LAMMPS – An Object Oriented Scientific Application

Dr. Axel Kohlmeyer
(with a little help from several friends)

Associate Dean for Scientific Computing
College of Science and Technology
Temple University, Philadelphia

http://sites.google.com/site/akohlmey/

a.kohlmeyer@temple.edu

LAMMPS is a Collaborative Project

A few lead developers and many significant contributors:

- Steve Plimpton, Aidan Thompson, Paul Crozier, Axel Kohlmeyer

  - Roy Pollock (LLNL), Ewald and PPPM solvers
  - Mike Brown (ORNL), GPU package
  - Greg Wagner (Sandia), MEAM package for MEAM potential
  - Mike Parks (Sandia), PERI package for Peridynamics
  - Rudra Mukherjee (JPL), POEMS package for rigid body motion
  - Reese Jones (Sandia), USER-ATC package for coupling to continuum
  - Ilya Valuev (JIHT), USER-AWPMD package for wave-packet MD
  - Christian Trott (Sandia), USER-CUDA package, KOKKOS package
  - A. Jaramillo-Botero (Caltech), USER-EFF electron force field package
  - Christoph Kloss (JKU), LIGGGHTS package for DEM and fluid coupling
  - Metin Aktulga (LBL), USER-REAXC package for C version of ReaxFF
  - Georg Gunzenmuller (EMI), USER-SPH package
  - Ray Shan (Sandia), COMB package, QEQ package
  - Trung Nguyen (ORNL), RIGID package, GPU package
  - Francis Mackay and Coling Denniston (U Western Ontario), USER-LB

LAMMPS is an Extensible Project

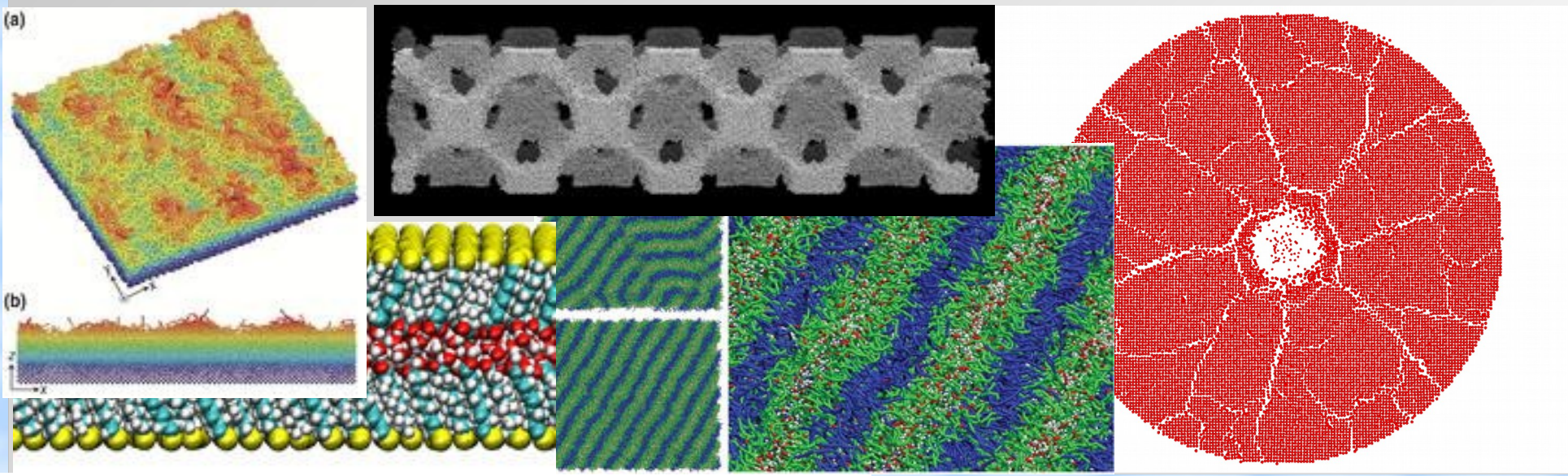
- ~2900 C/C++/CUDA files, 120 Fortran files, about 900,000 lines of code in core executable
- Only about 200 files are essential, about 600 files are compiled by default, 2300 are optional
- Optional files are included through derived C++ classes, extra functionality in bundled libraries
- Three levels of “package support”:
  - Core packages (officially supported)
  - USER-<NAME> packages (supported by individuals)
  - USER-MISC package (mixed bag of everything else)

