GPU Acceleration in LAMMPS

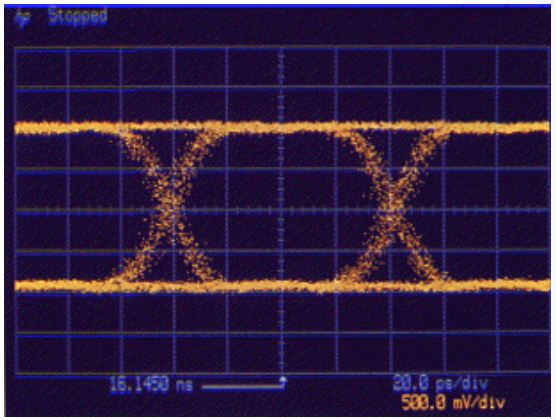
W. Michael Brown
National Center for Computational Sciences
Oak Ridge National Labs
August 10, 2011

WHY DO WE CARE ABOUT GPUS?

Clock Rates are not Increasing

Power = Capacitance * Frequency * Voltage^2 + Leakage

- Traditionally, as Frequency increased, Voltage decreased, keeping the total power in a reasonable range
- But we have run into a wall on voltage
  - As the voltage gets smaller, the difference between a “one” and “zero” gets smaller. Lower voltages mean more errors.
- Capacitance increases with the complexity of the chip
- Total power dissipation is limited by cooling



Power to move data

Energy_to_move_data = bitrate * length^2 / cross_section_area_of_wire

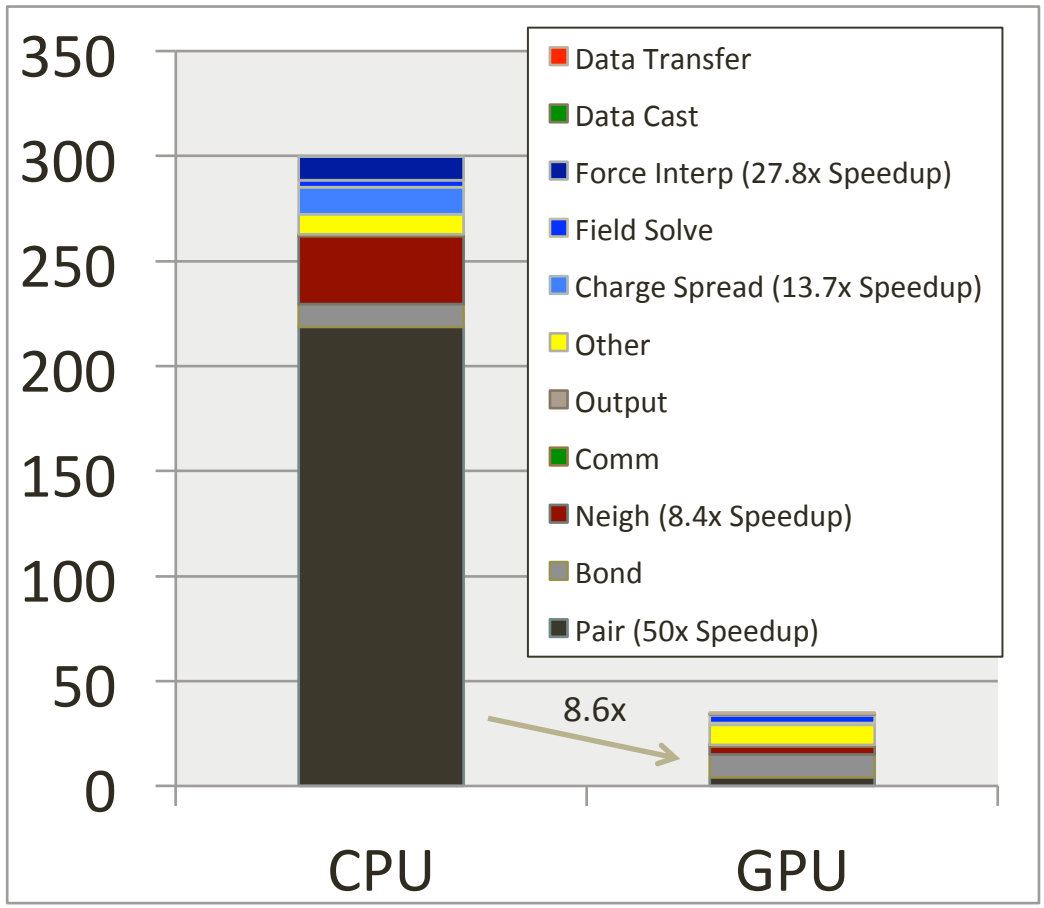
- The energy consumed increases proportionally to the bit-rate, so as we move to ultra-high-bandwidth links, the power requirements will become an increasing concern.
- The energy consumption is highly distance-dependent (the square of the length term), so bandwidth is likely to become increasingly localized as power becomes a more difficult problem.
- Improvements in chip lithography (making smaller wires) will not improve the energy efficiency or data carrying capacity of electrical wires.

D. A. B. Miller and H. M. Ozaktas, "Limit to the Bit-Rate Capacity of Electrical Interconnects from the Aspect Ratio of the System Architecture," Journal of Parallel and Distributed Computing, vol. 41, pp. 42-52 (1997) article number PC961285.

Implications for Future Systems

- Clock rates will stay largely the same as today, need to increase the parallelism of systems to improve performance
- Energy cost of moving data is very large. We will have to explicitly manage data locality to limit power consumption.

- GPUs already exploit massive parallelism to achieve high peak performance, requiring a programming model that enforces good use of data localization in order to achieve performance with low electrical power
- Most of the important concepts in GPU programming are important for shared-memory programming on multi/many core CPU chips
- GPUs can be used today to reduce electrical power, space, cooling demands, and operating system images in HPC



GPU Acceleration Currently Available for:

- Neighbor list builds
- Particle-Particle Particle-Mesh
- Non-bonded short-range potentials

GPU ACCELERATION IN LAMMPS

Accelerated Non-bonded Short-Range Potentials

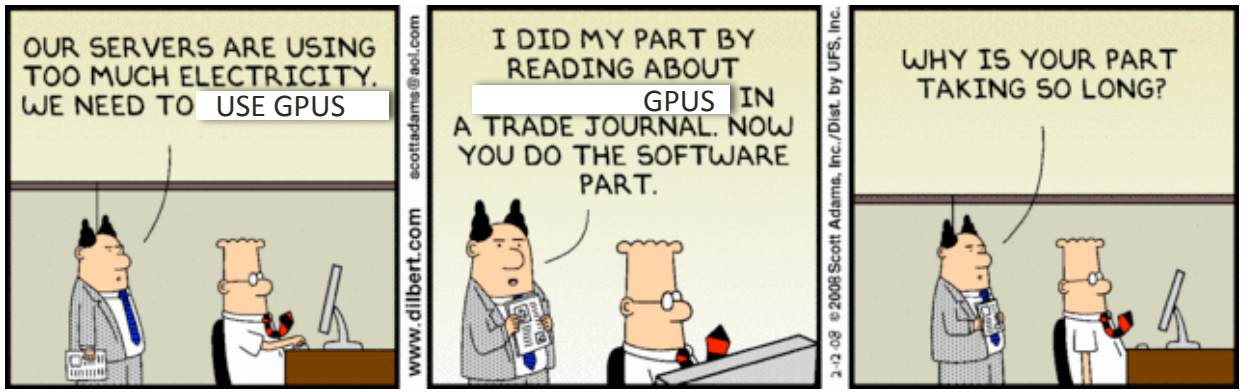
- Single, mixed, and double precision support for:
  - lj/cut
  - lj96/cut
  - lj/expand
  - lj/cut/coul/cut
  - lj/cut/coul/long
  - lj/charmm/coul/long
  - lj/class2
  - lj/class2/coul/long
  - morse
  - cg/cmm
  - cg/cmm/coul/long
  - coul/long
  - gayberne
  - resquared

