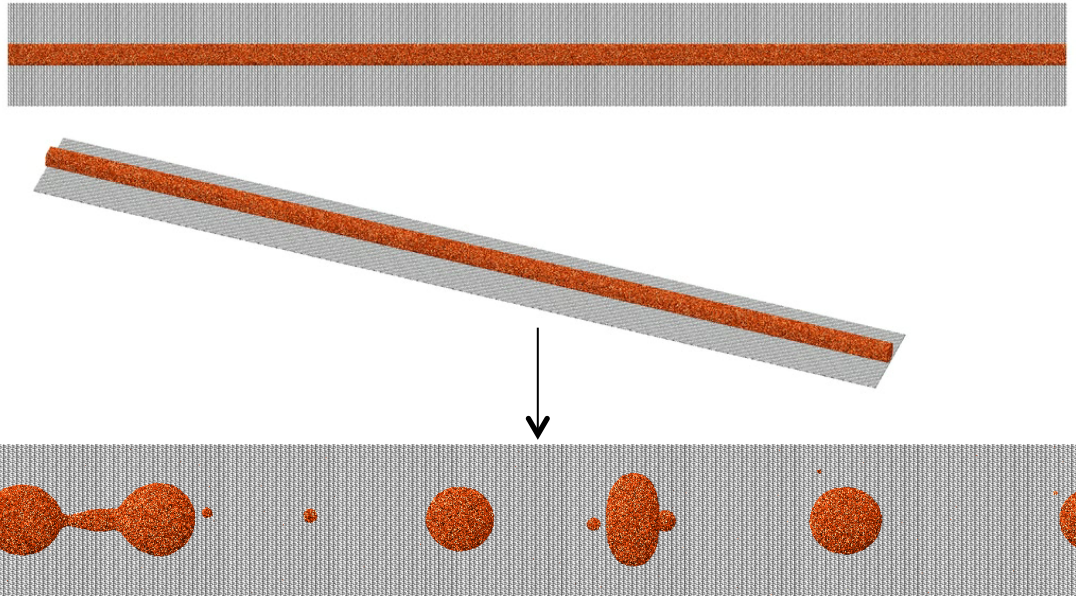


	Atomic Fluid (cutoff = 2.5σ)	Atomic Fluid (cutoff = 5.0σ)	Bulk Copper	Protein	Liquid Crystal
Speedup (1 Node)	1.92	4.33	2.12	2.6	5.82
Speedup (900 Nodes)	1.68	3.96	2.15	1.56	5.60

Example Science Problems 1.

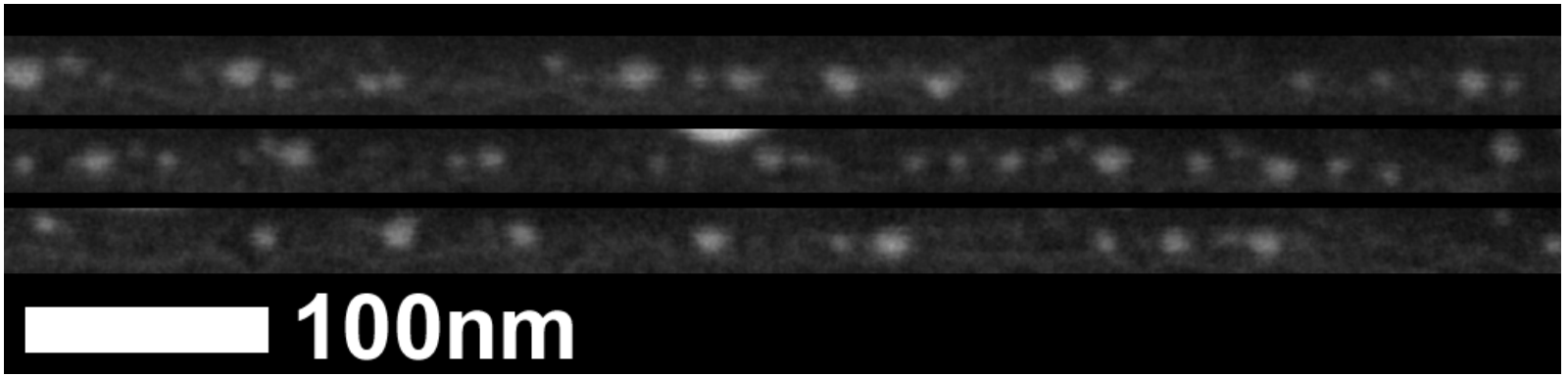
- Controlling the movement of nanoscale objects is a significant goal of nanotechnology.
- Pulsed laser melting offers a unique opportunity to dictate materials assembly where rapid heating and cooling rates and ns melt lifetimes are achievable
- Using both experiment and theory we have investigated ways of controlling how the breakage occurs so as to control the assembly of metallic nanoparticles
- Here, we illustrate MD simulation to investigate the evolution of the Rayleigh-Plateau liquid instability for copper lines deposited on a graphite substrate
- Simulations are performed with GPU acceleration on Jaguar at the same scales as experiment