A Short History of LAMMPS

- Started around 1995 as a DOE/Industry partnership under the lead of Steve Plimpton
- Development used Fortran 77 until 1999
- Converted to Fortran 90 for dynamical memory management. Final Fortran version in 2001. Switch to C++ to make adding modules easier
- Current version is a complete rewrite in C++ merging in features from several MD codes written at Sandia (ParaDyn, Warp, GranFlow, GRASP) and many community contributions

What LAMMPS Is

- Large-scale Atomic/Molecular Massively Parallel Simulator
  (each word is an attribute)
- Three-legged stool, supported by force fields and methods:
  one foot in biomolecules and polymers (soft materials)
  one foot in materials science (solids)
  one foot in mesoscale to continuum



LAMMPS General Features

- Classical Molecular Dynamics (MD) (+ Lattice Boltzman, Peridynamics, DEM Simulations, FE coupling extension)

  - open-source distribution, precompiled binaries for popular platforms
  - runs on a single processor or in parallel (with optional load balancing)
  - distributed-memory message-passing parallelism (MPI)
  - GPU (CUDA and OpenCL) and OpenMP support for many code features
  - spatial-decomposition of simulation domain for parallelism
  - optional libraries used: MPI, serial FFT, JPEG, PNG, Voro++, OpenKIM
  - integrated parallel visualizer (snapshot images and movies)
  - easy to extend with new features and functionality
  - syntax for defining and using variables and formulas
  - syntax for looping over runs and breaking out of loops
  - run one or multiple simulations simultaneously (in parallel) from one script
  - can be build as library, invoke LAMMPS through library interface
  - Python wrapper and module included, combine with Pizza.py toolkit
  - couple with other codes: LAMMPS calls other code, other code calls LAMMPS, or umbrella code calls both

Particle and Model Types

- simple atoms, metals
- coarse-grained particles (e.g. bead-spring polymers)
- united-atom polymers or organic molecules
- all-atom polymers, organic molecules, proteins, DNA
- granular materials
- coarse-grained mesoscale models
- finite-size spherical and ellipsoidal particles
- finite-size line segment (2d) and triangle (3d) particles
- point dipolar particles
- rigid collections of particles
- hybrid combinations of these

Force Fields

- Simple pairwise additive potentials: Lennard-Jones, Buckingham, Morse, Born-Mayer-Huggins, Yukawa, Soft, Class 2 (COMPASS), Mie, hydrogen bond, tabulated, Coulombic, point-dipole
- Manybody potentials: EAM, Finnis/Sinclair EAM, modified EAM (MEAM), embedded ion method (EIM), EDIP, ADP, Stillinger-Weber, Tersoff, REBO, AIREBO, ReaxFF, COMB, BOP
- Electron force fields: eFF, AWPMD
- Coarse-grained: DPD, GayBerne, REsquared, colloidal, DLVO, SDK
- Mesoscopic potentials: Granular media, Peridynamics, SPH, LB
- Potentials for bond/angles/dihedrals: harmonic, FENE, Morse, nonlinear, Class 2, quartic (breakable), CHARMM, OPLS, cvff, umbrella
- implicit solvent potentials: hydrodynamic lubrication, Debye
- long-range Coulombics and dispersion: Ewald, Wolf, PPPM (similar to particle-mesh Ewald), Ewald/N for long-range Lennard-Jones
- **hybrid potentials: multiple pair, bond, angle, dihedral, improper potentials can be used in one simulation**
- **overlaid potentials: superposition of multiple pair potentials**

Ensembles, Boundary Conditions

- 2d or 3d systems
- orthogonal or non-orthogonal (triclinic symmetry) simulation domains
- constant NVE, NVT, NPT, NPH, Parinello/Rahman integrators
- thermostatting options for groups and geometric regions of atoms
- pressure control via Nose/Hoover or Berendsen barostatting in 1 to 3 dimensions, coupled and uncoupled
- simulation box deformation (tensile and shear)
- harmonic constraint forces, collective variables (MTD, ABF, SMD)
- rigid body constraints
- SHAKE bond and angle constraints
- bond breaking, formation, swapping
- walls of various kinds
- non-equilibrium molecular dynamics (NEMD)
- Properties and manipulations can be controlled by custom functions

