LAMMPS/miniMD overview:
communication/computation performance

Paul S. Crozier

August 24, 2010

miniMD vs. LAMMPS

Similarities:

- Underlying MD algorithms
  - Velocity verlet integration
  - Spatial decomposition
  - Essentially identical results
- "Look and feel"
  - Input script
  - Thermo output
- Performance
  - miniMD slightly faster
  - Scaling

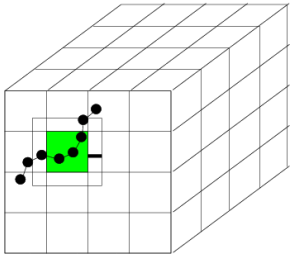
Differences:

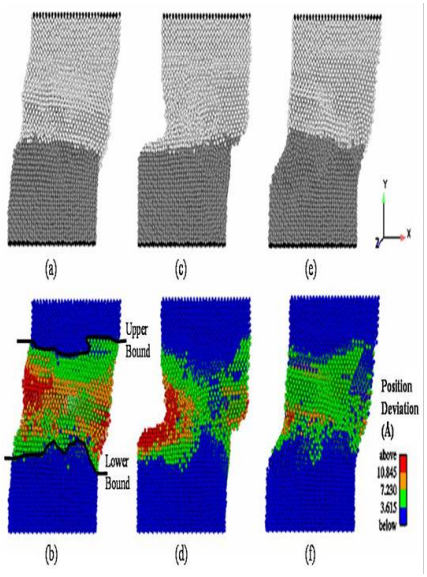
- 3 vs 130 klocs
- No optional packages
- Few commands/options
- Easier to build
- More portable
- Easier to overhaul / try new ideas
- No long-range electrostatics
- Only one pair style: LJ

LAMMPS

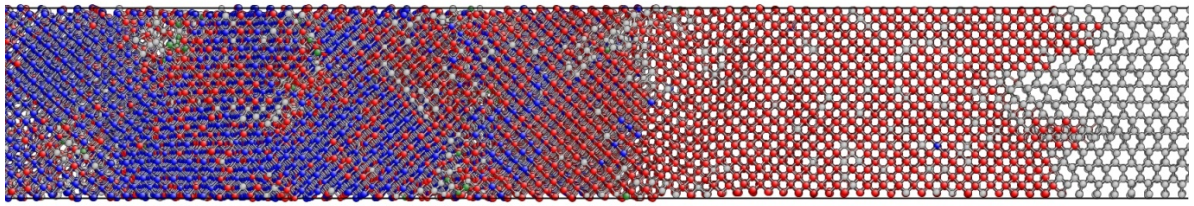
(Large-scale Atomic/Molecular Massively Parallel Simulator)

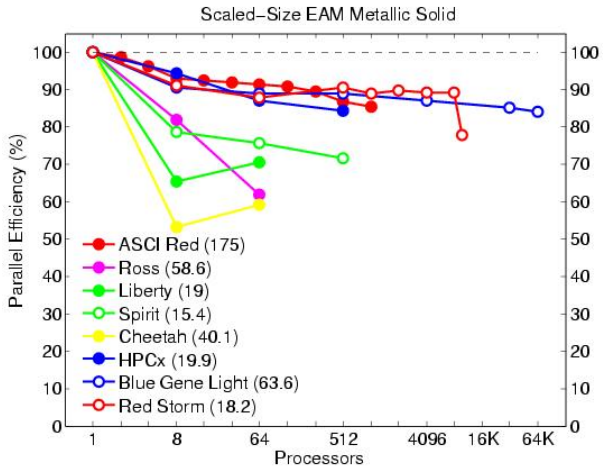
http://lammps.sandia.gov





- Classical MD code.
- Open source, highly portable C++.
- Freely available for download under GPL.
- Easy to download, install, and run.
- Well documented.
- Easy to modify or extend with new features and functionality.
- Active user's e-mail list with over 300 subscribers.
- Since Sept. 2004: over 20k downloads, grown from 53 to 125 kloc.
- Spatial-decomposition of simulation domain for parallelism.
- Energy minimization via conjugate-gradient relaxation.
- Radiation damage and two temperature model (TTM) simulations.
- Atomistic, mesoscale, and coarse-grain simulations.
- Variety of potentials (including many-body and coarse-grain).
- Variety of boundary conditions, constraints, etc.