11.4M Cu Atom Simulations on Graphitic Substrate



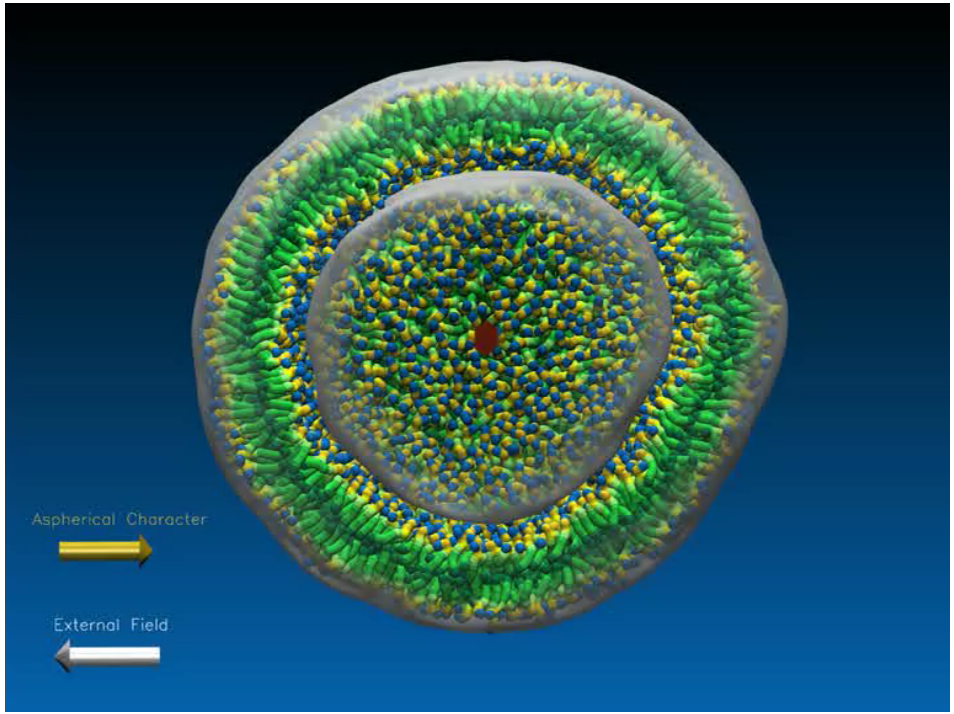
2.7X Faster than 512 XK6 w/out Accelerators

100 nm



Example Science Problems 2.

- Membrane fusion, which involves the merging of two biological membranes in a controlled manner, is an integral part of the normal life cycle of all living organisms.
- Viruses responsible for human disease employ membrane fusion as an essential part of their reproduction cycle.
- Membrane fusion is a critical step in the function of the nervous system
  - Correct fusion dynamics requires realistic system sizes
- Kohlmeyer / Klein INCITE Award