Methods

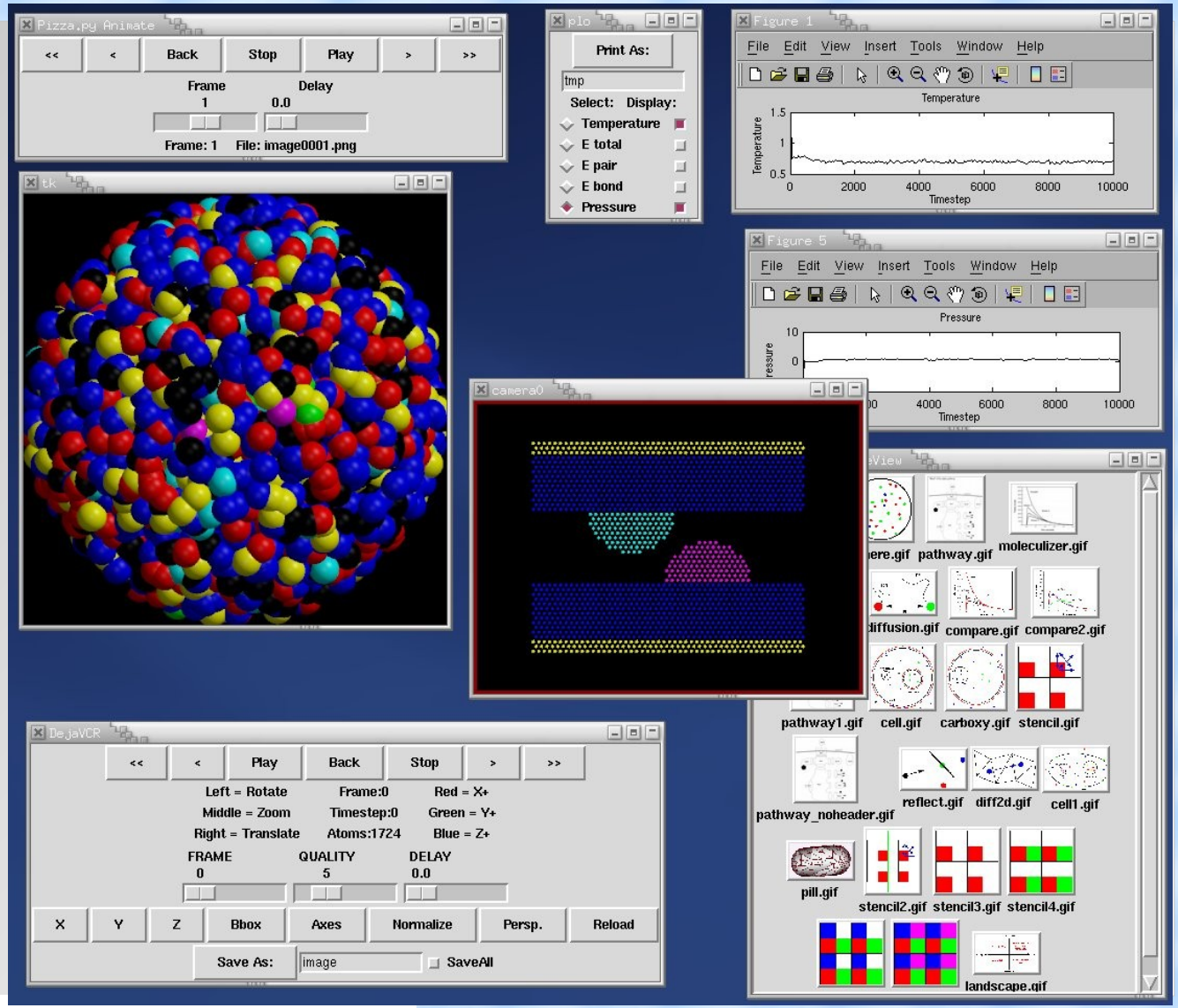
- Integrators:
  - Velocity Verlet, r-RESPA multi-timestepping, Brownian dynamics, rigid bodies
  - Energy minimization with various algorithms
- Multi-replica methods:
  - Nudged-elastic band
  - Parallel replica MD,Temperature accelerated MD
  - Parallel tempering MD
  - Split short-range / long-range force computation
  - Multi-walker metadynamics and ABF

Not so Common Features

- generalized aspherical particles
- stochastic rotation dynamics (SRD)
- real-time visualization and interactive MD
- atom-to-continuum coupling with finite elements
- grand canonical Monte Carlo insertions/deletions
- direct simulation Monte Carlo for low-density fluids
- Peridynamics mesoscale modeling
- targeted and steered molecular dynamics
- two-temperature electron model
- Dynamic grouping of particles
- On-the-fly parallel processing of data (direct and via rerun)