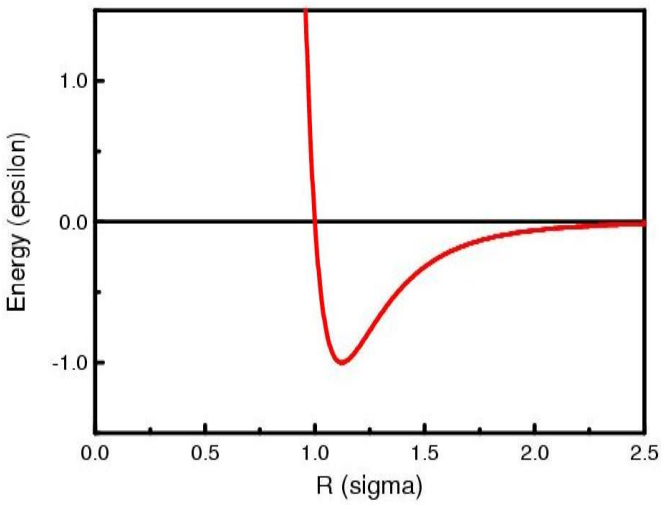


Force fields available in LAMMPS

- Biomolecules: CHARMM, AMBER, OPLS, COMPASS (class 2),
  long-range Coulombics via PPPM, point dipoles, ...
- Polymers: all-atom, united-atom, coarse-grain (bead-spring FENE), bond-breaking, …
- Materials: EAM and MEAM for metals, Buckingham, Morse, Yukawa,
  Stillinger-Weber, Tersoff, AI-REBO, Reaxx FF, ...
- Mesoscale: granular, DPD, Gay-Berne, colloidal, peri-dynamics, DSMC ...
- Hybrid: can use combinations of potentials for hybrid systems:
  water on metal, polymers/semiconductor interface,
  colloids in solution, …

Classical MD Basics

- Each of N particles is a point mass
  - atom
  - group of atoms (united atom)
  - macro- or meso- particle
- Particles interact via empirical force laws
  - all physics in energy potential → force
  - pair-wise forces (LJ, Coulombic)
  - many-body forces (EAM, Tersoff, REBO)
  - molecular forces (springs, torsions)
  - long-range forces (Ewald)
- Integrate Newton's equations of motion
  - F = ma
  - set of N, coupled ODEs
  - advance as far in time as possible
- Properties via time-averaging ensemble snapshots (vs MC sampling)



MD Timestep

- Velocity-Verlet formulation:
  - update V by ½ step (using F)
  - update X (using V)
  - build neighbor lists (occasionally)
  - compute F (using X)
  - apply constraints & boundary conditions (on F)
  - update V by ½ step (using new F)
  - output and diagnostics

- CPU time break-down:
  - forces = 80%
  - neighbor lists = 15%
  - everything else = 5%

Computational Issues

- These have a large impact on CPU cost of a simulation:
  - Level of detail in model
  - Cutoff in force field
  - Long-range Coulombics
  - Neighbor lists
  - Newton's 3rd law (compute on ghost atoms, but more communication)
  - Timestep size (vanilla, SHAKE, rRESPA)
  - Parallelism

Classical MD in Parallel

- MD is inherently parallel
  - forces on each atom can be computed simultaneously
  - X and V can be updated simultaneously

- Most MD codes are parallel
  - via distributed-memory message-passing paradigm (MPI)

- Computation scales as N = number of atoms
  - ideally would scale as N/P in parallel

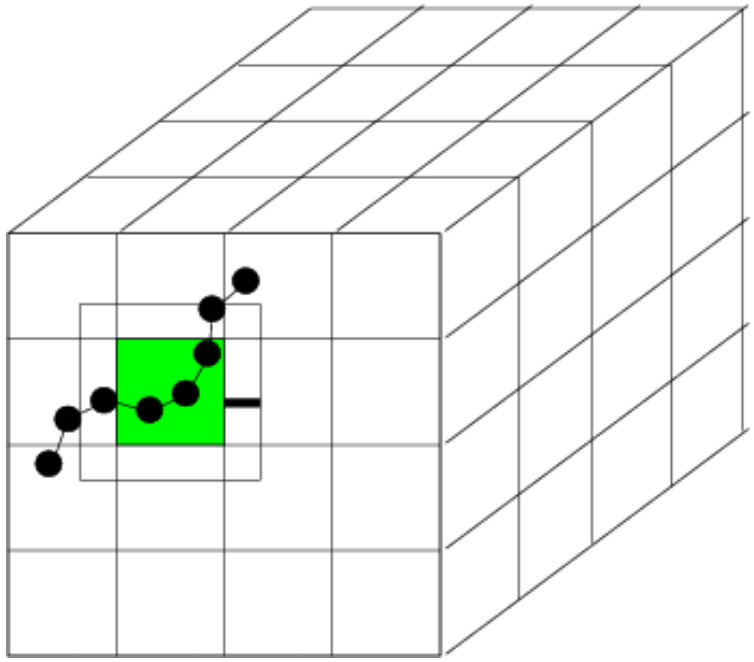
- Can distribute:
  - atoms communication = scales as N
  - forces communication = scales as N/sqrt(P)
  - space communication = scales as N/P or $(N/P)^{2/3}$

