Exceptional service in the national interest

Sandia National Laboratories

LAMMPS Tutorial

Stan Moore

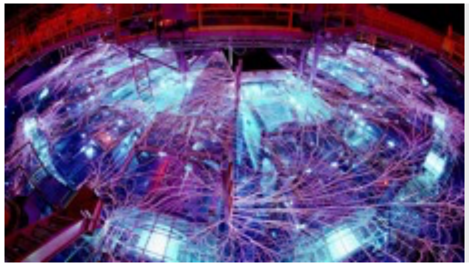
ISC 2021 Student Cluster Competition

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

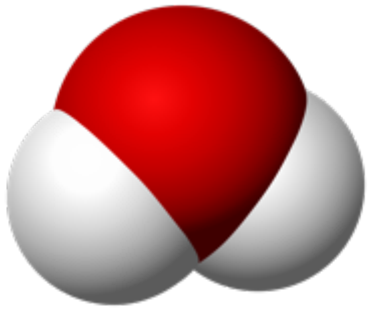
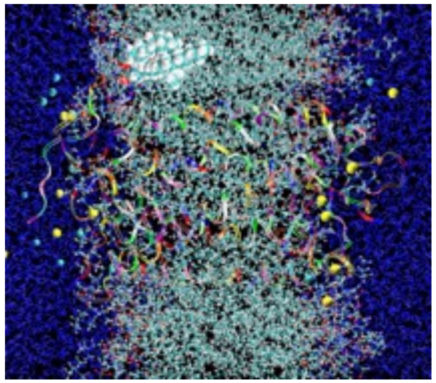
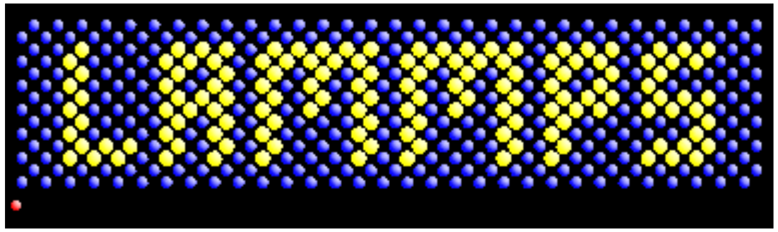
About Me

- Stan Moore
  - One of the LAMMPS code developers at Sandia National Laboratories in Albuquerque, New Mexico
  - Been at Sandia for ~9 years
  - Main developer of the KOKKOS package in LAMMPS (runs on GPUs and multi-core CPUs)
  - Expertise in long-range electrostatics
  - PhD in Chemical Engineering, dissertation on molecular dynamics method development for predicting chemical potential

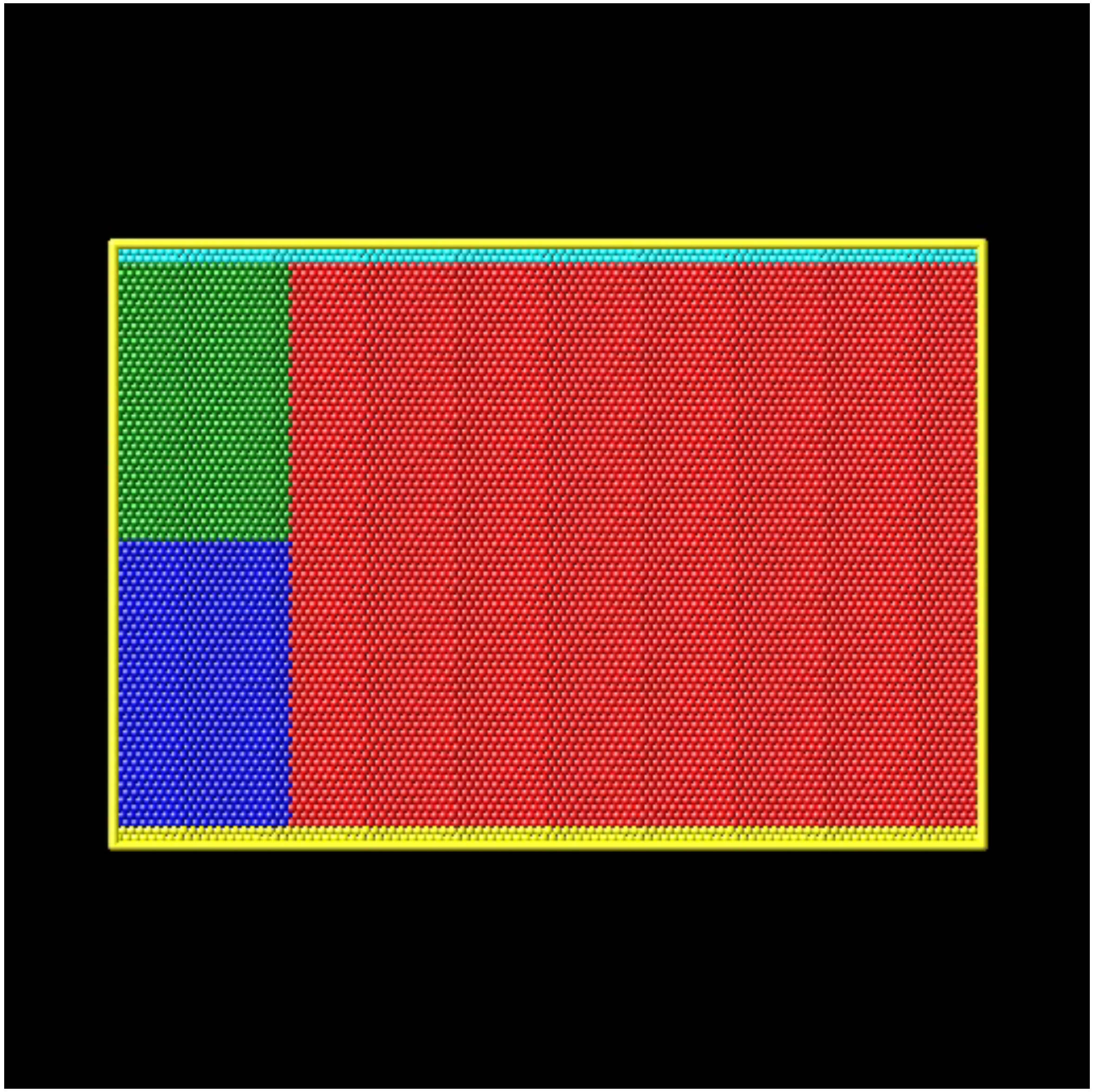


Molecular Dynamics (MD)