Neighbor List Builds

- Acceleration is Optional
  - *Can't* use neighbor list builds on the accelerator with triclinic box or with "hybrid" models that use multiple accelerated pair styles
  - *Don't* use neighbor list builds on the accelerator with hybrid pair styles or when a compute or fix requires a neighbor list
- Neighbor list builds use a sort to order atoms
  - This results in additional overhead, but gives deterministic results
- Speedup for neighbor-list builds with acceleration is currently highly dependent on the number of particles

Particle-Particle Particle-Mesh

- Acceleration is optional
- Can be performed in single or double precision
- Charge assignment and force interpolation routines are accelerated, Poisson solve is not
  - Accelerated routines use the same FFT library as the standard routines
  - Unlikely to see any significant improvement from acceleration for typical system sizes or parallel runs on current hardware



"All routines must be ported to the GPU in order to compete with multi-core CPUs due to Amdahl's law"

"Data transfer is the bottleneck when using GPU acceleration"

"Double precision calculations are much slower than single precision on current hardware"

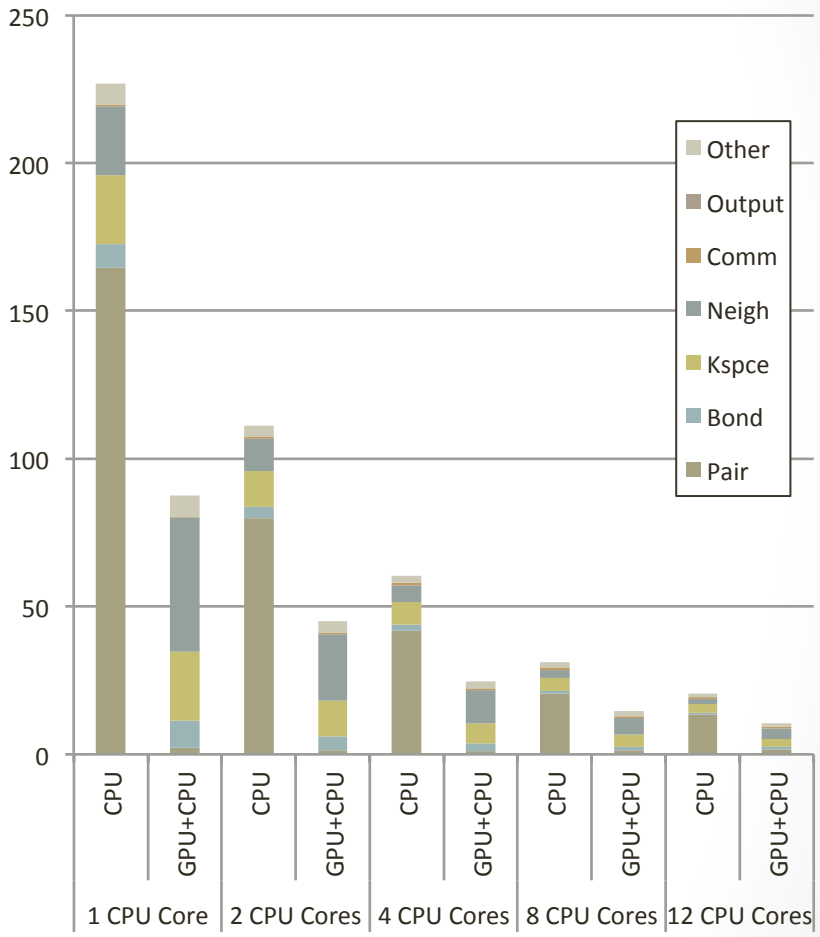
Often true, but *not* always…

UNDERSTANDING GPU ACCELERATION

(IN LAMMPS)

"All routines must be ported to the GPU in order to compete with multi-core CPUs due to Amdahl's law"

- Codes can be competitive if they parallelize non-accelerated routines on the CPU
- 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
  - Speedup vs 12 CPU cores is still > 2x because we use all 12 cores in addition to the GPU