Parallelism via Spatial-Decomposition

- Physical domain divided into 3d boxes, one per processor
- Each proc computes forces on atoms in its box
  using info from nearby procs
- Atoms "carry along" molecular topology
  as they migrate to new procs
- Communication via
  nearest-neighbor 6-way stencil

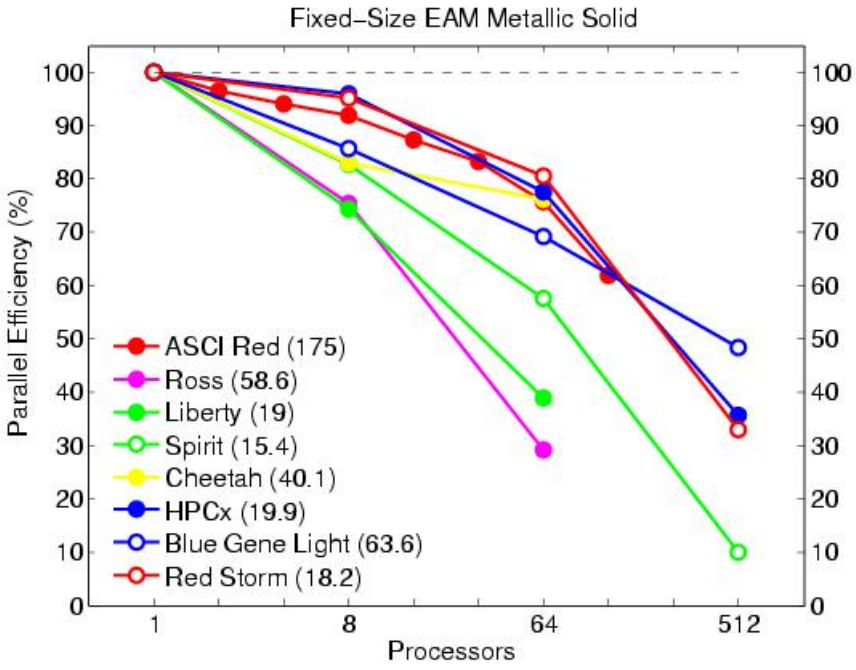
- Optimal scaling for MD: N/P
  so long as load-balanced

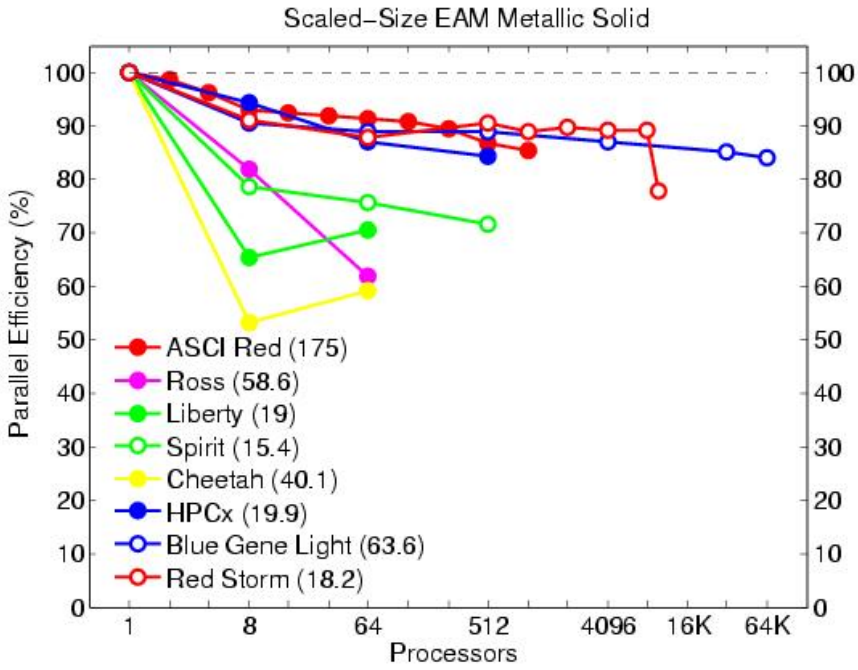
- Computation scales as N/P
- Communication scales
  sub-linear as $(N/P)^{2/3}$
  (for large problems)
- Memory scales as N/P