- Molecular Dynamics models atom behavior classically by using Newton's laws of motions
- Normally use an empirical expression for forces (does not include electrons)
- Atom positions → forces → velocities → new positions
- Spherical cutoff gives O(N) linear scaling, can simulate billions of atoms on a supercomputer

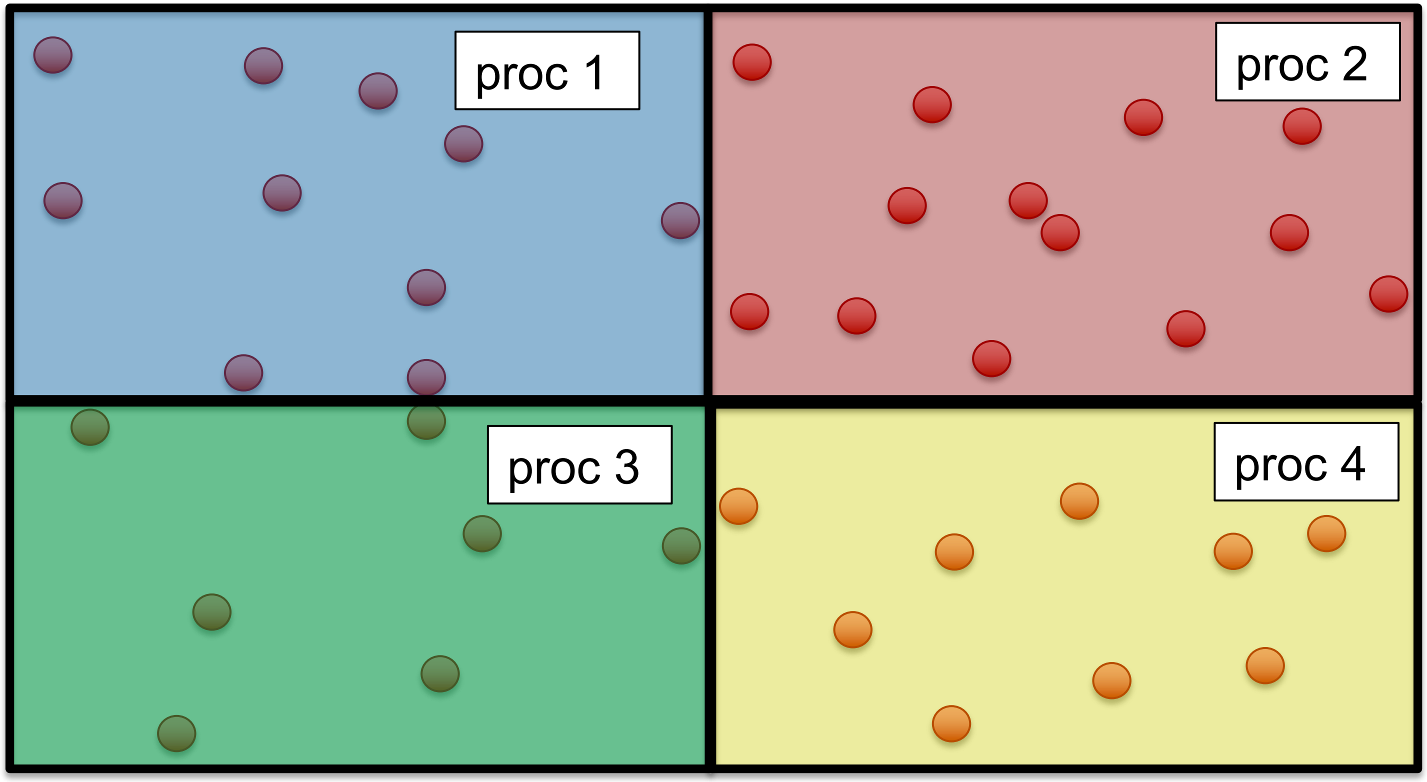




Simple Example: Crack