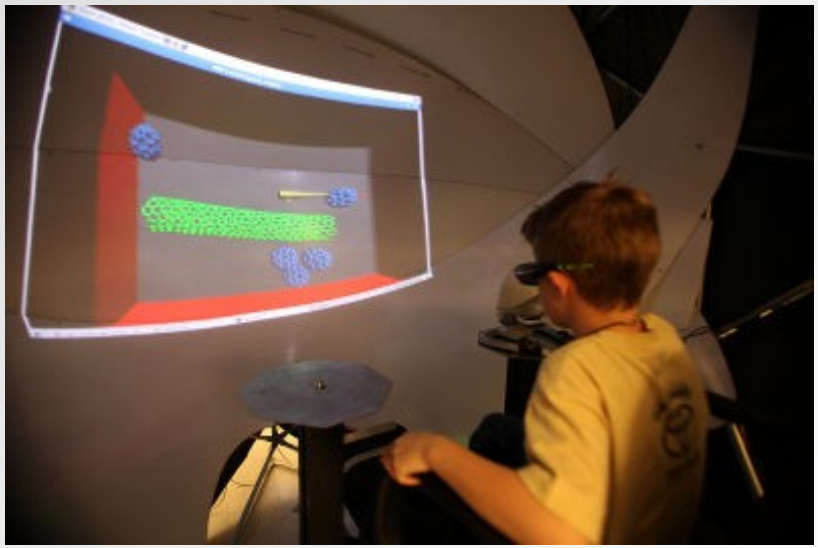
Pizza.py Companion Toolkit

- Each tool is a Python class
- Use multiple tools simultaneously from command-line, scripts, or GUIs
- Tools for building LAMMPS input, reading LAMMPS output, conversion, analysis, plotting, viz, etc
- GUI-based tool to run a LAMMPS simulation in real-time ...



LAMMPS for Outreach
The Nano Dome

- Single person immersive, stereo-3d, haptic, and interactive simulation/visualization environment
- Combines HPC, visualization, molecular simulation, virtual reality, and STEM outreach

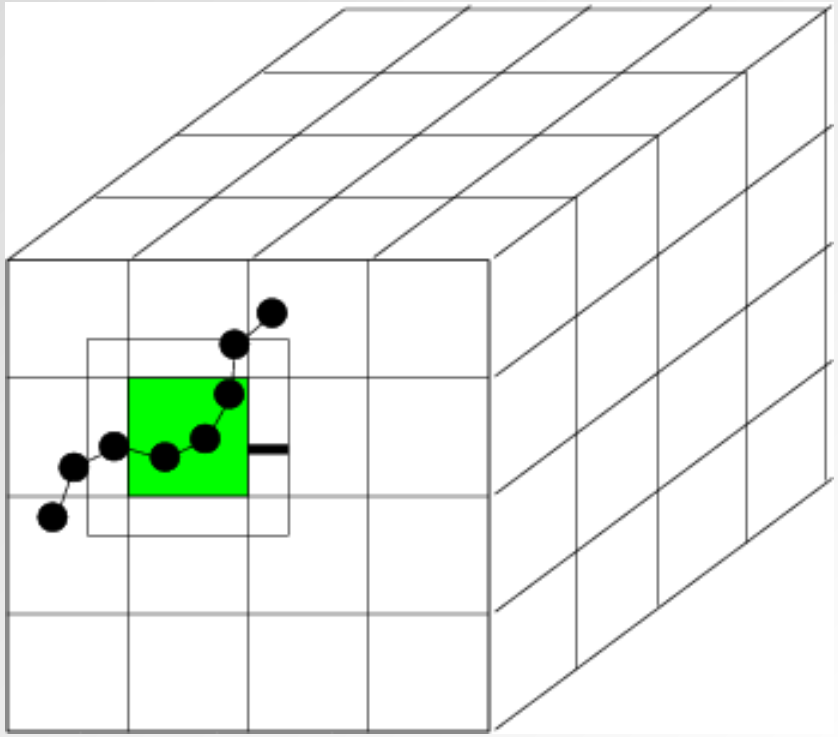


Serial Performance

- Low-level data structures are lists managed by classes
  C-like, Fortran-like
  x[N][3] = coordinates = 3N contiguous memory locations
  one simulation allocates many atom-based arrays
- Neighbor lists
  O(N) binning
  Verlet list with skin, stored in large “pages” of integers
  keep for 10-20 steps
  biggest memory requirement in code
- Performance is same as C and same as Fortran
  we don’t do things that slow down pair and neighbor routines
  people do care how fast your code is