Parallel performance, EAM

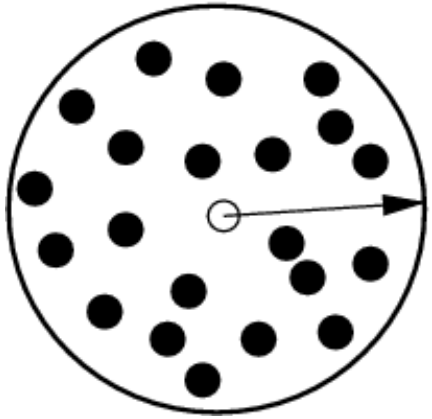
- Fixed-size (32K atoms) and scaled-size (32K atoms/proc) parallel efficiencies
- Metallic solid with EAM potential





Cutoff in Force Field

- Forces = 80% of CPU cost
- Short-range forces → O(N) scaling for classical MD
  - constant density assumption
  - pre-factor is cutoff-dependent
- # of pairs/atom = cubic in cutoff
  - 2x the cutoff → 8x the work
- Use as short a cutoff as can justify:
  - LJ = 2.5σ (standard)
  - all-atom and UA = 8-12 Angstroms
  - bead-spring = $2^{1/6}\,\sigma$ (repulsive only)
  - Coulombics = 12-20 Angstroms
  - solid-state (metals) = few neighbor shells (due to screening)
- Test sensitivity of your results to cutoff



Long-range Coulombics

- Systems that need it:
  - Charged polymers (polyelectrolytes)
  - Organic & biological molecules
  - Ionic solids
  - Not metals (screening)
- Computational issue:
  - Coulomb energy only falls off as 1/r
- Options:
  - cutoff scales as N, but large contribution at 10 Angs
  - Ewald scales as $N^{3/2}$
  - particle-mesh Ewald scales as $N\log_2 N$
  - multipole scales as N (but doesn't beat PME)

Ewald Summation

- Replace point charges with extended Gaussians:

$$\rho_i(r) = Z_i \left(\frac{G^2}{\pi}\right)^{3/2} \exp\left[-G^2 (r - r_i)^2\right]$$

- Interacting charges gives:

$$U = \sum_{i=1}^{N} \sum_{j>i}^{N} \frac{Z_i Z_j}{r_{ij}} \operatorname{erfc}\left(G \frac{r_{ij}}{\sqrt{2}}\right) + \frac{1}{2} \int\int \frac{\rho(r)\rho(r')}{|r - r'|} dr dr' - \frac{G}{\sqrt{2\pi}} \sum_{i=1}^{N} Z_i^2$$

- Short-range and long-range portion
- Ewald method replaces integral with sum over K-points
- Parallel:
  - requires sum of K-vector across all processors
  - MPI_Allreduce operation (scalability issue)

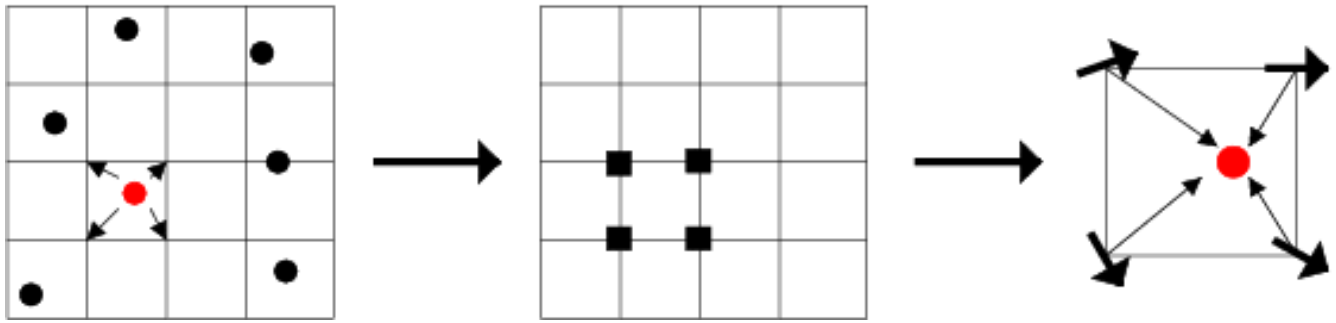
- User-specified accuracy + cutoff → G + # of K-points
- Scales as $N^{3/2}$ if grow cutoff as $N^{1/6}$