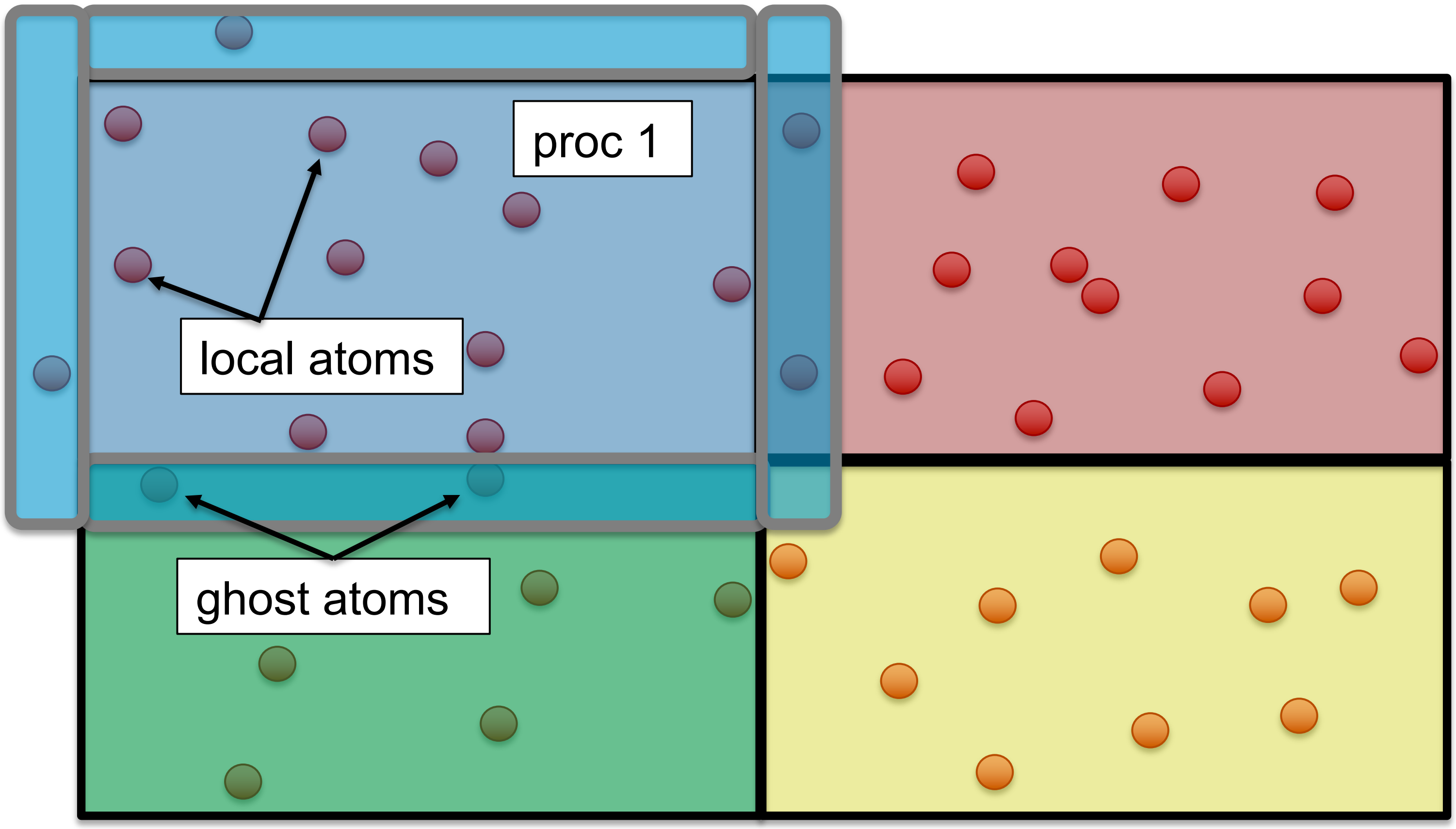
MPI Parallelization Approach

- Domain decomposition: each processor owns a portion of the simulation domain and atoms therein



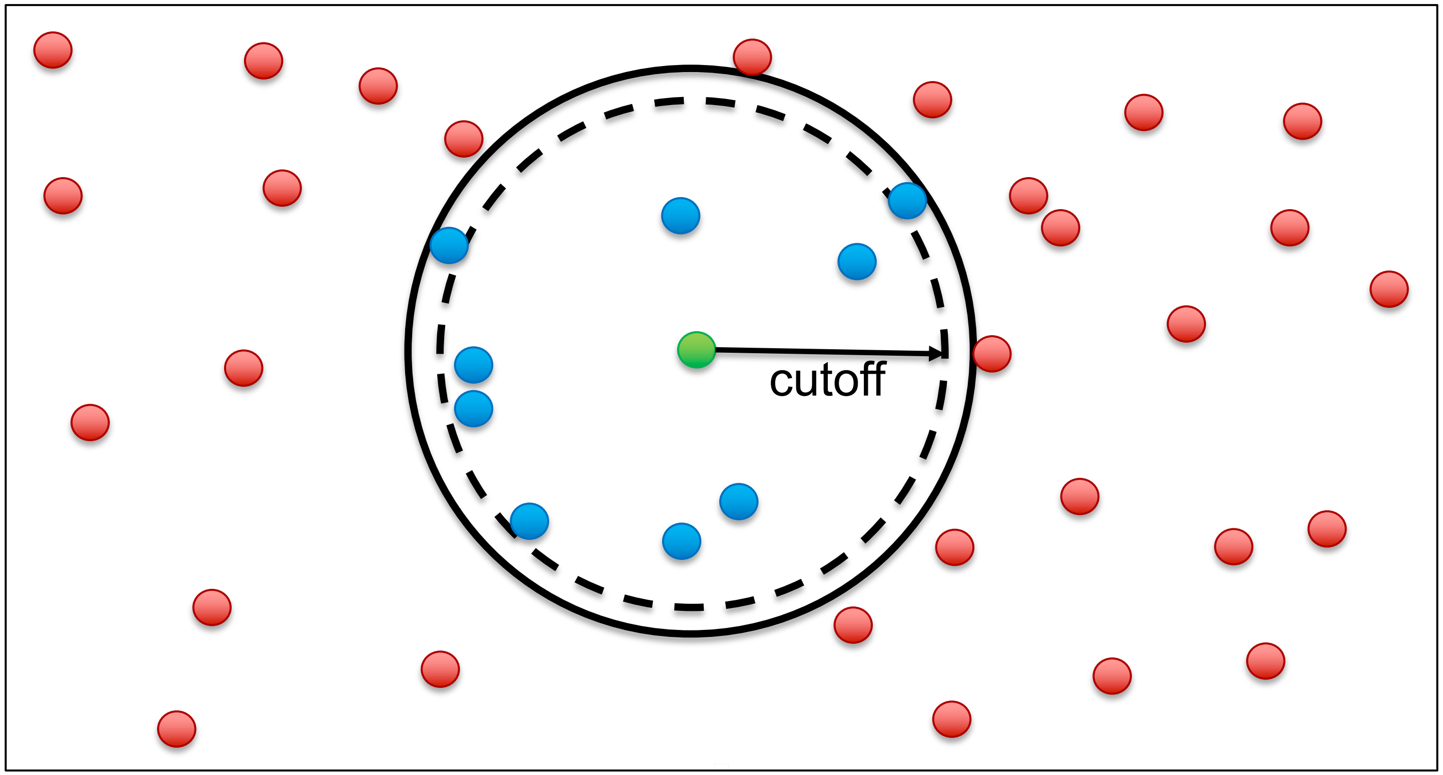
Ghost Atoms

- The processor domain is also extended to include needed ghost atoms (copies of atoms located on other processors)
- Communicated via MPI (message passing interface)