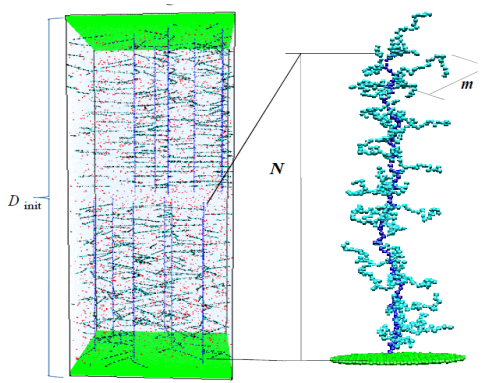


39M Particle Liposome System2.7X Faster than 900 XK6 w/out Accelerators

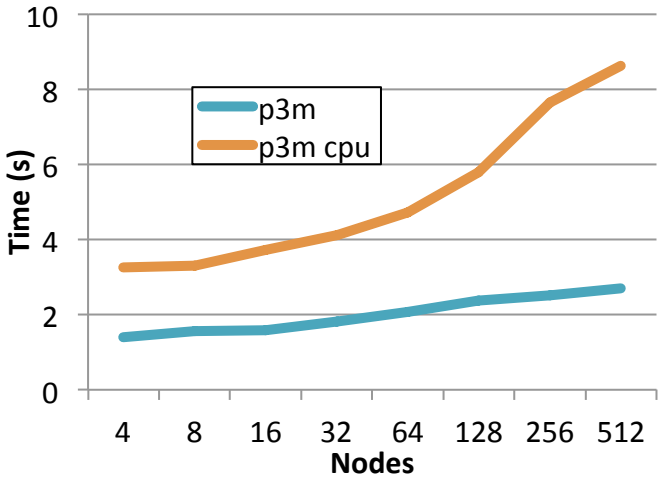
The “Porting Wall”

- Although you can get performance improvements by porting existing models/algorithms and running simulations using traditional parameterizations…
  - There is a limit to the amount of parallelism that can be exposed to decrease the time-to-solution
  - Increasingly desirable to re-evaluate computational physics methods and models with an eye towards approaches that allow for increased concurrency and data locality
    - Parameterize simulations to shift work towards routines well-suited for the accelerator
    - Methods/models with increased computational requirements can perform better if they can increase concurrency, allow for larger time-steps, etc.
- Computational scientists will play a critical role in exploiting the performance of current and next-generation architectures
- Some very basic examples…

Electrostatics Example with Polyelectrolyte Brushes



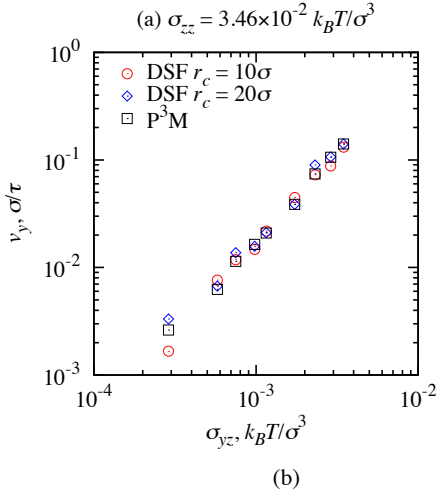
- Electrostatics are typically solved by splitting the Coulomb potential into a short-range potential that decays to zero at the cutoff and a long-range potential that converges rapidly in k-space
  - Long-range component is typically solved with discretization on a mesh
    - Poisson solve with 3D FFTs
    - Communications bottleneck
  - The traditional parameterizations that work well on older high-performance computers are not necessarily optimal for many simulations
  - Shift the work towards the short-range component
    - Disproportionate performance improvements on accelerator with larger short-range cutoffs
    - Benefits from acceleration **improve** at larger node counts!