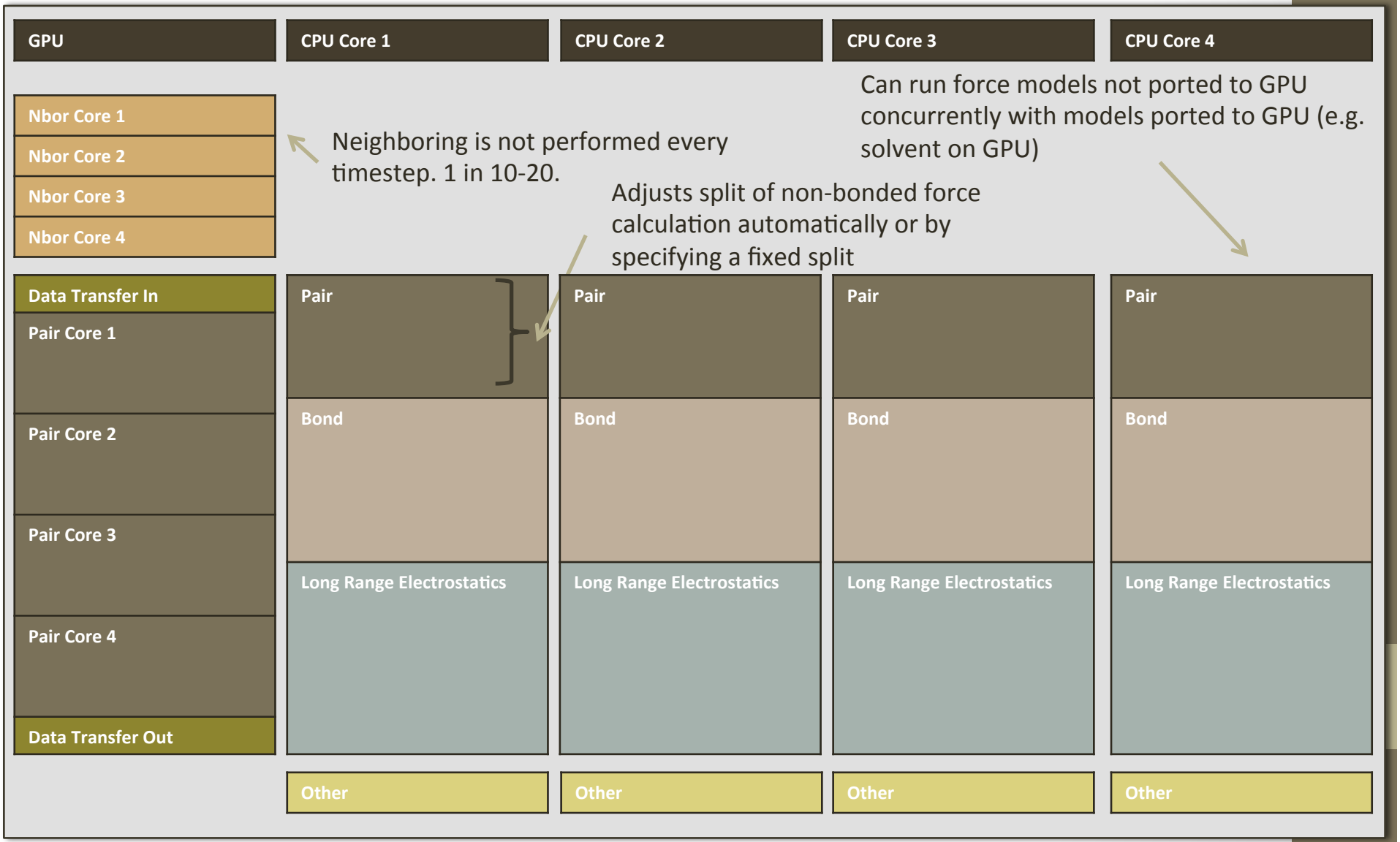


Minimizing the Code Burden

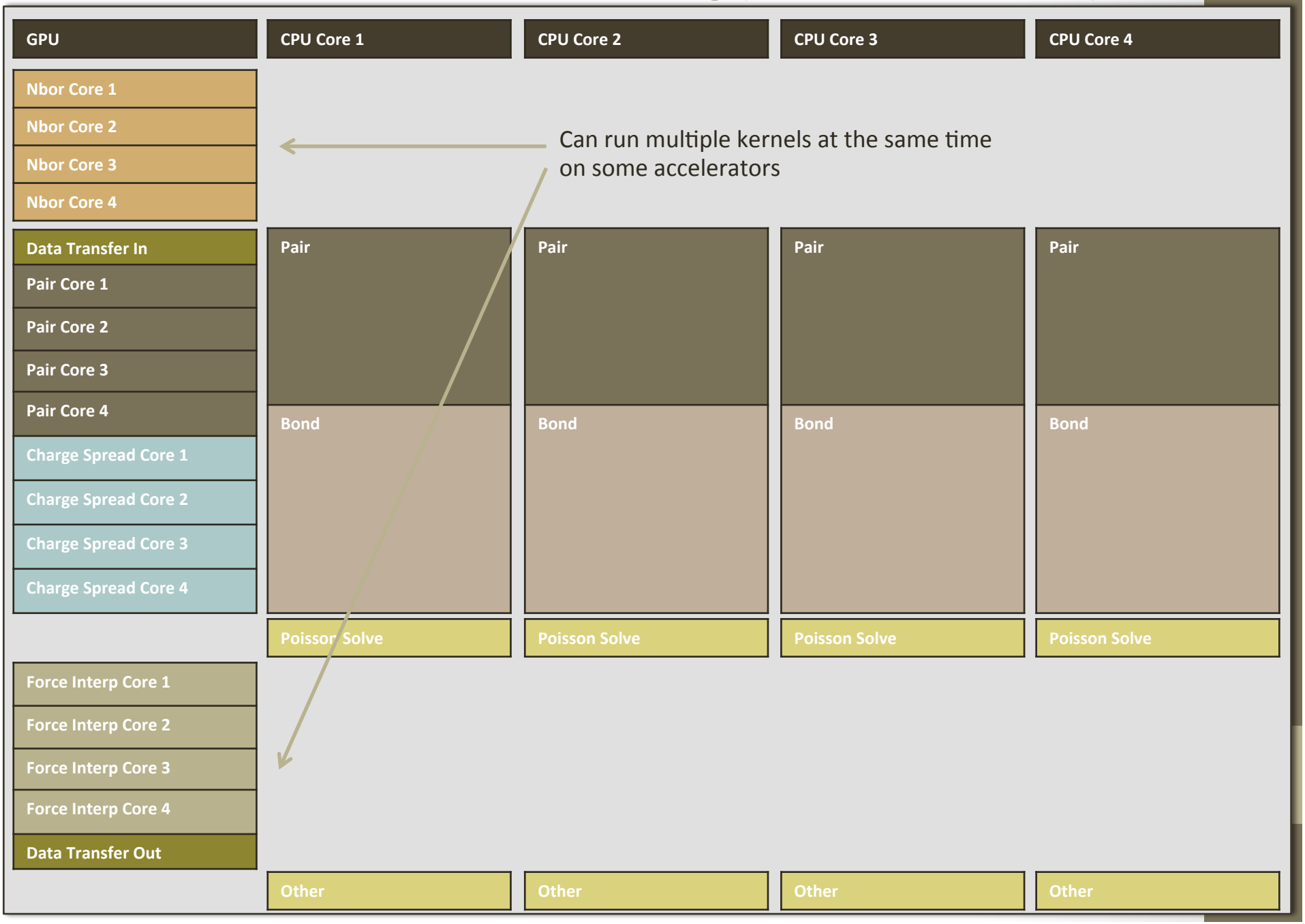
- Focus porting efforts on the routines that dominate the computational time and have high potential for speedup with accelerators
- Use concurrent CPU/GPU calculations where possible and overlap host-accelerator data transfer with computation

Host-Device Load Balancing

(- PPPM Acceleration) *Not to scale*



Host-Device Load Balancing (+PPPM Acceleration)



Minimizing the Code Burden

- Advantages
  - 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%.

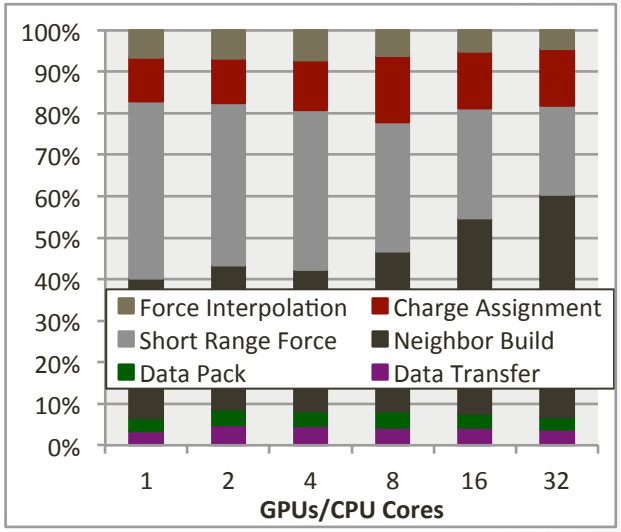
Minimizing the Code Burden

- Disadvantages
  - For some simulations, the upper bound for performance improvements due to porting additional routines and removing data transfer is as high as 50% (for up to 1 million particles on a single GPU with current hardware).
  - Increasing the number of MPI processes can impact interprocess communication performance
  - Multiple MPI processes sharing a GPU currently results in the execution of more kernels, each with a smaller amount of work
  - Parallelization of the CPU routines with OpenMP would likely be a significant effort for efficient simulations in LAMMPS.

USER-CUDA package is now available in LAMMPS as an alternative library for acceleration. In some cases, the entire simulation can be run on the GPU.

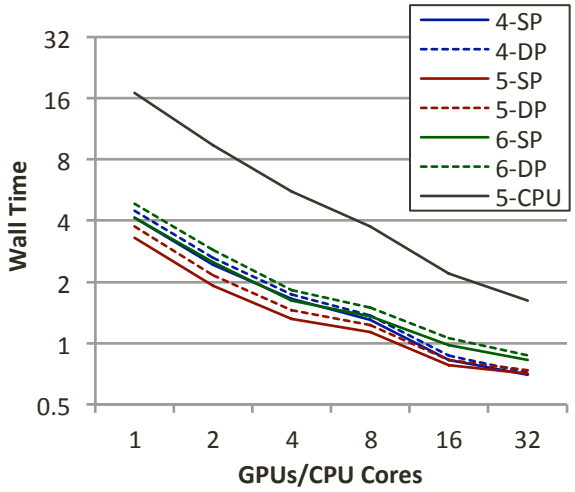
"Data transfer is the bottleneck when using GPU acceleration"