Neighbor Lists

- Neighbor lists are a list of neighboring atoms within the interaction cutoff + skin for each central atom
- Extra skin allows lists to be built less often



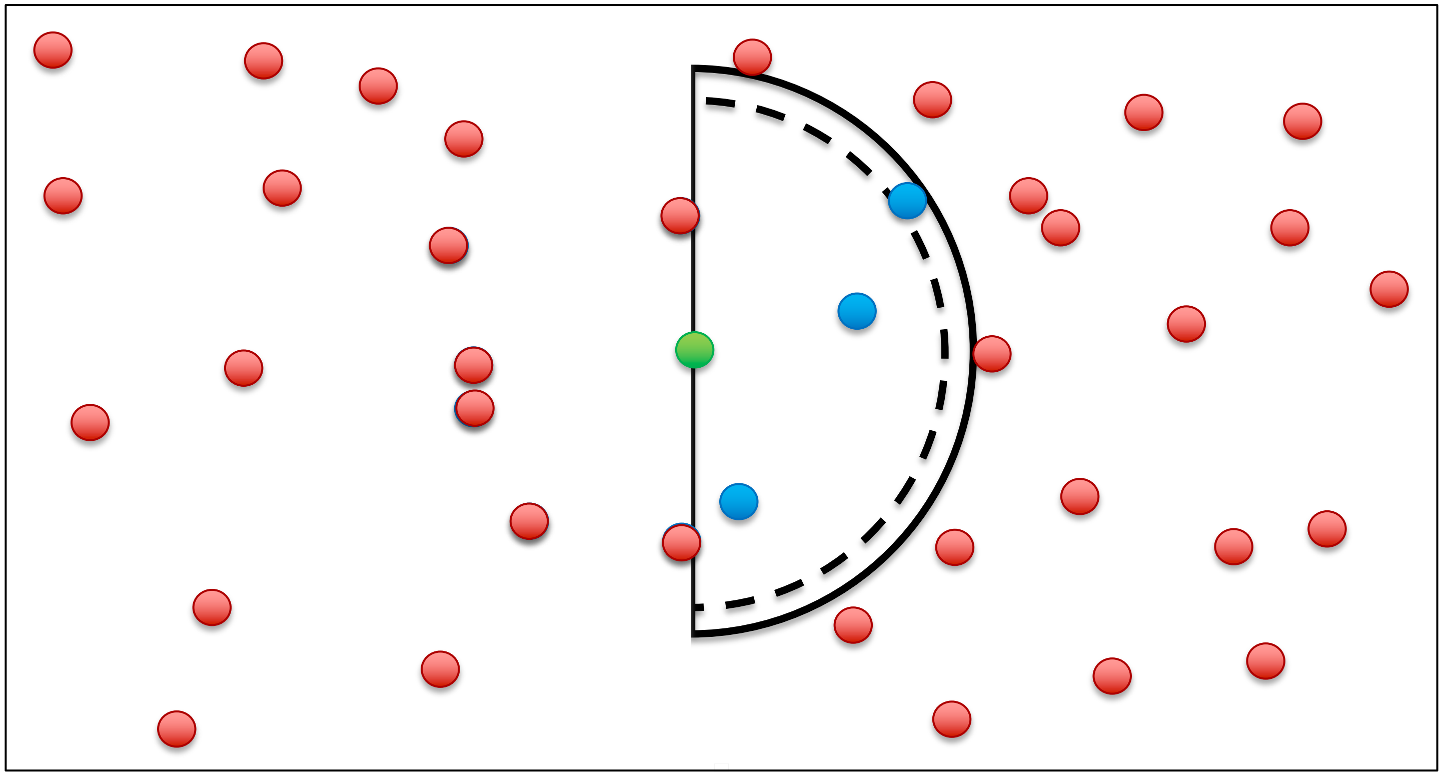
Newton Option

- Newton flag to *off* means that if two interacting atoms are on different processors, **both processors compute their interaction** and the resulting force information is not communicated
- Setting the newton flag to *on* saves computation but increases communication
- Performance depends on problem size, force cutoff lengths, a machine's compute/communication ratio, and how many processors are being used
- Newton off typically better for GPUs

```
newton on #default
newton off
```