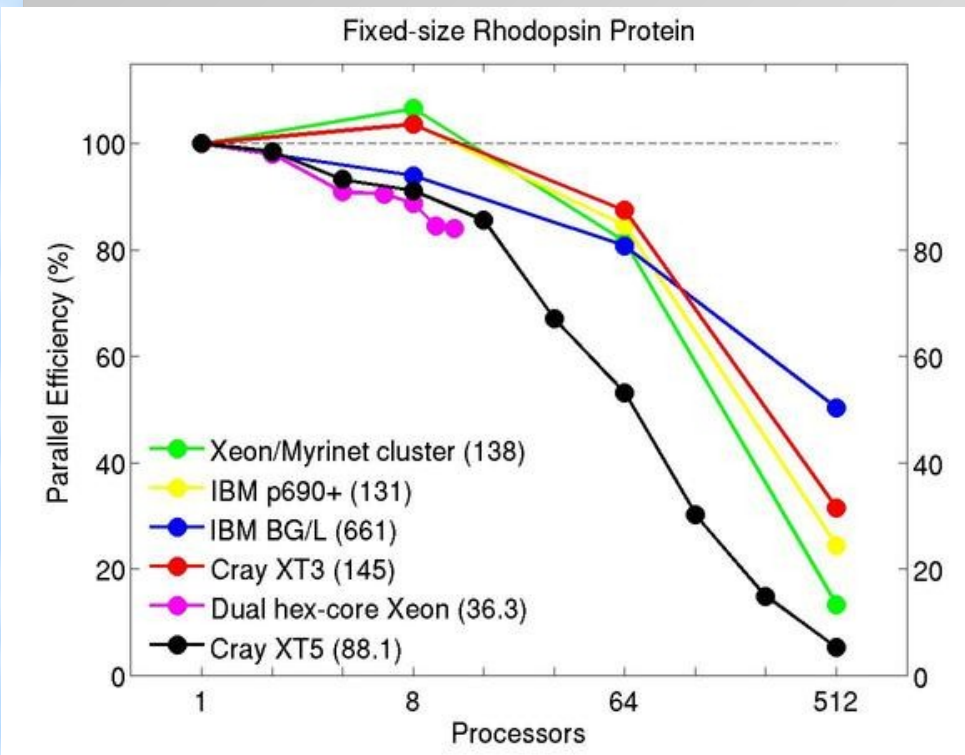
Parallelism via Spatial-Decomposition

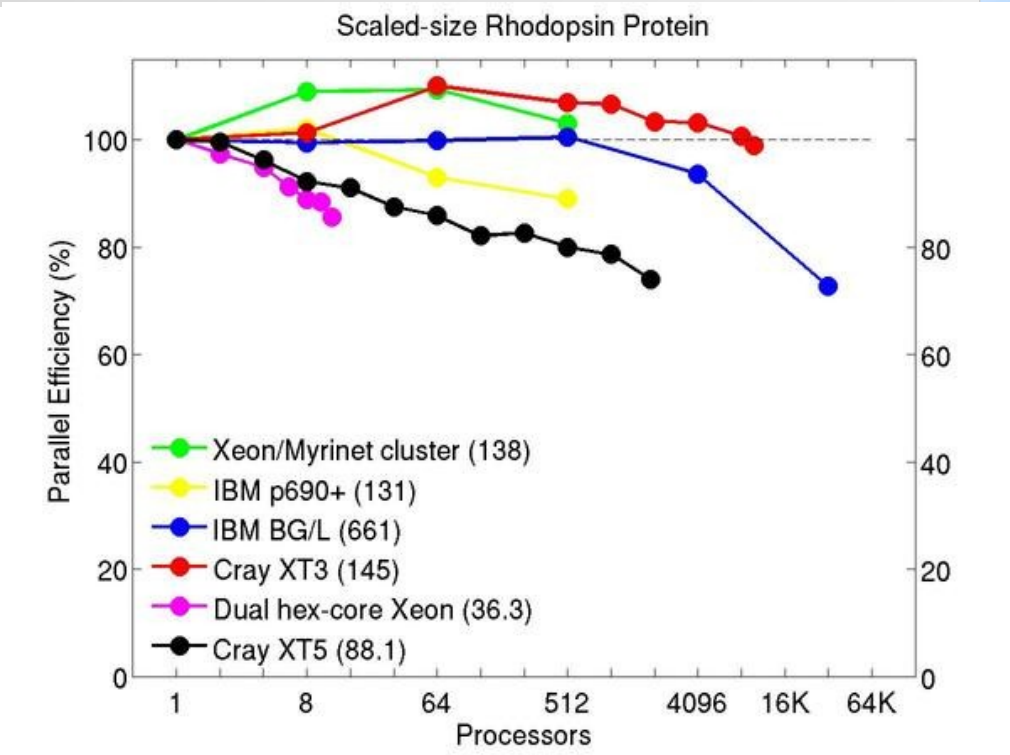
- Physical domain divided into 3d boxes, one per processor
- Communication of “ghost” atoms via
  nearest-neighbor 6-way stencil
- Each processor computes forces on atoms in its box
- Atoms "carry along" molecular topology
  as they migrate to new procs
- Work hard for optimal scaling:
  N/P so long as load-balanced
- Computation scales as N/P
- Communication scales
  sub-linear as $(N/P)^{2/3}$
  (for large problems)
- Memory scales as N/P
- Optional load balancing by atom count via moving domain dividing planes
- Optional recursive bisectioning decomposition