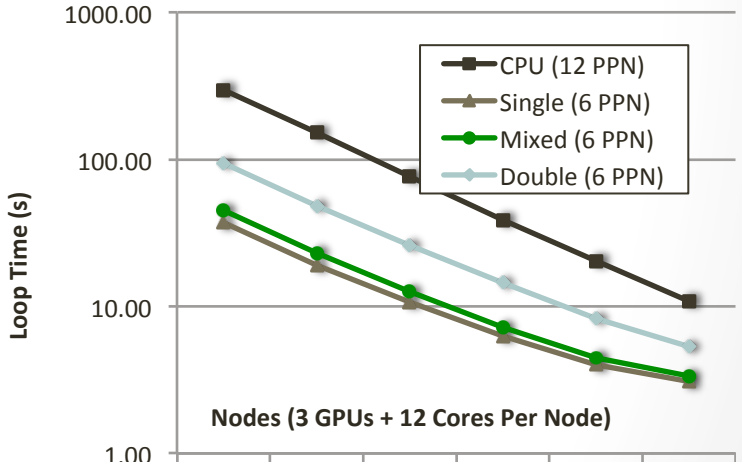
- If neighbor list builds are performed on the GPU, data transfer can be a small fraction of the total simulation time
  - For this reason, accelerated neighbor list builds can be important despite the relatively poor performance on the GPU
  - For the rhodopsin benchmark, data transfer is less than 6% of the *GPU* calculation times
    - Will be a smaller percentage of the total simulation time because the calculation of bonded forces, time integration with a thermostat/barostat, SHAKE, and the Poisson solve must be calculated somewhere and we need to do MPI comm
  - In some cases, can benefit from overlapping data transfer with computation



"Double precision calculations are much slower than single precision on current hardware"

- Less double precision ALUs on current hardware
- Full double precision is usually significantly slower for most kernels
- Mixed precision can give very similar performance
- Double precision performance can be more similar for latency-bound kernels (PPPM).





	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

CUDA

OpenCL

Libraries

Directives

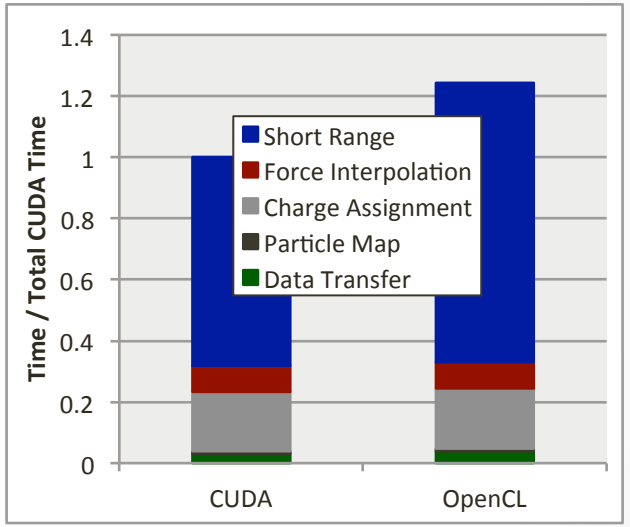
…

ACCELERATOR PROGRAMMING MODEL/API

Geryon Library

- Allows same code to compile with CUDA Runtime, CUDA Driver, and OpenCL APIs
- Allows portability to AMD accelerators, CPUs, and any future chips with OpenCL support
- Currently supports all kernels except for neighbor list builds
  - This will probably be added soon

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



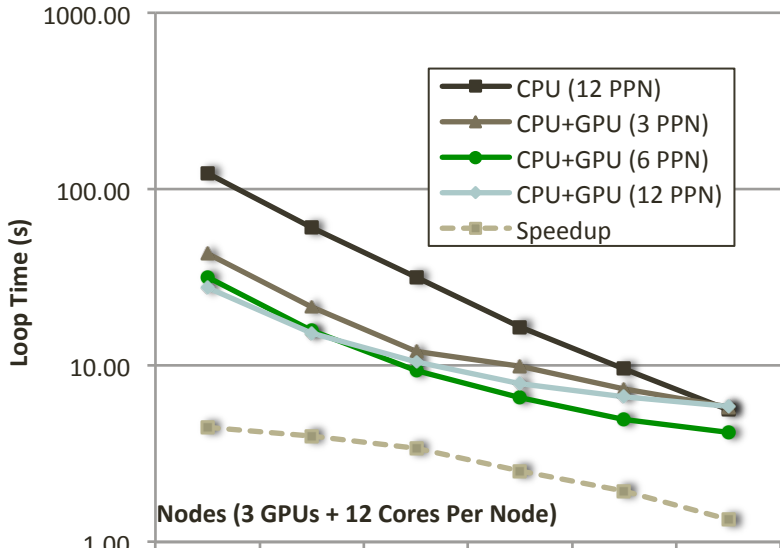
Can't give a single number for performance improvements to expect from acceleration

- Depends on the model, the system, and the simulation parameters
- Depends on the hardware, CPUs, the type of accelerator, and the interconnect bandwidths
- Even on the same GPU, results can differ by up to 30% depending on the ECC configuration
- Following are results from standard LAMMPS benchmarks and some science applications on the Keeneland GPU Cluster
  - 2 Intel Westmere 2.8 GHz Hex Core CPUs
  - 3 Tesla M2070 GPUs, ECC On, QDR Infiniband, 120 Nodes
  - Input files available at: http://users.nccs.gov/~wb8/gpu/keeneland.htm

BENCHMARKS

Atomic Lennard-Jones Fluid
(256k atoms)

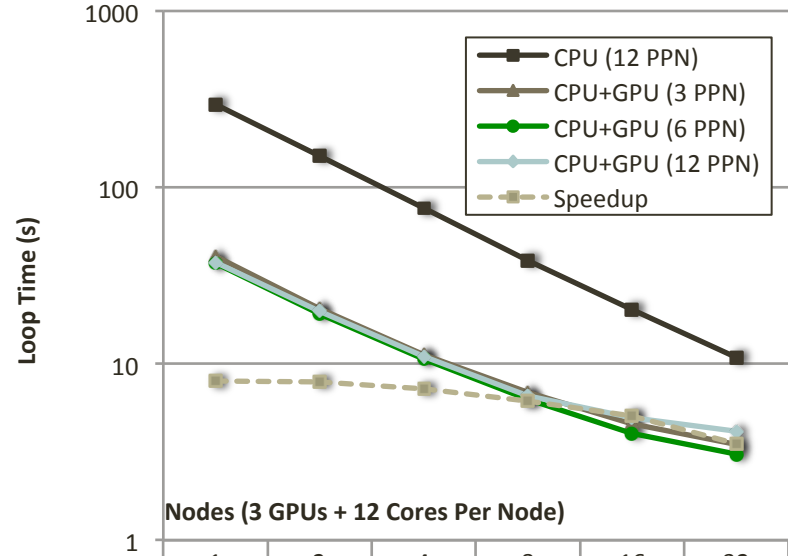
- Good lower bound to LAMMPS performance improvements
  - Low arithmetic intensity, small neighbor lists



	1	2	4	8	16	32
CPU (12 PPN)	123.03	60.93	31.80	16.59	9.62	5.61
CPU+GPU (3 PPN)	43.42	21.53	12.09	9.91	7.34	5.84
CPU+GPU (6 PPN)	31.69	15.79	9.35	6.57	4.94	4.18
CPU+GPU (12 PPN)	27.54	15.25	10.48	7.94	6.68	5.85
Speedup	4.47	4.00	3.40	2.53	1.95	1.34

Atomic Lennard-Jones Fluid
(256k atoms)

- For problems requiring a larger cutoff, results are much better
  - Here, the cutoff is increased from 2.5 sigma to 5.0 sigma