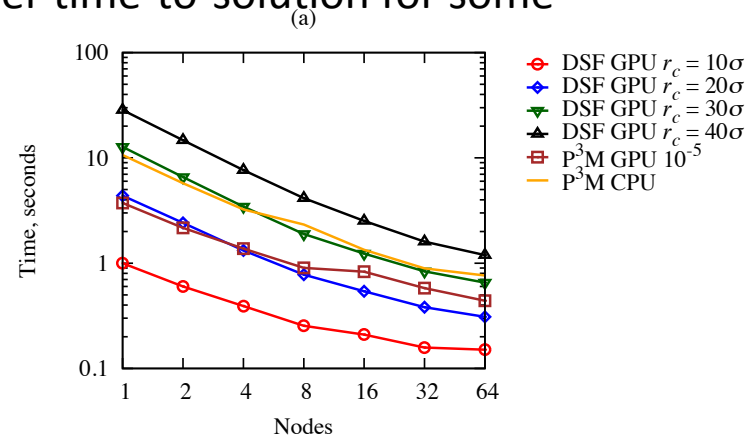


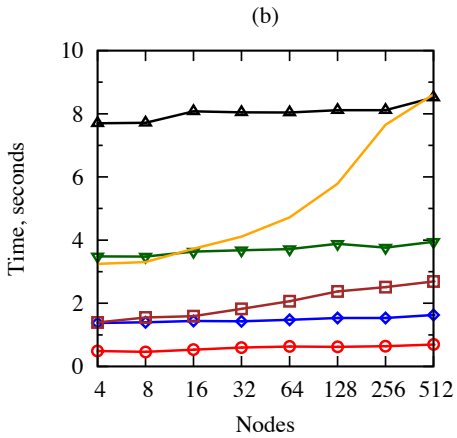
Nguyen, T.D., Carrillo, J.-M., Dobrynin, A.V., Brown, W.M. **A Case Study of Truncated Electrostatics for Simulation of Polyelectrolyte Brushes on GPU Accelerators**. *Journal of Chemical Theory and Computation, In press.*

Electrostatics Example with Polyelectrolyte Brushes

- Eliminate the long-range component?
  - In condensed phase systems, charges can be effectively short-range due to screening from other charged particles in the system
  - Use a large short range cutoff, correct for issues due to non-neutral spherical cutoff regions, and modify cell list calculation to be efficient for large cutoffs in parallel simulations
  - Accurate results with faster time-to-solution for some simulations



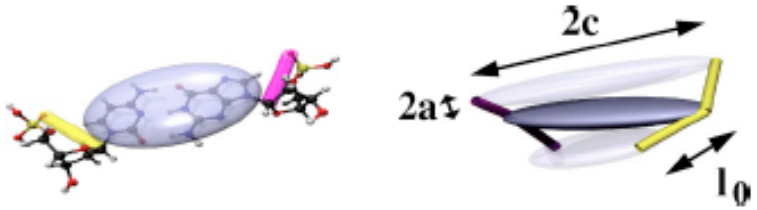




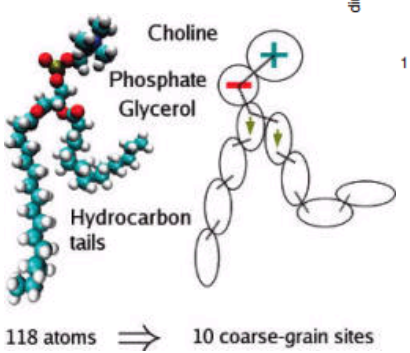
Nguyen, T.D., Carrillo, J.-M., Dobrynin, A.V., Brown, W.M. **A Case Study of Truncated Electrostatics for Simulation of Polyelectrolyte Brushes on GPU Accelerators**. *Journal of Chemical Theory and Computation, In Press.*

Alternative Potential Energy Models?

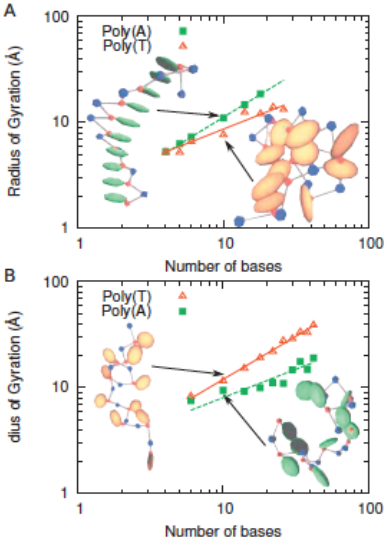
- Implementation of new models that can exploit high peak accelerator performance to improve accuracy and sampling
  - Computational cost of a single ellipsoid-ellipsoid interaction can be 15x that for Lennard-Jones on the CPU
  - With GPU acceleration, it is more competitive
    - Higher arithmetic intensity, so better speedup when compared to LJ acceleration
    - Better parallel efficiency *relative* to the CPU
      - Still get performance with fewer threads



Mergell, B., Ejtehadi, M.R., Everaers, R., PRE, 68, 021911 (2003)