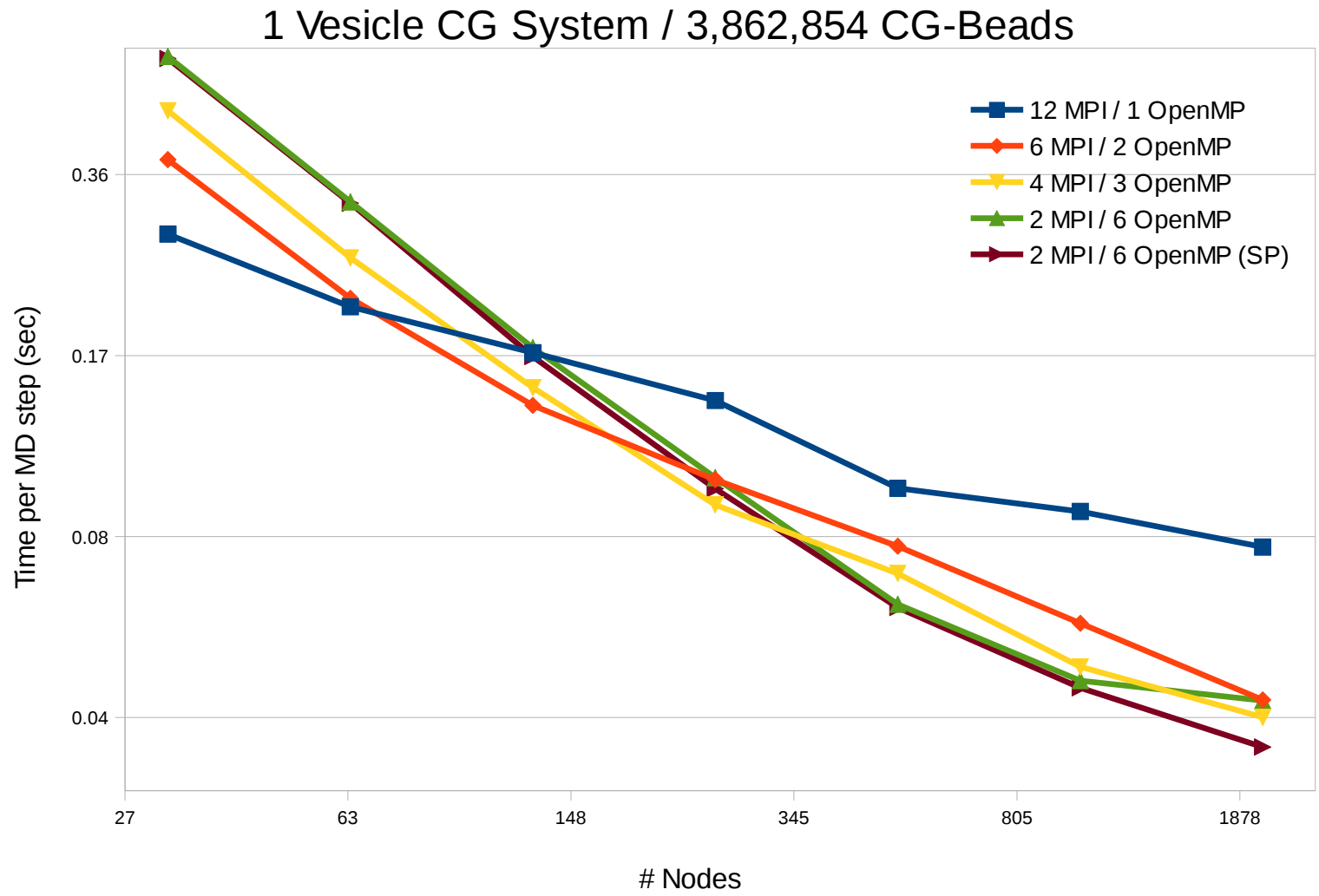
LAMMPS Performance

- Fixed-size (32K atoms) & scaled-size (32K/proc) parallel efficiencies
- Protein (rhodopsin) in solvated lipid bilayer