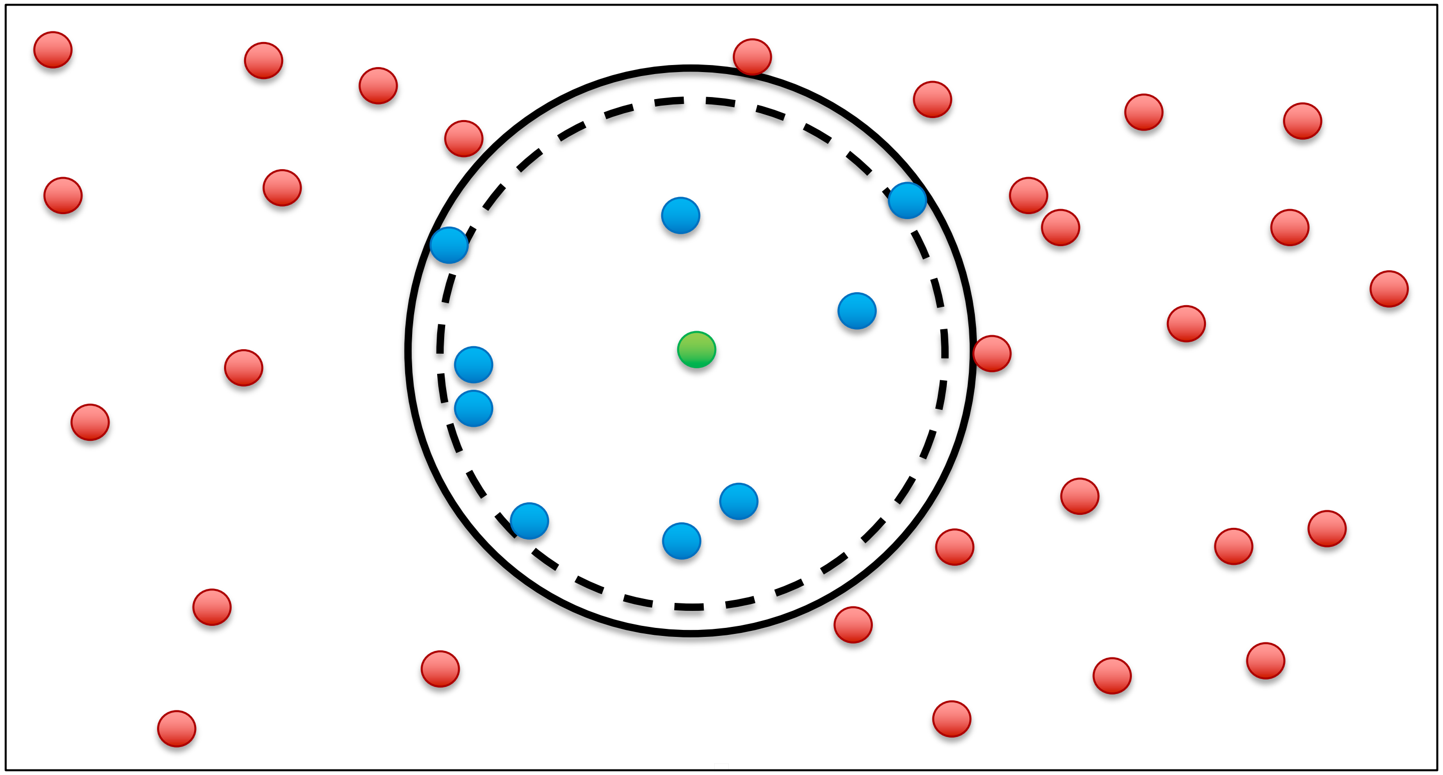
Half Neighbor List

- With newton flag on, each pair is stored only once (usually better for CPUs), requires atomic operations for thread-safety



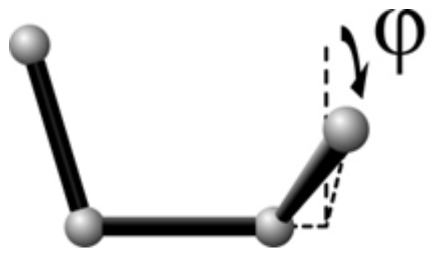
Full Neighbor List

- Each pair stored twice which doubles computation but reduces communication and doesn't require atomic operations for thread safety (can be faster on GPUs)



Molecular Topology

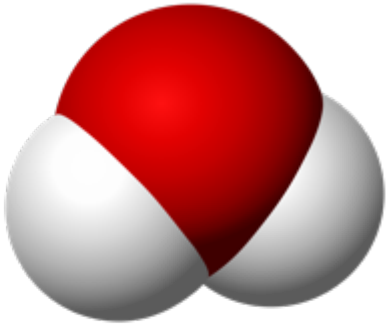
- Bonds: constrained length between two atoms
- Angles: constrained angle between three atoms
- Dihedrals: interactions between quadruplets of atoms
- Impropers: “improper” interactions between quadruplets of atoms



```
bond_style      harmonic
angle_style     charmm
dihedral_style  charmm
improper_style  harmonic
```

Fix Shake

- Applies bond and angle constraints to specified bonds and angles in the simulation
- Typically enables a longer timestep



```
fix    1 all shake 0.0001 5 0 m 1.0 a 232
```

Long-Range Electrostatics

- Truncation doesn't work well for charged systems due to long-ranged nature of Coulombic interactions
- Use Kspace style to add long-range electrostatics. PPPM method usually fastest, uses FFTs
- Specify a relative accuracy (i.e. 1e-4)
- Use `pair_style *coul/long` such as `lj/cut/coul/long` instead of `*coul/cut`
- Can vary Coulomb cutoff length and get the same answer

```
pair_style      lj/cut/coul/long 10.0
kspace_style    pppm 1e-4
```

Basic MD Timestep

- During each timestep (without neighborlist build):

1. Initial integrate
2. MPI communication
3. Compute forces (pair, bonds, kspace, etc.)
4. Additional MPI communication (if newton flag on)
5. Final integrate
6. Output (if requested on this timestep)

*Computation of diagnostics (fixes or computes) can be scattered throughout the timestep

LAMMPS Files

- Input file: text file with LAMMPS commands used to run a simulation
- Log file: text file with thermodynamic output from simulation
- Dump file: snapshot of atom properties, i.e. atom forces
- Restart file: binary checkpoint file with data needed to restart simulation
- Data file: text file that can be used to start or restart simulation

Downloading LAMMPS

- Github (https://github.com/lammps/lammps)
  - https://github.com/lammps/lammps/releases
  - Clone or download button, then download zip file
  - git clone … (beyond this tutorial)
- LAMMPS Website (http://lammps.sandia.gov)
  - Go to “download” link
  - Download gzipped tar file
- Stable version: more testing
- Development version: latest features and bug fixes