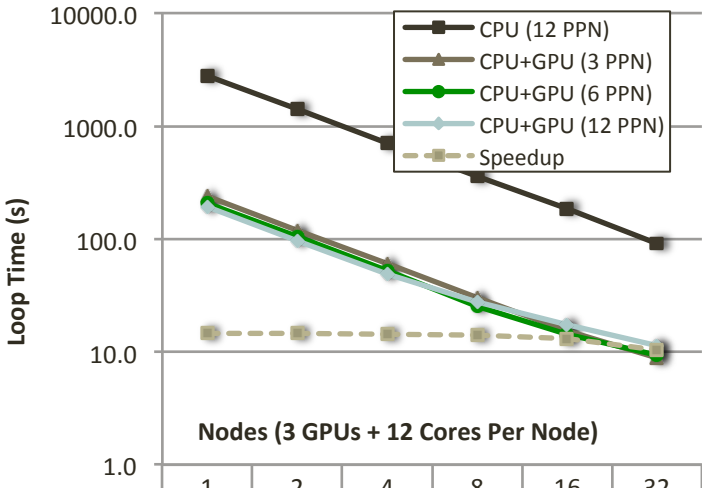


	1	2	4	8	16	32
CPU (12 PPN)	297.796	151.528	76.7366	38.7134	20.4366	10.83
CPU+GPU (3 PPN)	40.68	20.56	11.29	6.86	4.57	3.49
CPU+GPU (6 PPN)	37.44	19.16	10.66	6.31	4.02	3.08
CPU+GPU (12 PPN)	37.67	19.79	10.96	6.62	4.93	4.15
Speedup	7.95	7.91	7.20	6.14	5.09	3.52

Ellipsoidal Particles

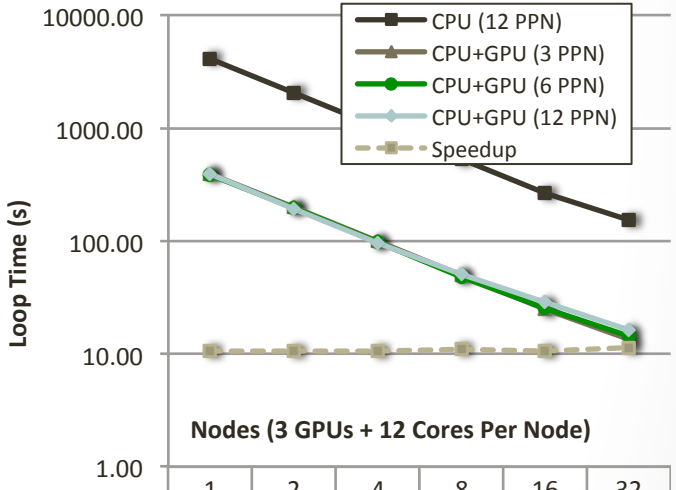
(256k particles)

- Good upper bound to LAMMPS performance improvements with acceleration
  - High arithmetic intensity, transcendentals, pairwise force calculation dominates the simulation time



	1	2	4	8	16	32
CPU (12 PPN)	2811.9	1418.3	704.33	356.73	186.92	91.32
CPU+GPU (3 PPN)	237.29	118.76	60.26	30.12	15.28	8.79
CPU+GPU (6 PPN)	207.67	104.63	52.32	25.31	14.36	9.35
CPU+GPU (12 PPN)	194.40	96.62	48.88	27.51	17.57	11.37
Speedup	14.46	14.68	14.41	14.09	13.01	10.40

Gay-Berne

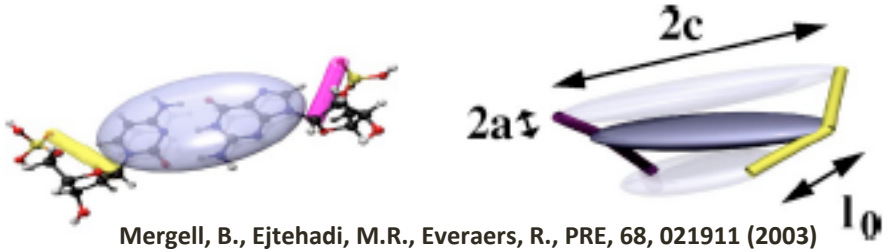


	1	2	4	8	16	32
CPU (12 PPN)	4125.08	2057.41	1025.87	533.41	264.88	153.68
CPU+GPU (3 PPN)	395.40	199.17	99.88	49.92	25.07	13.46
CPU+GPU (6 PPN)	389.47	195.69	97.93	48.32	25.49	14.39
CPU+GPU (12 PPN)	390.68	194.11	96.60	50.80	28.51	16.27
Speedup	10.59	10.60	10.62	11.04	10.57	11.42

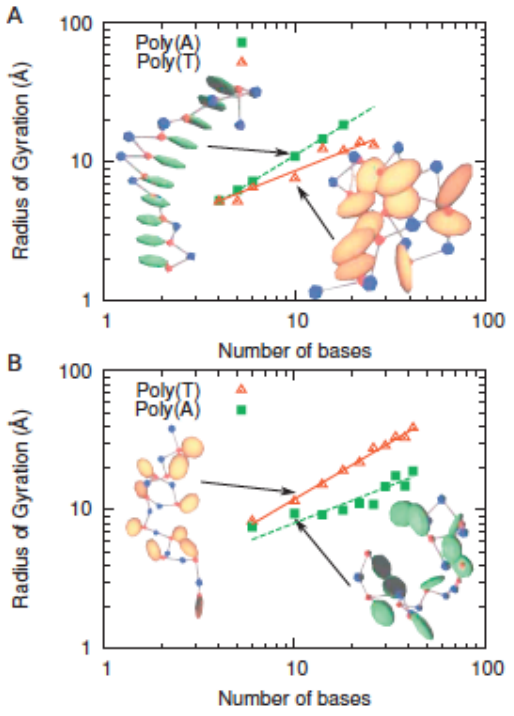
RE-Squared

Aspherical Coarse-Graining

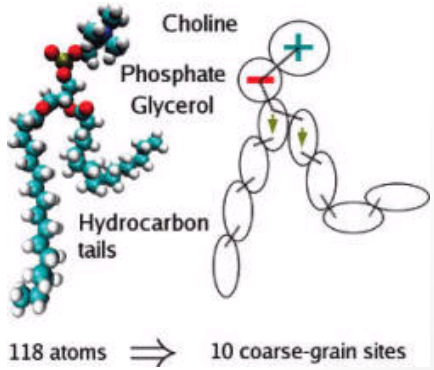
- Computational cost of a single ellipsoid-ellipsoid interaction can be 15x that for Lennard-Jones on the CPU
- With GPU acceleration, it is more competitive
  - Higher arithmetic intensity, so better speedup when compared to LJ acceleration
  - Better parallel efficiency *relative* to the CPU
    -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