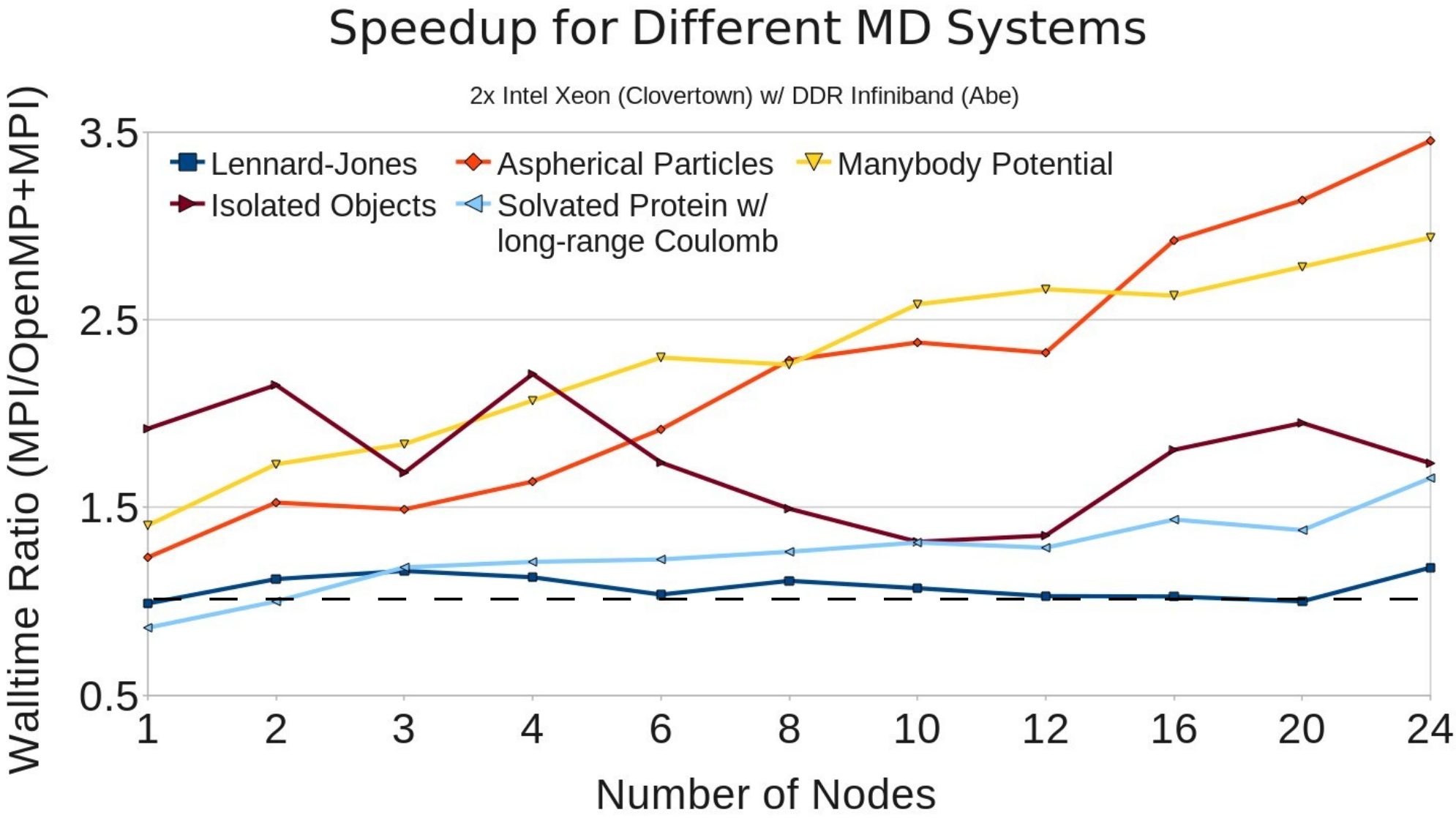




OpenMP/MPI Scaling on Cray XT5



OpenMP+MPI Best Effort vs. MPI



Extending LAMMPS

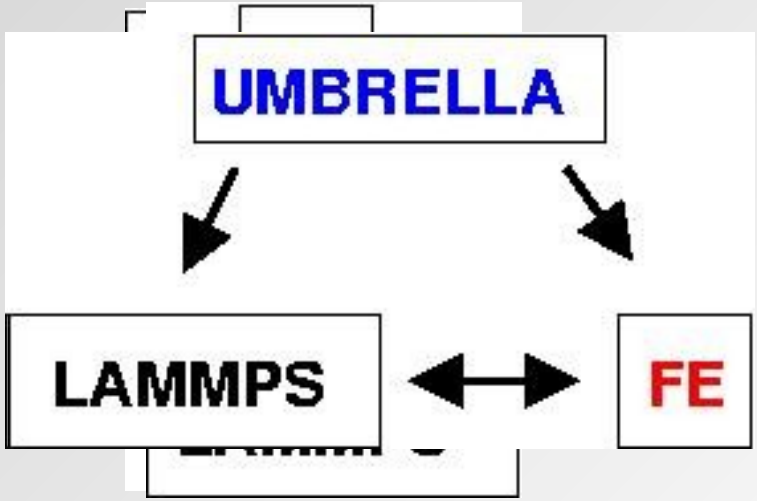
- In hindsight, this is best feature of LAMMPS
  > 80% of code is “extensions”
- Easy for us and others to add new features (“style”)
  new particle types
  new force fields
  new computations (T, per-atom stress, ...)
  new fix (BC, constraint, integrator, diagnostic, ...)
  new input command (read_data, velocity, run, ...)
- Adding a feature only requires 2 lines in a header file and recompiling

```
# include "pair_airebo.h"
PairStyle ( airebo, PairAIRebo )
```

- Enabled by C++
  virtual parent class for all styles, e.g. pair potentials
  defines interface the feature must provide
  compute(), init(), coeff(), restart(), etc

Coupling LAMMPS to Other Codes

- Method 1: MD is the driver
  - MD → FE
  - enabled by fixes, link to external library
  - coupled rigid body solver from RPI
- Method 2: Other code is the driver
  - FE → MD
  - build LAMMPS as a library
  - call from C++, C, Fortran
  - low-overhead to run MD in spurts
  - invoke low-level ops (get/put coords)



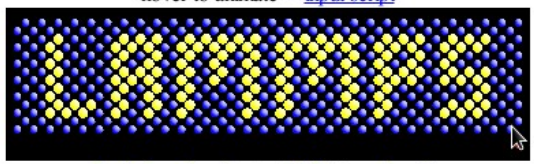
- Method 3: Umbrella code is the driver
  - Umbrella code calls MD and FE
  - RPI group linking LAMMPS to their FE codes for deformation problems
  - could run LAMMPS on P procs, FE on Q procs, talk to each other
- Challenge: balance the computation so both codes run efficiently

lammps.sandia.gov

LAMMPS Molecular Dynamics Simulator

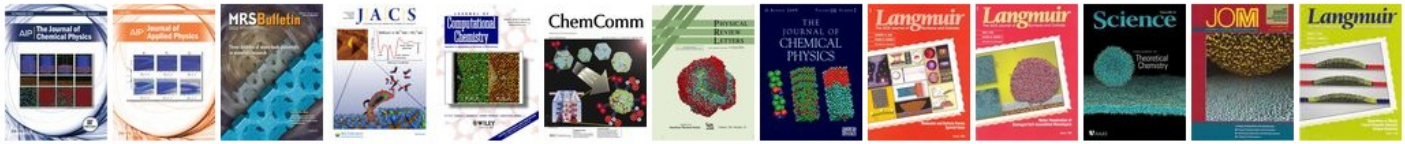