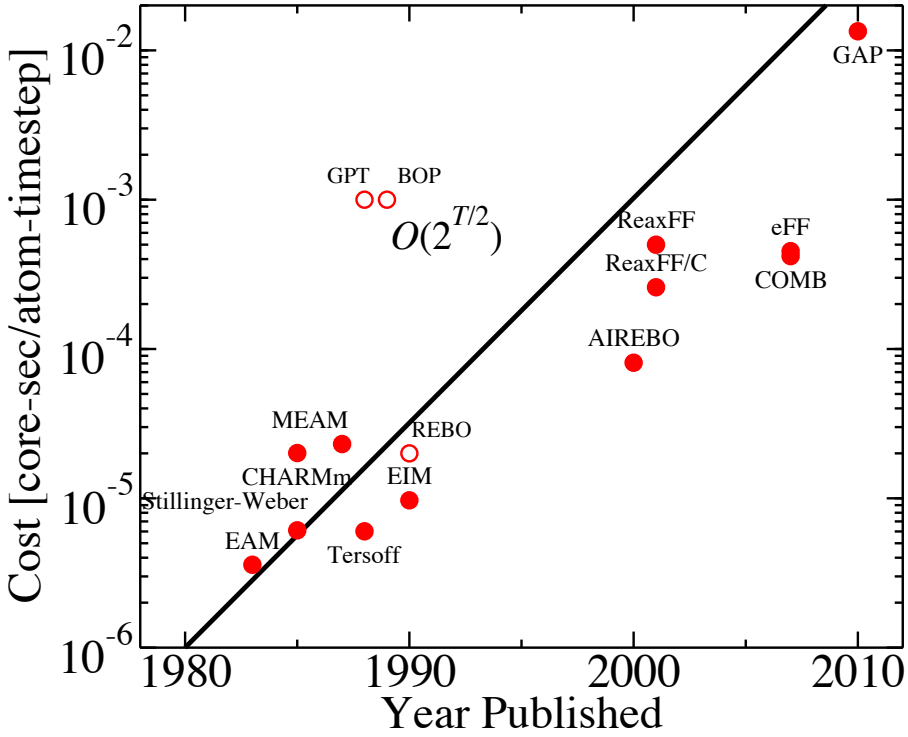


Orsi, M. et al, J. Phys. Chem. B, Vol. 112, No. 3, 2008



Morriss-Andrews, A., Rottler J., Plotkin S. S., JCP 132, 035105, 2010.

Moore's Law for Potentials



More Creative Examples…

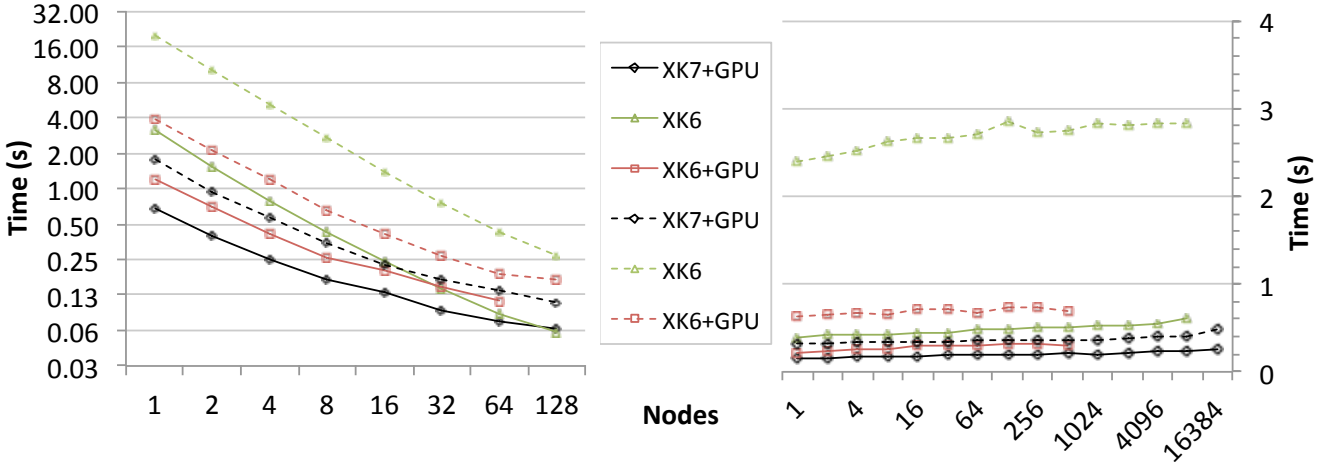
- Advanced time integrators, time parallelism, etc…