Compiling LAMMPS

- https://lammps.sandia.gov/doc/Build.html
- Need C++ compiler (GNU, Intel, Clang, nvcc)
- Need MPI library, or can use the “STUBS” library
- Many Makefiles in src/MAKE
- LAMMPS also has CMake interface

Running LAMMPS

- https://lammps.sandia.gov/doc/Run_basics.html
- Basic syntax: `[executable] -in [input_script]`
- In serial:

```
./lmp_serial -in in.lj
```

- In parallel:

```
mpirun -np 2 lmp_mpi -in in.lj
```

- Many other command line options, see https://lammps.sandia.gov/doc/Run_options.html

Optional Packages

- https://lammps.sandia.gov/doc/Packages_standard.html
- LAMMPS is very modular and has several optional packages
- Rhodopsin benchmark needs MOLECULE, KSPACE, RIGID packages installed

Traditional Make:

```
make yes-molecule
make no-molecule
```

CMAKE:

```
-D PKG_MOLECULE=yes
```

Accelerator Packages

- https://lammps.sandia.gov/doc/Speed_packages.html
- Some hardware components like GPUs, and multithreaded CPUs require special code (i.e. OpenMP, CUDA) to fully take advantage of the hardware
- LAMMPS has 5 accelerator packages:
  - USER-OMP
  - USER-INTEL
  - OPT
  - GPU
  - KOKKOS

OPT Package

- https://lammps.sandia.gov/doc/Speed_opt.html
- Methods rewritten in C++ templated form to reduce the overhead due to if tests and other conditional code
- Code also vectorizes better than the regular CPU version
- Contains 9 pair styles including Lennard-Jones
- No GPU support

Running OPT Package

- Compile LAMMPS with OPT package
- Run with 8 MPI: `mpiexec -np 8 ./lmp_exe -in in.lj -sf opt`
- `-sf opt` is the *suffix* command: automatically appends `/opt` onto anything it can
- For example, `pair_style lj/cut` automatically becomes `pair_style lj/cut/opt` (no changes to input file needed)
- https://lammps.sandia.gov/doc/suffix.html

USER-OMP Package

- https://lammps.sandia.gov/doc/Speed_omp.html
- Uses OpenMP to enable multithreading on CPUs
- MPI parallelization in LAMMPS is almost always more effective than OpenMP in USER-OMP on CPUs
- When running with MPI across multi-core nodes, MPI often suffers from communication bottlenecks and using MPI+OpenMP per node can be faster
- The more nodes per job and the more cores per node, the more pronounced the bottleneck and the larger the benefit from MPI+OpenMP

Running USER-OMP Package

- Compile LAMMPS with USER-OMP package
- Run with 2 MPI and 2 OpenMP threads:

```
export OMP_NUM_THREADS=2
mpiexec -np 2 ./lmp_exe -in in.lj -sf omp
```

USER-INTEL Package

- https://lammps.sandia.gov/doc/Speed_intel.html
- Allows code to vectorize and run well on Intel CPUs (with or without OpenMP threading)
- Can also be used in conjunction with the USER-OMP package
- Normally best performance out of all accelerator packages for CPUs

Running USER-INTEL Package

- Compile LAMMPS with USER-INTEL package
- To run using 2 MPI and 2 threads on a Intel CPU:

```
mpiexec -np 2 ./lmp_exe -in in.lj -pk intel
0 omp 2 mode double -sf intel
```

- `-pk` is the package command that sets package options, see https://lammps.sandia.gov/doc/package.html
- See also https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/1928986641/LAMMPS

GPU Package

- https://lammps.sandia.gov/doc/Speed_gpu.html
- Designed for one or more GPUs coupled to many CPU cores
- Only pair runs on GPU, fixes/bonds/computes run on CPU
- Atom-based data (e.g. coordinates, forces) move back and forth between the CPU(s) and GPU every timestep
- Asynchronous force computations can be performed simultaneously on the CPU(s) and GPU if using Kspace
- Provides NVIDIA and more general OpenCL support

Running GPU Package

- GPU library is found in lib/gpu
- Compile LAMMPS with GPU package
- Run with 16 MPI and 4 GPUs: `mpiexec -np 16 ./lmp_exe -in in.lj -sf gpu -pk gpu 4`
- Best to use CUDA MPS (Multi-Process Service) if using multiple MPI ranks per GPU
- Automatically overlaps pair-style on GPU with Kspace on CPU

Kokkos

- Abstraction layer between programmer and next-generation platforms
- Allows the same C++ code to run on multiple hardware (GPU, CPU)
- Kokkos consists of two main parts:
  1. Parallel dispatch—threaded kernels are launched and mapped onto backend languages such as CUDA or OpenMP
  2. Kokkos views—polymorphic memory layouts that can be optimized for a specific hardware

- Used on top of existing MPI parallelization (MPI + X)
- See https://github.com/kokkos/kokkos/wiki for more info

LAMMPS KOKKOS Package

- https://lammps.sandia.gov/doc/Speed_kokkos.html
- Supports OpenMP and GPUs
- Designed so that everything (pair, fixes, computes, etc.) runs on the GPU, minimal data transfer from GPU to CPU
- Package options can toggle full and half neighbor list, newton on/off, etc.

```
    -pk kokkos newton on neigh half
```

- https://lammps.sandia.gov/doc/package.html

Running Kokkos Package

- Compile LAMMPS with the KOKKOS package
- Run with 4 MPI and 4 GPUs: `mpiexec -np 4 ./lmp_exe -in in.lj -k on g 4 -sf kk`
- Run with 4 OpenMP threads: `./lmp_exe -in in.lj -k on t 4 -sf kk`