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



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

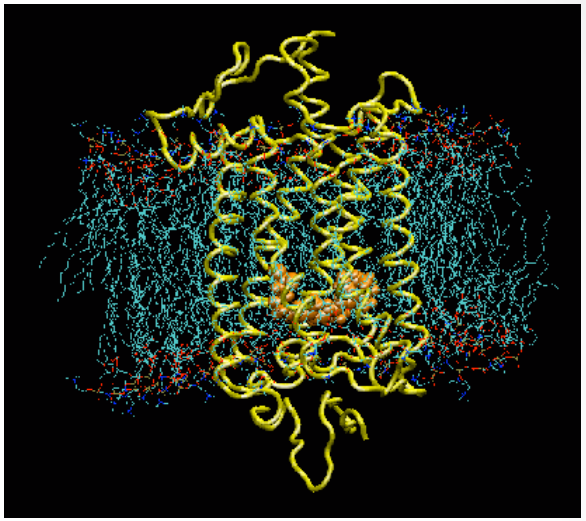


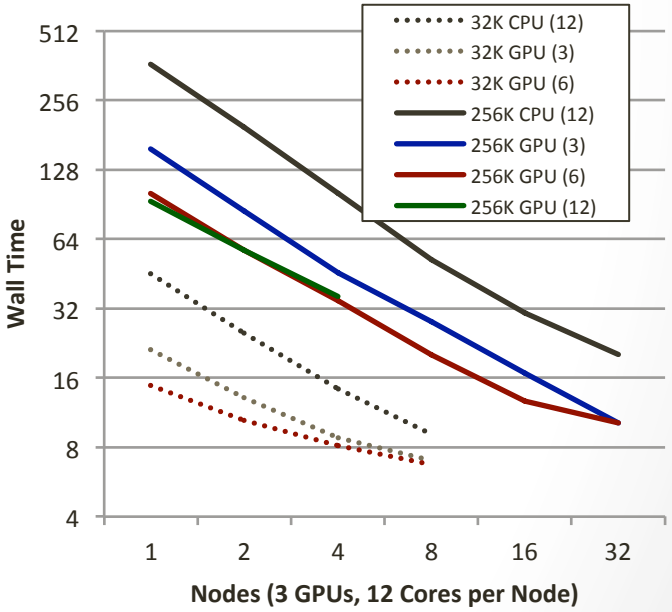
Orsi, M. et al, J. Phys. Chem. B, Vol. 112, No. 3, 2008

Rhodopsin Protein

(32K Atoms; Scaled 256K Atoms)

- CHARMM force field with a 10 angstrom cutoff
- Requires calculation of bonded terms, long range electrostatics (PPPM), and SHAKE-constrained time integration with a Nose-Hoover style thermostat/ barostat
- Poisson solve in PPPM is a larger percentage of the total simulation time and therefore can have a more significant impact on parallel efficiency

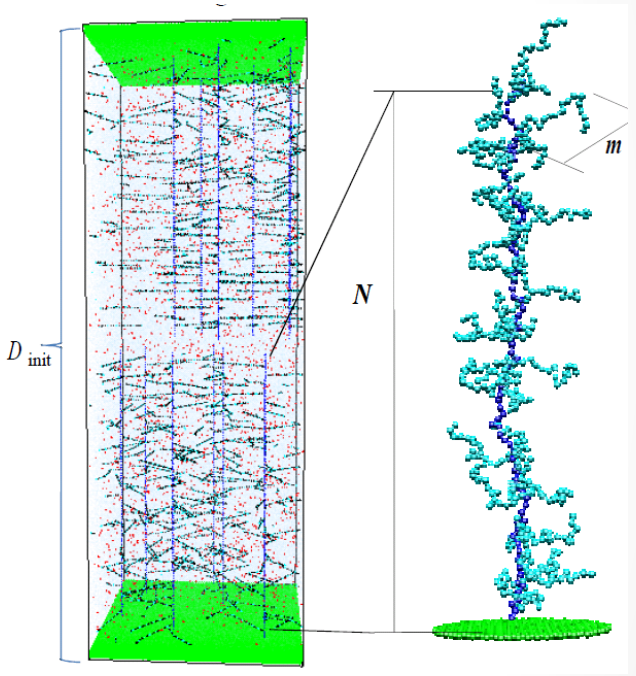




Bottle-Brush Polyelectrolytes

Courtesy of Jan-Michael Carrillo and Andrey Dobrynin
University of Connecticut

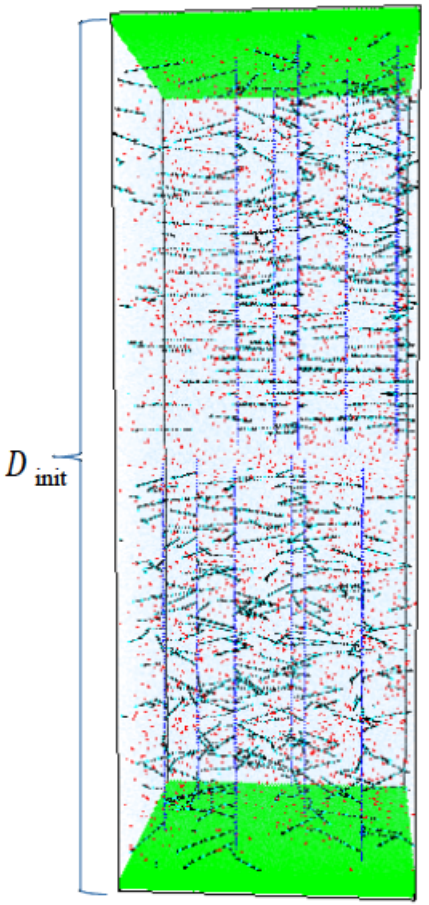
- Pure repulsive interaction potential was used for interaction among counterions, solvent and wall beads, with $\varepsilon_{LJ} = 1.0\ k_B T$ and $r_{cut} = 2^{1/6}\ \sigma$.
- Polymer - counterion, solvent and wall pairwise potential is pure repulsive and similar to above bullet
- Polymer-polymer pairwise interaction has $\varepsilon_{LJ} = 0.3\ k_B T$ and $r_{cut} = 2.5\ \sigma$
- Polymer bonds are fene bonds with $K = 30.0\ k_B T/\sigma^2$, $R_0 = 1.5\ \sigma$, $\varepsilon = 1.0\ k_B T$, and $\sigma = 1.0\ \sigma$



Russano, D., Carrillo, J.Y., Dobrynin, A.V. Interaction between Brush Layers of Bottle-Brush Polyelectrolytes, Langmuir, *In Press.*

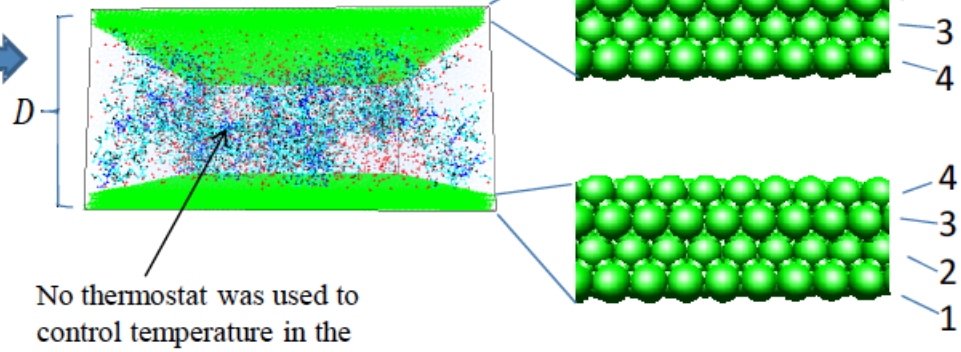
Bottle-Brush Polyelectrolytes

Initial Configuration



Pure repulsive interaction potential was used for interaction among particles belonging to layers 1-4, with $\varepsilon_{LJ} = 1.0\ k_B T$ and $r_{cut} = 2^{1/6}\ \sigma$. Layer 1-3 did not interact with liquid layer

Compress such that $\rho_b = 0.8\ \sigma^{-3}$

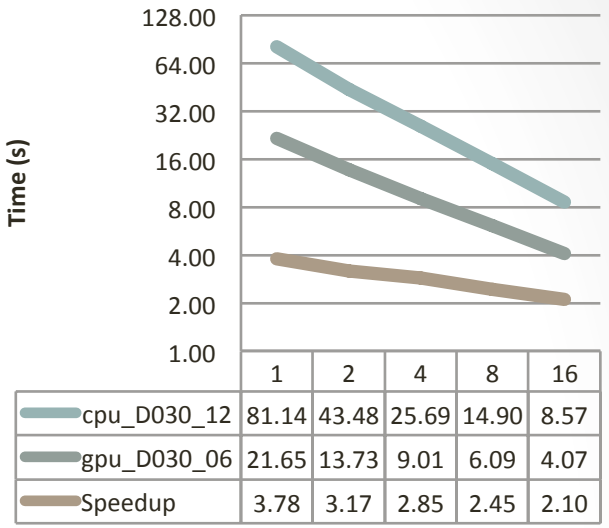


No thermostat was used to control temperature in the liquid layer. The temperature control was achieved through momentum exchange with the substrate