Kepler II vs Fermi +

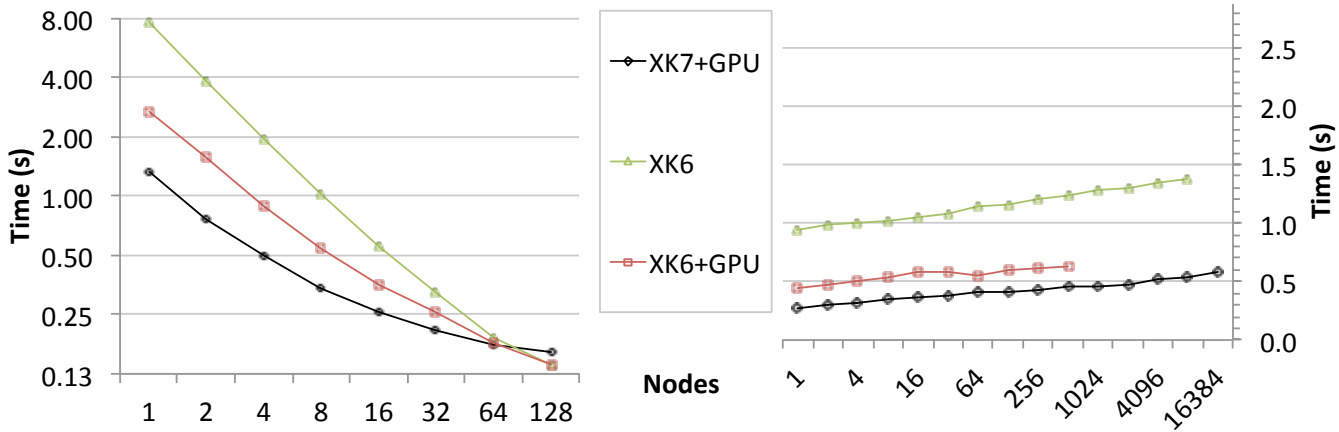
- More cores
  - Efficiently handle larger problems, better performance for complex models
- Improved thread performance
  - Faster time to solution for same problem
- Larger register file
  - Better performance for complex models such as the liquid crystal case
- Warp shuffle
  - Reduced shared memory usage for reductions across threads
- Hyper-Q
  - MPI processes sharing the GPU can share a context
    - Reduced memory overhead per process
    - Concurrent kernel execution from multiple processes