Overlapping with Kokkos

- https://lammps.sandia.gov/doc/Speed_kokkos.html
- Possible to overlap pair-style on GPU with Kspace, bonds, etc. on CPU
- Use `-pk kokkos pair/only on` to run only pair-style on GPU, everything else on CPU (like GPU package)
- Can manually specify `/kk/host` suffix to run on CPU, `/kk/device` suffix to run on GPU
- May need to compile with `--default-stream per-thread` flag to achieve overlap
- Can compile with both Cuda and OpenMP backends and run with OpenMP threading on CPU:

```
    -k on t 4 g 2 -sf kk
```

FFT Libraries

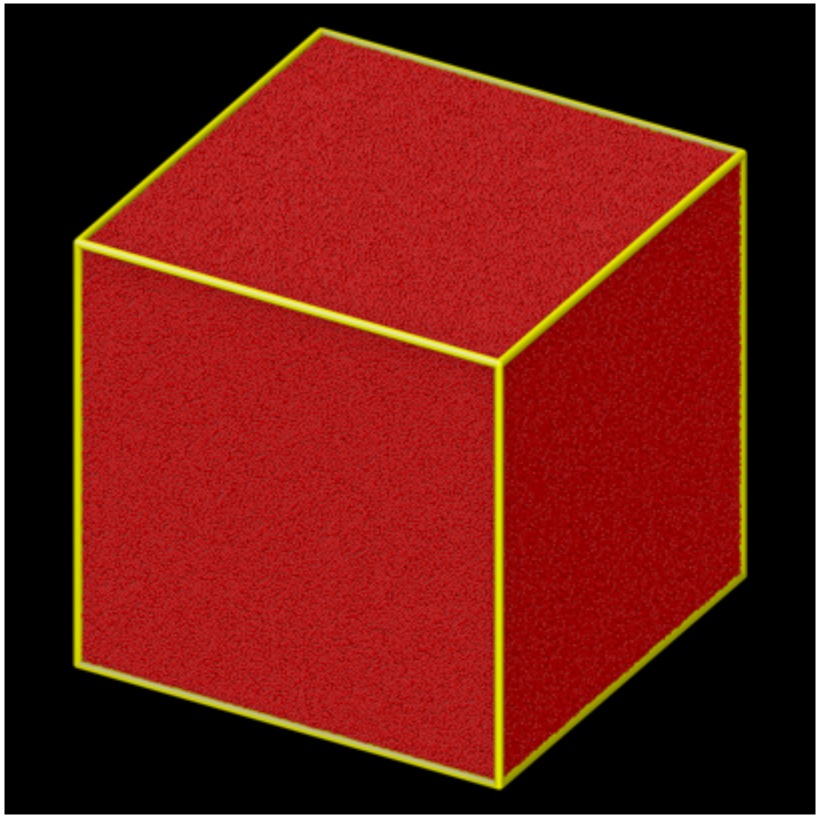
- https://lammps.sandia.gov/doc/Build_settings.html#fft
- LAMMPS needs FFT library for PPPM Kspace method
- The KISS FFT library is included with LAMMPS but other libraries can be faster
- KISS, FFTW, MKL, cuFFT options are supported

Processor and Thread Affinity

- Use mpirun command-line arguments (e.g. `--bind-to core`) to control how MPI tasks and threads are assigned to nodes and cores
- Also use OpenMP variables such as `OMP_PROC_BIND` and `OMP_PLACES`
- One must also pay attention to NUMA bindings between tasks, cores, and GPUs. For example, for a dual-socket system, MPI tasks driving GPUs should be on the same socket as the GPU

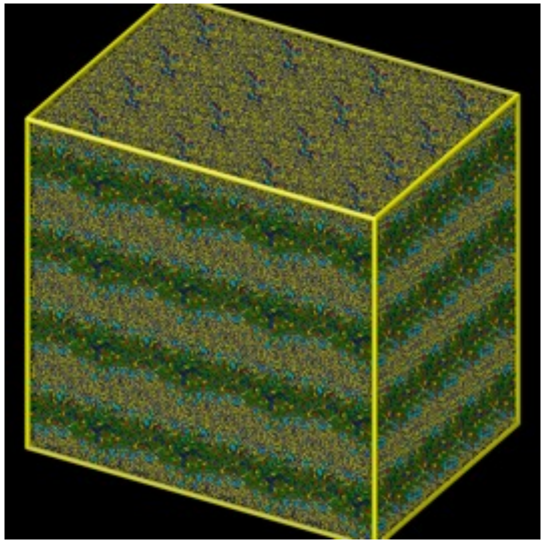
Lennard-Jones

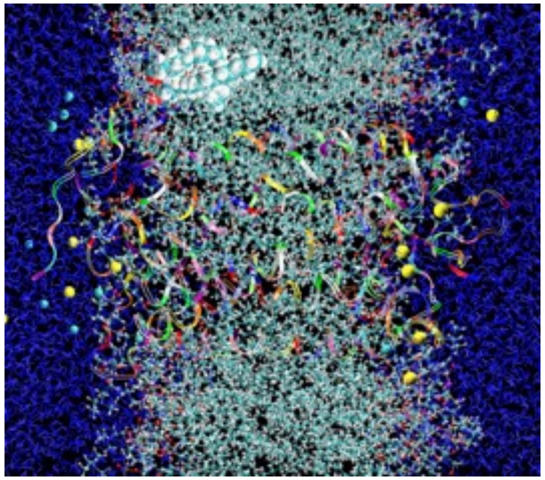
- Simple pair-wise model
- Similar to argon liquid/gas