Particle-mesh Methods for Coulombics

- Coulomb interactions fall off as 1/r so require long-range for accuracy

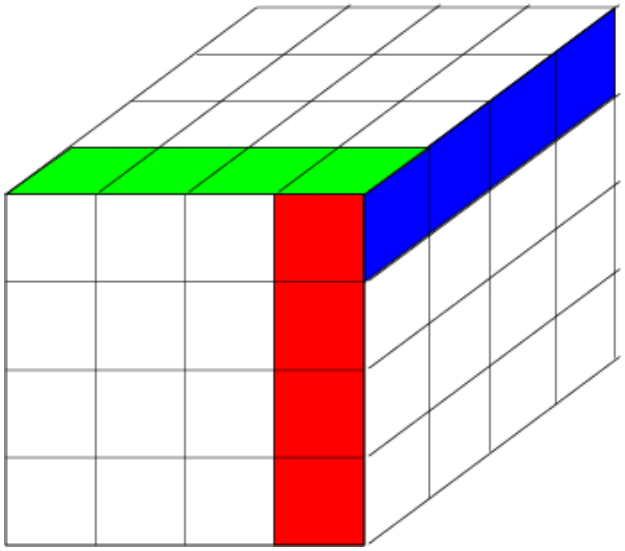
- Particle-mesh methods:
  - partition into short-range and long-range contributions
  - short-range via direct pairwise interactions
  - long-range:
    - interpolate atomic charge to 3d mesh
    - solve Poisson's equation on mesh (4 FFTs)
    - interpolate E-fields back to atoms

- FFTs scale as NlogN if cutoff is held fixed



Parallel FFTs

- 3d FFT is 3 sets of 1d FFTs
  - in parallel, 3d grid is distributed across procs
  - perform 1d FFTs on-processor
    - native library or FFTW (www.fftw.org)
  - 1d FFTs, transpose, 1d FFTs, transpose, ...
    - "transpose" = data transfer
    - transfer of entire grid is costly