***lamp**: a device that generates light, heat, or therapeutic radiation; something that illumines the mind or soul* -- www.dictionary.com

hover to animate -- input script



physical analog (start at 3:25) & explanation

Big Picture	Code	Documentation	Results	Related Tools	Context	User Support
Features	Download	Manual	Publications	Pre/Post Processing	Authors	Mail list
Non-features	SourceForge	Developer Guide	Pictures	Pizza.py Toolkit	History	MD to LAMMPS glossary
FAQ	Latest Features & Bug Fixes	Tutorials	Movies	Offsite LAMMPS packages	Funding	User Scripts and HowTos
Wish list	Unfixed bugs	Commands	Benchmarks	Visualization	Open source	Workshops
.	.	.	Citing LAMMPS	Other MD codes	.	Contribute to LAMMPS



LAMMPS is a classical molecular dynamics code, and an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator.

LAMMPS has potentials for soft materials (biomolecules, polymers) and solid-state materials (metals, semiconductors) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale.

LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. The code is designed to be easy to modify or extend with new functionality.

LAMMPS is distributed as an open source code under the terms of the GPL. The current version can be downloaded here. Links are also included to older F90/F77 versions. Periodic releases are also available on SourceForge.

lammps.sandia.gov/#nogo ...andia National Laboratories, a US Department of Energy laboratory. The main authors of LAMMPS are listed on this page along with contact info and other contributors. Funding for