- Layers 2 and 3 are thermostated using Langevin Thermostat with friction coeff = $1/7\ \tau_{LJ}$
- Positions of the beads belonging to layer 1 were fixed
- Nearest neighbors were bonded by harmonic bonds with potential energy $E=K_b(r-r_o)^2$ where r_o is equal to σ and $K_b = 1000\ k_B T/\ \sigma^2$

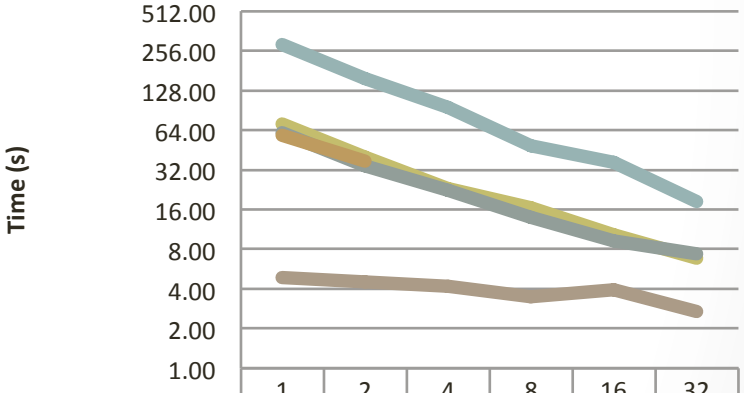
Bottle-Brush Polyelectrolytes

- GPU Benchmarks
  - 500 MD steps were performed with an integration timestep of 0.01 τ_{LJ}
  - Long range electrostatics via PPPM (1e-4)
  - Explicit solvent
  - Include all of the “fixes”, statistics calculations, and I/O from production runs
    - (1 configuration dump and 1 checkpoint dump)



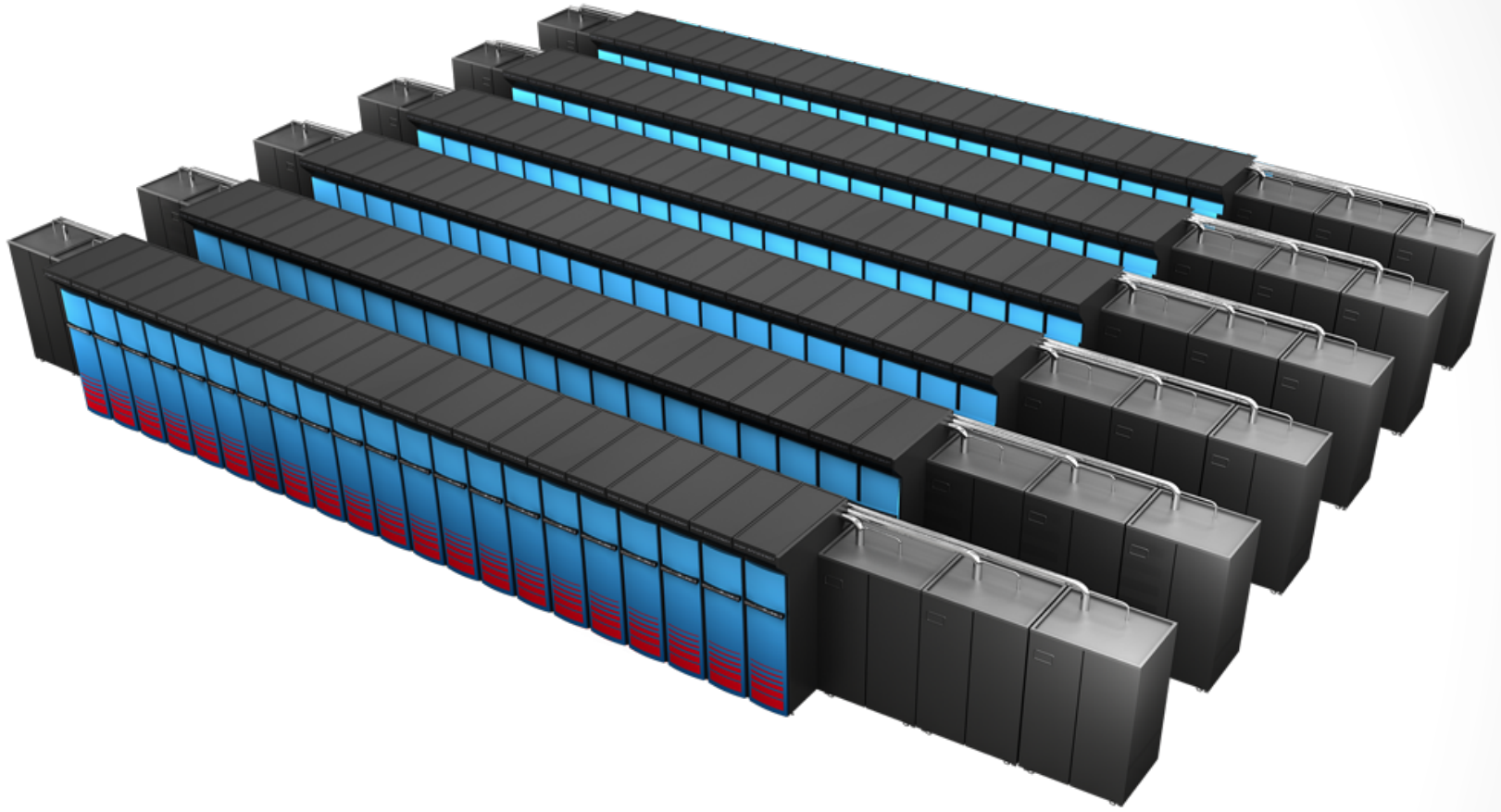
	1	2	4	8	16
cpu_D030_12	81.14	43.48	25.69	14.90	8.57
gpu_D030_06	21.65	13.73	9.01	6.09	4.07
Speedup	3.78	3.17	2.85	2.45	2.10

161194 Particles



	1	2	4	8	16	32
cpu_D140_12	286.44	157.09	94.77	48.80	36.29	18.39
gpu_D140_03	71.33	40.08	22.97	16.59	10.33	6.83
gpu_D140_06	60.85	34.55	22.50	13.92	9.23	7.38
gpu_D140_12	58.54	37.25				
Speedup	4.89	4.55	4.21	3.50	3.93	2.69