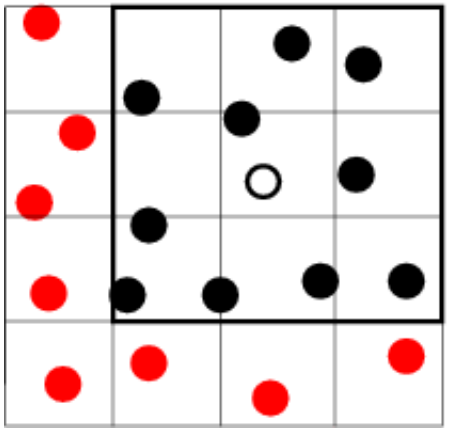


- FFTs for PPPM can scale poorly
  - on large # of procs and on clusters

- Good news: Cost of PPPM is only ~2x more than 8-10 Angstrom cutoff

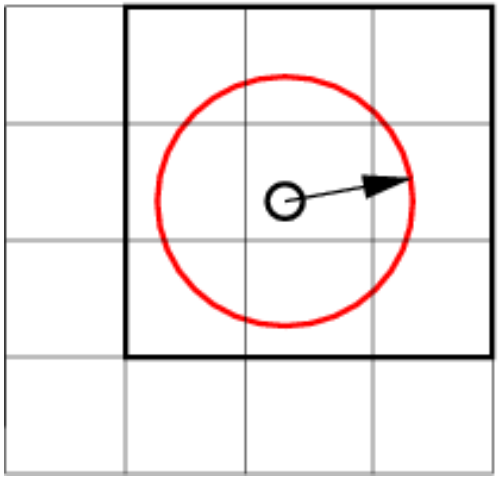
Neighbor Lists

- Problem: how to efficiently find neighbors within cutoff?
- Simple solution:
  - for each atom, test against all others
  - $O(N^2)$ algorithm
- Verlet lists:
  - *Verlet, Phys Rev, 159, p 98 (1967)*
  - $R_{neigh} = R_{force} + \Delta_{skin}$
  - build list: once every few timesteps
  - other timesteps: scan thru larger list
  - for neighbors within force cutoff
  - rebuild list: any atom moves 1/2 of skin
- Link-cells (bins):
  - *Hockney, et al, J Comp Phys, 14, p 148 (1974)*
  - grid simulation box into bins of size R_{force}
  - each timestep: search 27 bins for neighbors



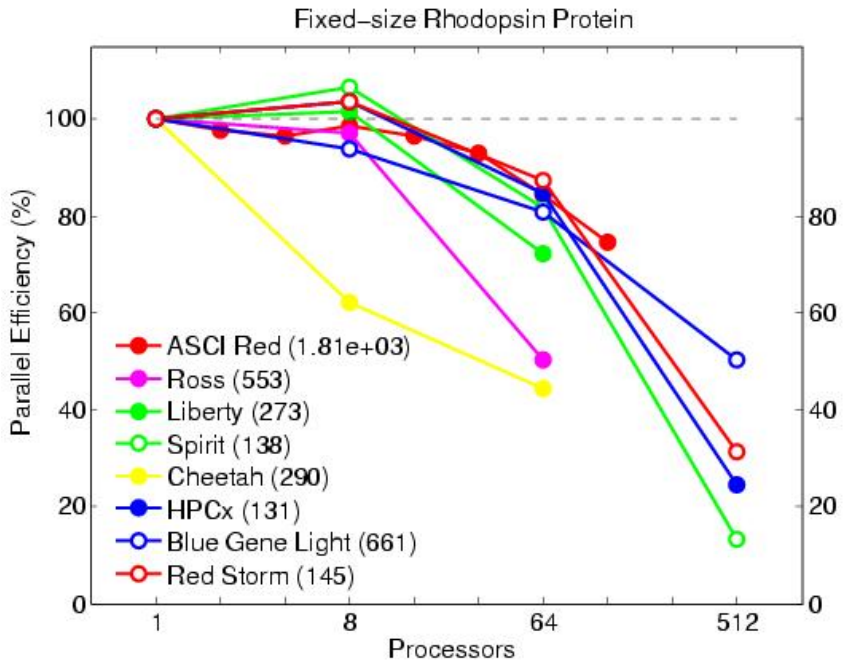
Neighbor Lists (continued)

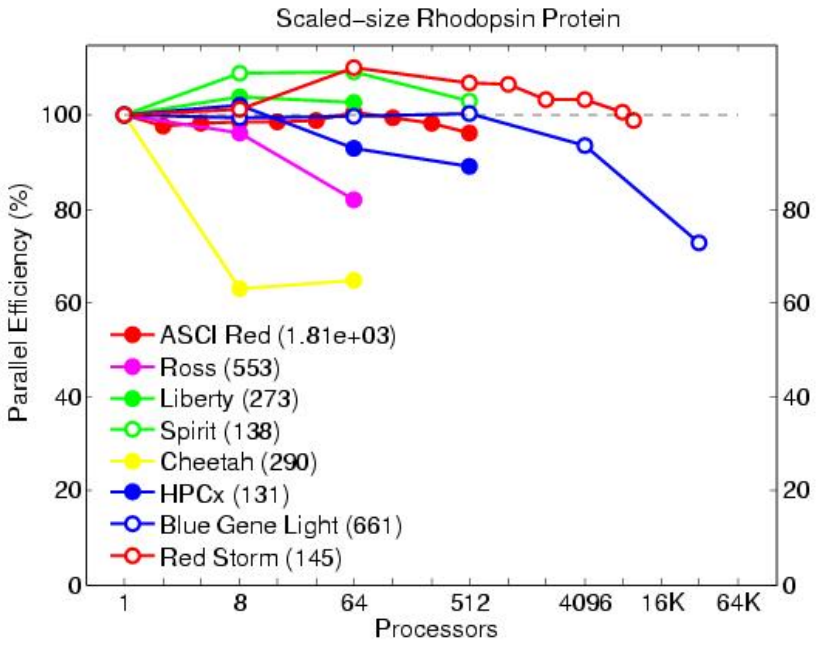
- Verlet list is ~6x savings over bins
  - $V_{sphere} = 4/3\ \pi\ r^3$
  - $V_{cube} = 27\ r^3$
- Fastest methods do both:
  - link-cell to build Verlet list
  - Verlet list on non-build timesteps
  - O(N) in CPU and memory
  - constant-density assumption
  - this is what LAMMPS implements