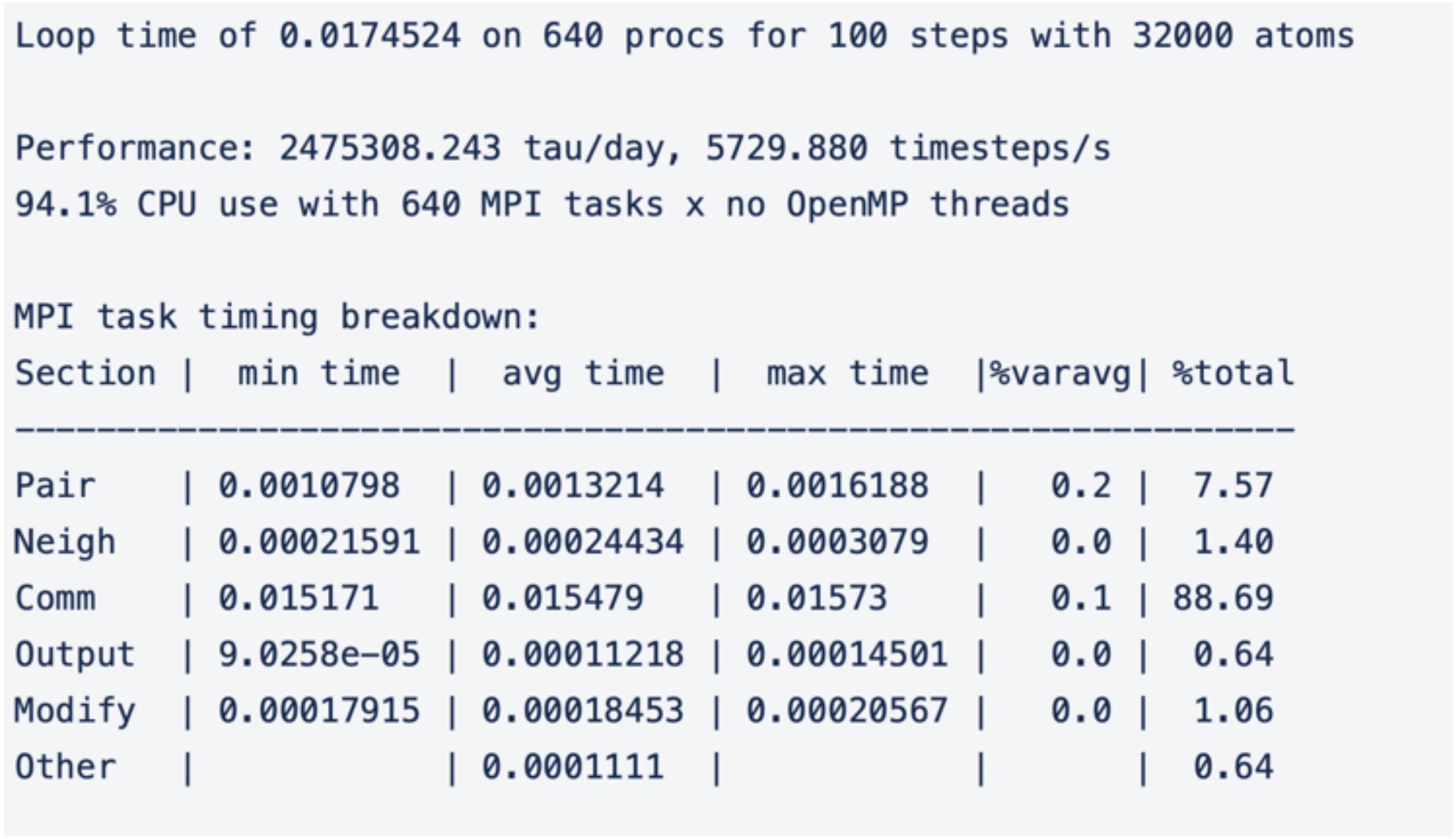
Rhodopsin

- Protein found in eyes (https://en.wikipedia.org/wiki/Rhodopsin)
- Model includes molecular topology (bonds, angles, etc.)
- Uses long-range electrostatics
- Requires KSPACE, MOLECULE, and RIGID packages





Measuring performance



```
Loop time of 0.0174524 on 640 procs for 100 steps with 32000 atoms

Performance: 2475308.243 tau/day, 5729.880 timesteps/s
94.1% CPU use with 640 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.0010798  | 0.0013214  | 0.0016188  |   0.2 |  7.57
Neigh   | 0.00021591 | 0.00024434 | 0.0003079  |   0.0 |  1.40
Comm    | 0.015171   | 0.015479   | 0.01573    |   0.1 | 88.69
Output  | 9.0258e-05 | 0.00011218 | 0.00014501 |   0.0 |  0.64
Modify  | 0.00017915 | 0.00018453 | 0.00020567 |   0.0 |  1.06
Other   |            | 0.0001111  |            |       |  0.64
```

- On GPUs, timing breakdown won’t be accurate without CUDA_LAUNCH_BLOCKING=1 (but will slow down simulation and prevent overlap)

Tuning Rules

- What is not allowed: **basically anything that changes the simulation results**
- What is allowed: any system or LAMMPS change that makes the simulation go faster without changing the results (half vs full neighbor list, newton on/off, etc.)
- For a full list see: https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/1928986641/LAMMPS

Visualization Resources

- LAMMPS “dump image” command: https://lammps.sandia.gov/doc/dump_image.html (uncomment line in input files)
- VMD: https://www.ks.uiuc.edu/Research/vmd/
- OVITO: https://www.ovito.org/about/ovito-pro/

Getting Help

- Look at LAMMPS documentation, latest version here: http://lammps.sandia.gov/doc/Manual.html)
- Search mail list archives here: https://sourceforge.net/p/lammps/mailman/lammps-users
- Subscribe to the LAMMPS mail list: http://lammps.sandia.gov/mail.html and then post questions
- Look at mail list posting guidelines first: http://lammps.sandia.gov/guidelines.html

Sandia National Laboratories

Questions?