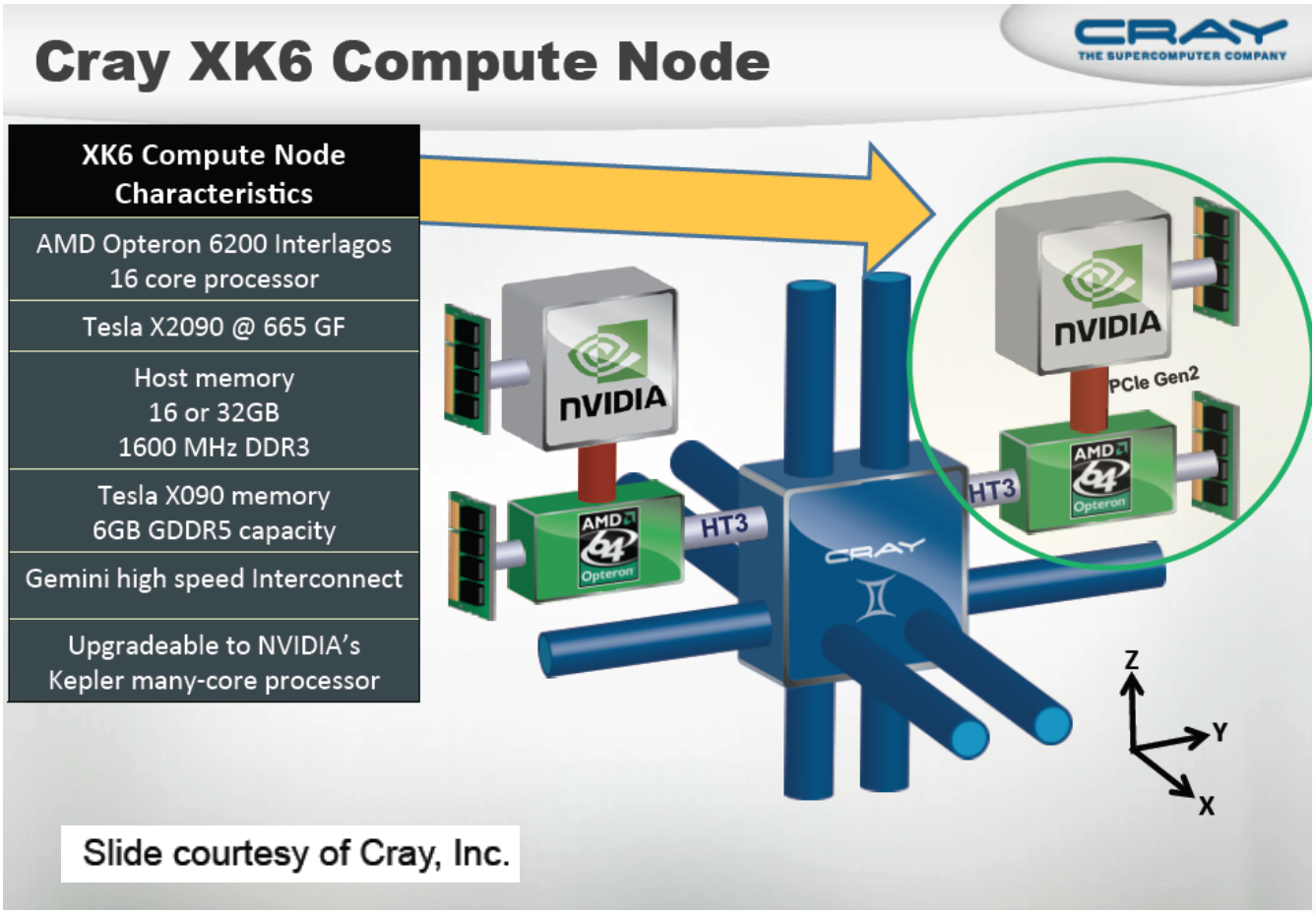
587971 Particles



GPU ACCELERATION AT ORNL

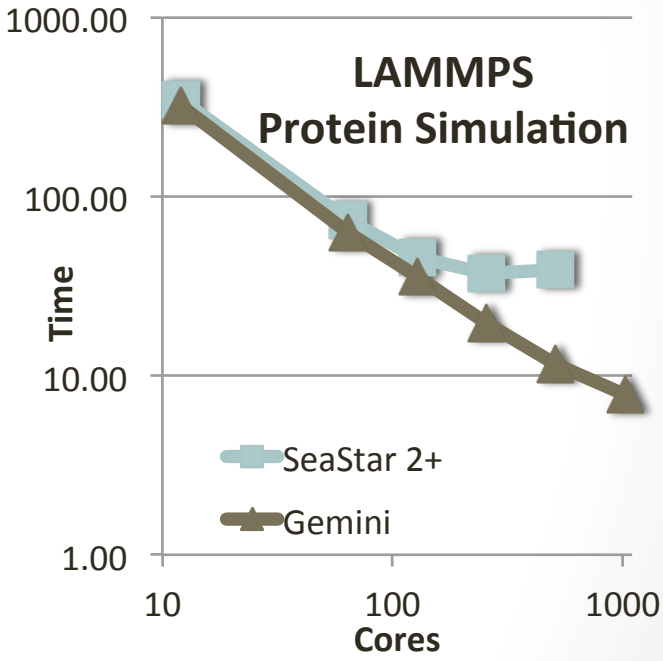
Titan

- Titan, the successor to the Jaguar supercomputer, will use Cray's accelerated XK6 node design with GPUs



Titan System Goals

- Designed for science from the ground up
- Operating system upgrade of today's Linux Operating System
- Gemini interconnect
  - 3-D Torus
  - Globally addressable memory
  - Advanced synchronization features
- 10-20 PF peak performance
- 9x performance of today's XT5
- Larger memory
- 3x larger and 4x faster file system



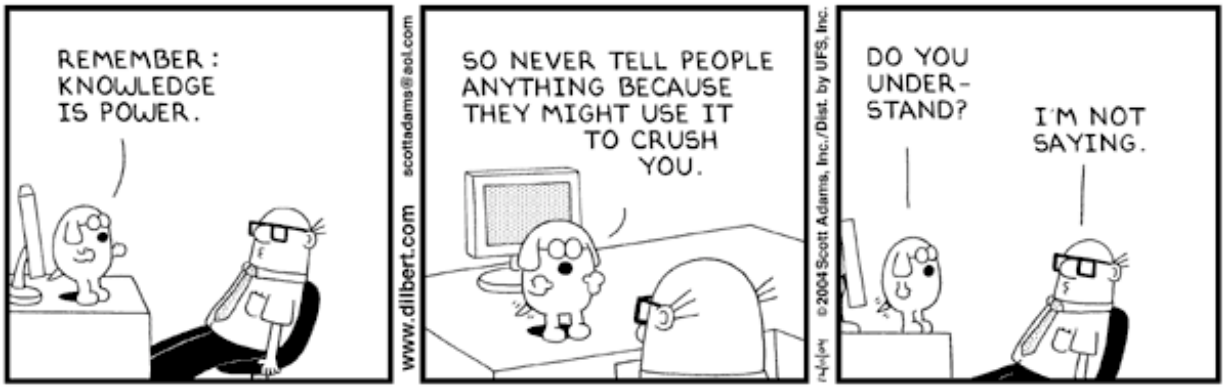
Courtesy of Sarah Anderson, Cray Inc.

Getting Access to Jaguar/Titan

- INCITE - Novel Computational Impact on Theory and Experiment Program
  - Typically awards millions of processor-hours per project
  - Addresses grand challenges in science and engineering
  - There is an annual call for INCITE proposals and awards are made on an annual basis
  - http://www.er.doe.gov/ascr/INCITE/index.html
- ALCC – The ASCR Leadership Computing Challenge
  - Emphasis on high-risk, high-payoff simulations
  - DOE energy mission areas
    - advancing the clean energy agenda
    - understanding the Earth's climate, …
  - open to national labs, academia, industry
  - http://www.er.doe.gov/ascr/Facilities/ALCC.html
- DD – Director's Discretion projects
  - dedicated to leadership computing preparation, INCITE and ALCC scaling, and application performance to maximize scientific application efficiency and productivity
  - http://www.nccs.gov/user-support/access/project-request/

Future Work

- Improve performance at smaller particle counts
  - Neighbor list build is the problem
- Improve long-range performance
  - MPI/Poisson Solve is the problem
- Further performance improvements focused on specific science problems
  - Collaborations welcome
- Questions?



Details on Methods Used

- 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.** *Submitted.*