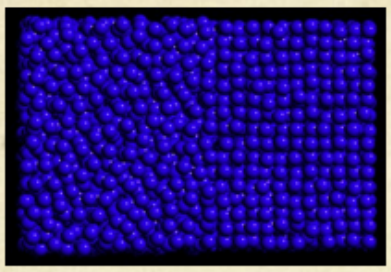
LAMMPS Features and Capabilities

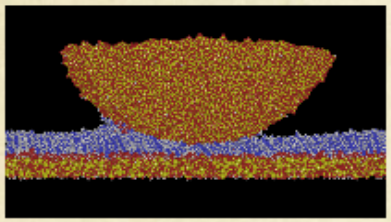
Steve Plimpton
Sandia National Labs
sjplimp@sandia.gov

LAMMPS Users and Developers Workshop
International Centre for Theoretical Physics (ICTP)
March 2014 - Trieste, Italy

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
Presentation: SAND2014-2239C

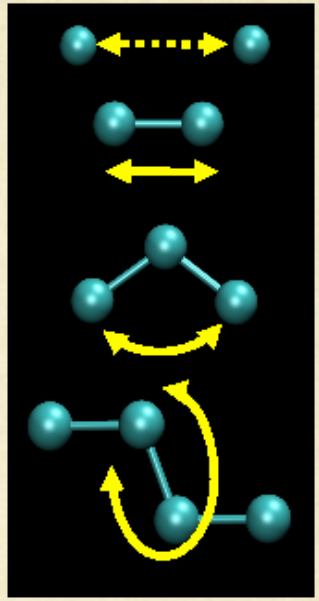
Classical MD in a nutshell





Initial positions and velocities

Interatomic potential



- Classical simulation technique
- Empirical interactions
- Evolve system, analyze

⇨

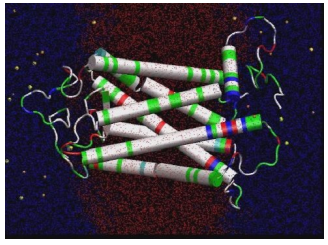
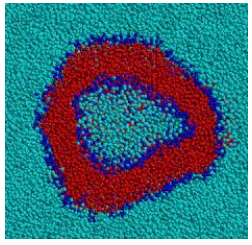
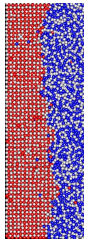
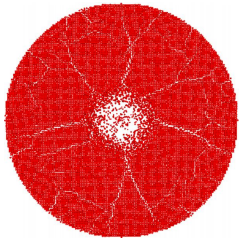
Positions and velocities at later times

Thanks to Aidan Thompson

LAMMPS from 10,000 meters

Large-scale Atomic/Molecular Massively Parallel Simulator
http://lammps.sandia.gov

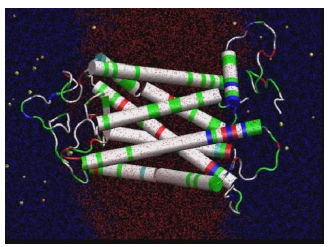
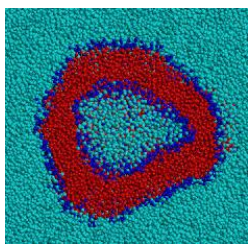
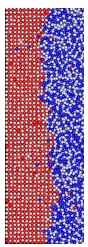
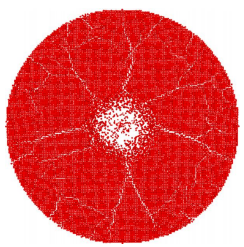
- Classical MD code
- Open source, portable C++
- 3-legged stool: soft matter, solids, mesoscale

LAMMPS from 10,000 meters

Large-scale Atomic/Molecular Massively Parallel Simulator
http://lammps.sandia.gov

- Classical MD code
- Open source, portable C++
- 3-legged stool: soft matter, solids, mesoscale

- Particle simulator at varying length and time scales
  electrons $\Rightarrow$ atomistic $\Rightarrow$ coarse-grained $\Rightarrow$ continuum
- Spatial-decomposition of simulation domain for parallelism
- MD, non-equilibrium MD, energy minimization
- GPU and OpenMP enhanced
- Can be coupled to other scales: QM, kMC, FE, CFD, ...