Early Kepler Benchmarks on Titan

Atomic Fluid

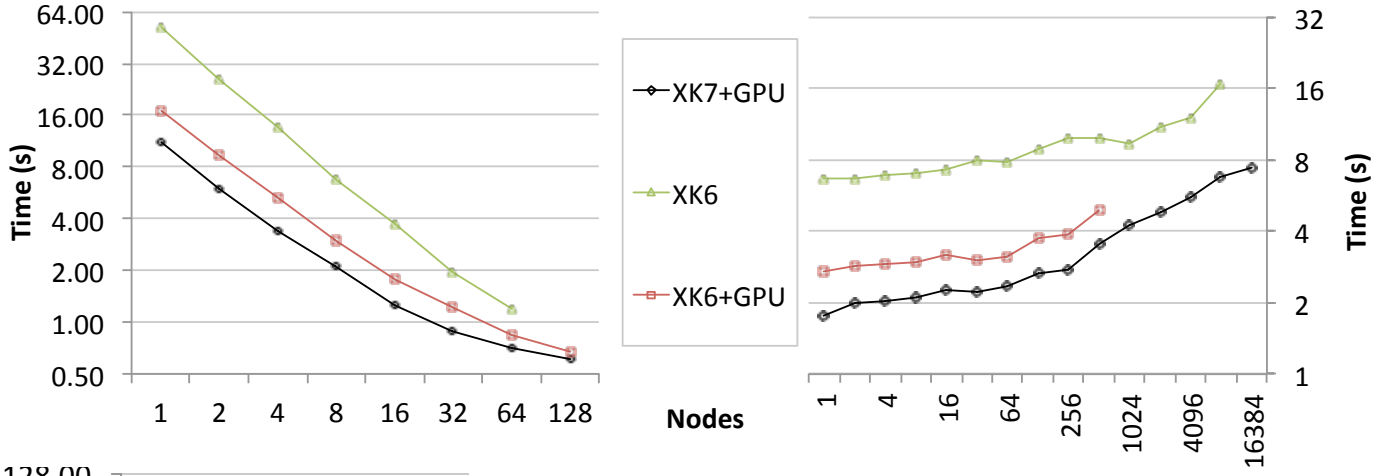


Bulk Copper

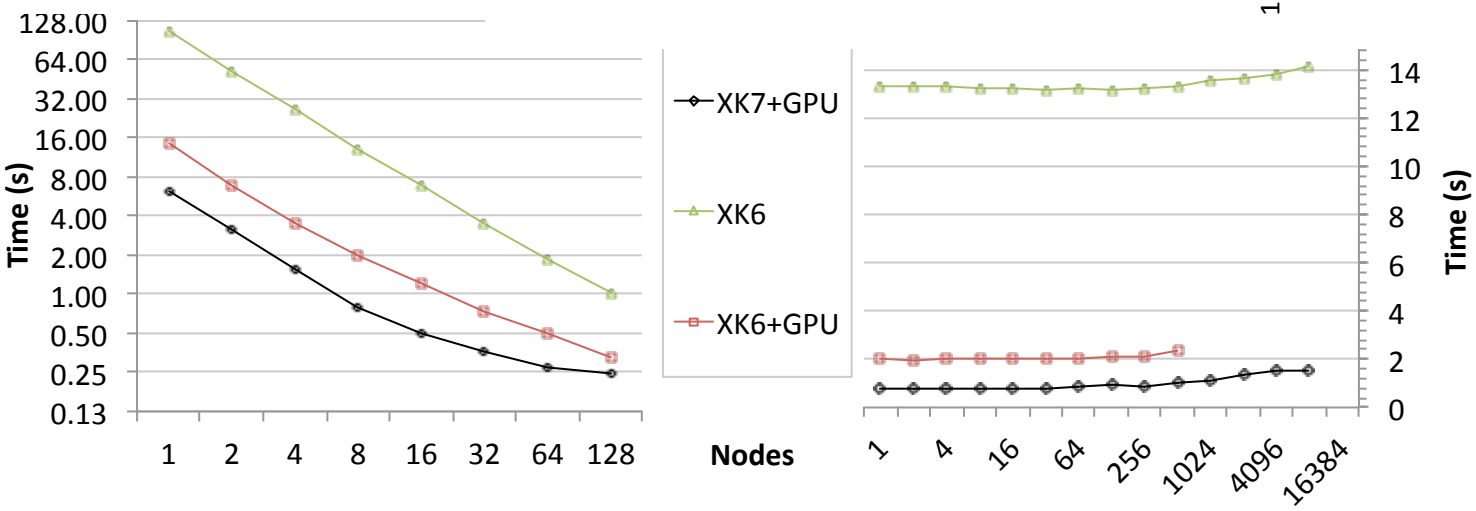


Early Kepler Benchmarks on Titan

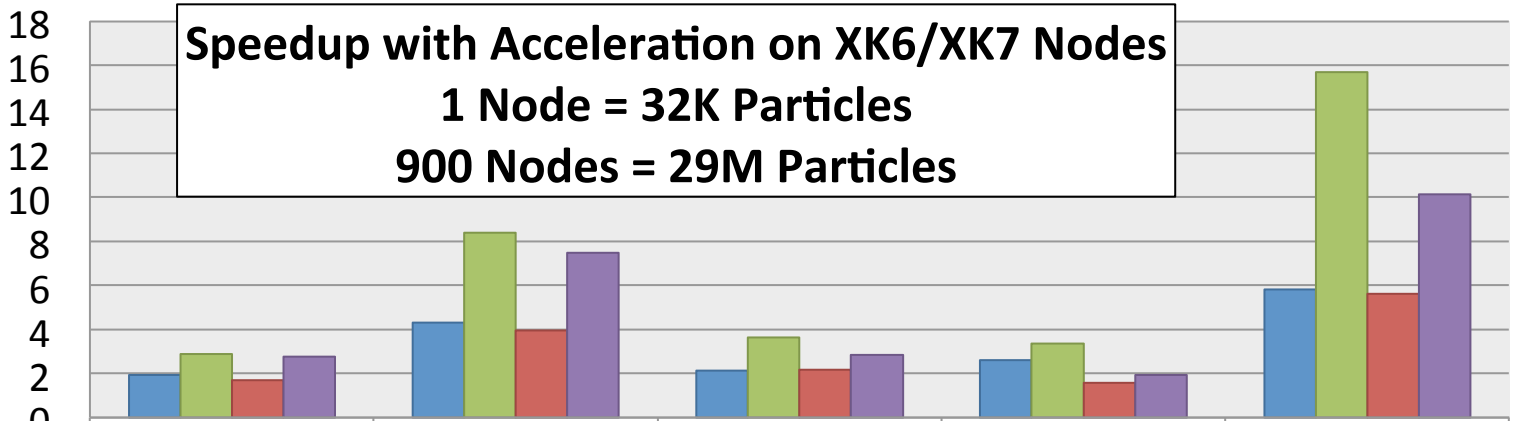
Protein



Liquid Crystal



Early Titan XK6/XK7 Benchmarks



	Atomic Fluid (cutoff = 2.5σ)	Atomic Fluid (cutoff = 5.0σ)	Bulk Copper	Protein	Liquid Crystal
XK6 (1 Node)	1.92	4.33	2.12	2.6	5.82
XK7 (1 Node)	2.90	8.38	3.66	3.36	15.70
XK6 (900 Nodes)	1.68	3.96	2.15	1.56	5.60
XK7 (900 Nodes)	2.75	7.48	2.86	1.95	10.14

Ongoing Work

- Implementation and evaluation of alternative algorithms for long-range electrostatics.
  - Multigrid(-like) Methods for Poisson Solve, etc.
    - O(N), no FFTs
- Implementation of complex models well suited for accelerators
- Improvements driven by specific science problems

Publications

- Brown, W.M., Wang, P. Plimpton, S.J., Tharrington, A.N. **Implementing Molecular Dynamics on Hybrid High Performance Computers – Short Range Forces.** *Computer Physics Communications.* 2011. 182: p. 898-911.
- Brown, W.M., Kohlmeyer, A., Plimpton, S.J., Tharrington, A.N. **Implementing Molecular Dynamics on Hybrid High Performance Computers – Particle-Particle Particle-Mesh.** *Computer Physics Communications.* 2012. 183: p. 449-459.