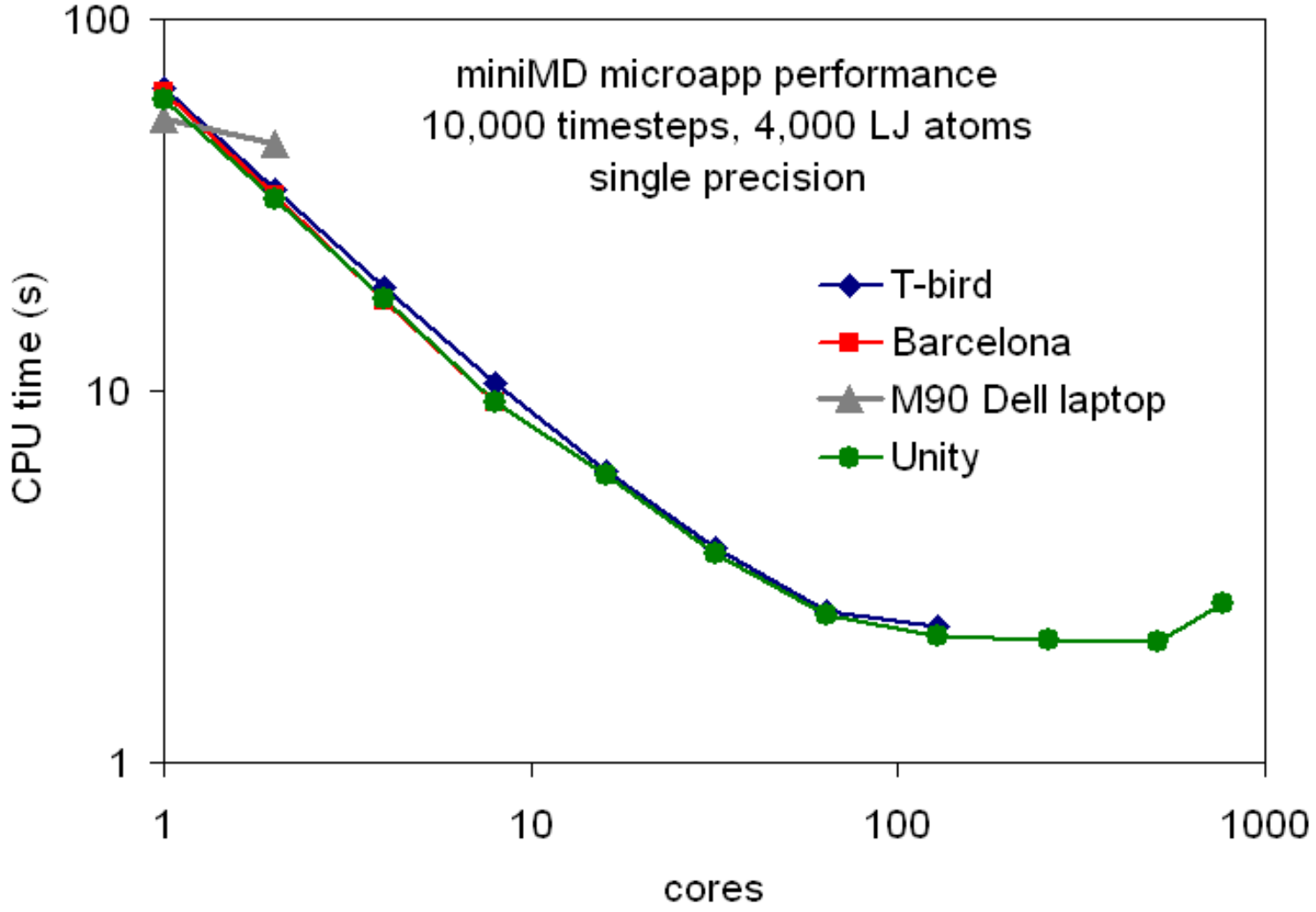
Parallel performance, rhodopsin

- Fixed-size (32K atoms) & scaled-size (32K/proc) parallel efficiencies
- Protein (rhodopsin) in solvated lipid bilayer
- Billions of atoms on 64K procs of Blue Gene or Red Storm
- Opteron speed: 4.5E-5 sec/atom/step (12x for metal, 25x for LJ)

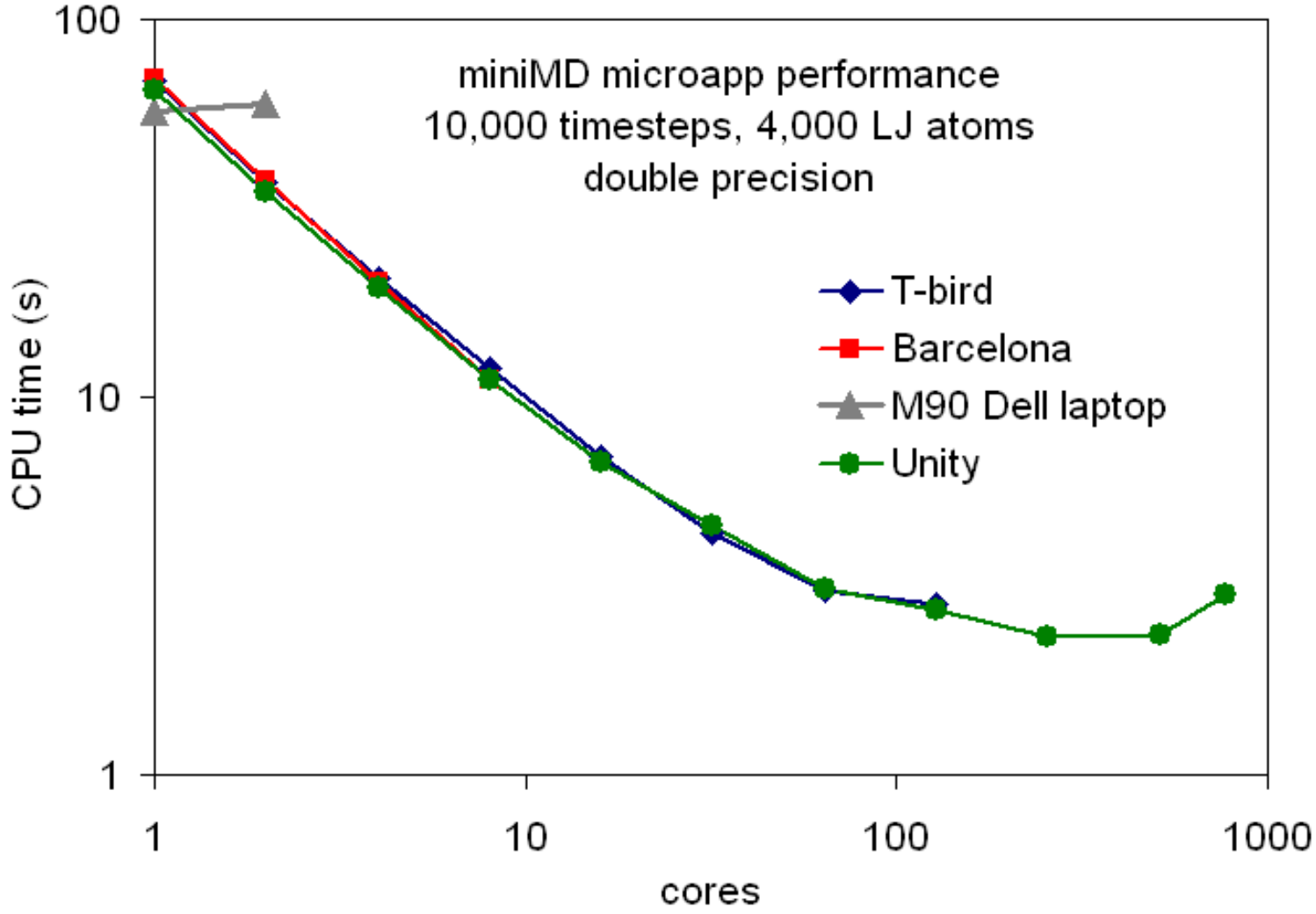




miniMD scaling results: single precision



miniMD scaling results: double precision



miniMD scaling results: timings breakdown