- Brown, W. M., Nguyen, T.D., Fuentes-Cabrera, M., Fowlkes, J. D., Rack, P. D., Berger, M., Bland, A. S. A**n Evaluation of Molecular Dynamics Performance on the Hybrid Cray XK6 Supercomputer**. *Proceedings of the ICCS Workshop on Large Scale Computational Physics*. 2012. Published in *Procedia Computer Science*, 2012. 9 p. 186-195.
- Nguyen, T.D., Carrillo, J.-M., Dobrynin, A.V., Brown, W.M. **A Case Study of Truncated Electrostatics for Simulation of Polyelectrolyte Brushes on GPU Accelerators**. *Journal of Chemical Theory and Computation.* In Press.

Acknowledgements

- LAMMPS
  - Steve Plimpton (SNL) and many others
- LAMMPS Accelerator Library
  - W. Michael Brown (ORNL), Trung Dac Nguyen (ORNL), Peng Wang (NVIDIA), Axel Kohlmeyer (Temple), Steve Plimpton (SNL), Inderaj Bains (NVIDIA)
- Geryon Library
  - W. Michael Brown (ORNL), Manuel Rodriguez Rodriguez (ORNL), Axel Kohlmeyer (Temple)
- K-Space Partitions
  - Yuxing Peng and Chris Knight (University of Chicago)
- Metallic Nanoparticle Dewetting
  - Miguel Fuentes-Cabrera (ORNL), Trung Dac Nguyen (ORNL), W. Michael Brown (ORNL), Jason Fowlkes (ORNL), Philip Rack (UT, ORNL)
- Lipid Vesicle Science and Simulations
  - Vincenzo Carnevale (Temple), Axel Kohlmeyer (Temple), Michael Klein (Temple), et. al.
- Enhanced Truncation/Bottlebrush Simulations
  - Trung Dac Nguyen (ORNL), Jan-Michael Carrillo (UC), Andrew Dobrynin (UC), W. Michael Brown (ORNL)
- Multilevel Summation
  - Arnold Tharrington (ORNL)
- NVIDIA Support
  - Carl Ponder, Mark Berger