Reasons to use LAMMPS

1. Versatile
   - bio, materials, mesoscale
   - atomistic, coarse-grained, continuum
   - use with other codes, e.g. multiscale models
2. Good parallel performance
3. Easy to extend
   - Tuesday AM - Modifying & Extending LAMMPS
   - Wednesday PM - Hands-on: Writing new code for LAMMPS
4. Well documented
   - extensive web site
   - 1300 page manual
5. Active and supportive user community
   - 45K postings to mail list, 1500 subscribers
   - quick turn-around on Qs posted to mail list

Resources for learning LAMMPS

- Examples: about 35 sub-dirs under examples in distro
- Manual: doc/Manual.html
  - Intro, Commands, Packages, Accelerating
  - Howto, Modifying, Errors
- Alphabetized command list: one doc page per command
  - doc/Section_commands.html 3.5
- Web site: http://lammps.sandia.gov
  - Pictures, Movies - examples of others work
  - Papers - find a paper similar to what you want to model
  - Workshops - slides from LAMMPS simulation talks
- Mail list: search it, post to it
  - http://lammps.sandia.gov/mail.html
- These slides (more info than I can probably present!)

Structure of typical input scripts

1. Units and atom style
2. Create simulation box and atoms
   - region, create_box, create_atoms, region commands
     - lattice command vs box units
   - read_data command
     - data file is a text file
     - look at examples/micelle/data.micelle
     - see read_data doc page for full syntax
3. Define groups
4. Set attributes of atoms: mass, velocity
5. Pair style for atom interactions
6. Fixes for time integration and constraints
7. Computes for diagnostics
8. Output: thermo, dump, restart
9. Run or minimize
10. Rinse and repeat (script executed one command at a time)

Debugging an input script

LAMMPS tries hard to flag many kinds of errors and warnings

1. If an input command generates the error ...
   - % lmp_linux -echo screen < in.polymer
   - re-read the doc page for the command
2. For input, setup, run-time errors ...
   - search doc/Section_errors.html for text of error message
   - also for warnings, they are usually important
   - if specific input command causes problems, look for IMPORTANT NOTE info on doc page
   - look in the source code file at the line number
3. Search the mail list, others may have similar problem
   - google for: lammps-users fix npt, or error message

Debugging an input script

LAMMPS tries hard to flag many kinds of errors and warnings

1. If an input command generates the error ...
   - % lmp_linux -echo screen < in.polymer
   - re-read the doc page for the command
2. For input, setup, run-time errors ...
   - search doc/Section_errors.html for text of error message
   - also for warnings, they are usually important
   - if specific input command causes problems, look for IMPORTANT NOTE info on doc page
   - look in the source code file at the line number
3. Search the mail list, others may have similar problem
   - google for: lammps-users fix npt, or error message
4. Remember: an input script is like a program
   - start with small systems
   - start with one processor
   - turn-on complexity one command at a time
   - monitor thermo output, viz the results (use dump image)

Debug by examining screen output

```
LAMMPS (15 Aug 2013)
Lattice spacing in x,y,z = 1.28436 2.22457 1.28436
Created orthogonal box = (0 0 -0.321089)
            to (51.3743 22.2457 0.321089)
  4 by 1 by 1 MPI processor grid
Created 840 atoms
120 atoms in group lower
120 atoms in group upper
240 atoms in group boundary
600 atoms in group flow
Setting atom values ...
  120 settings made for type
Setting atom values ...
  120 settings made for type
Deleted 36 atoms, new total = 804
Deleted 35 atoms, new total = 769
```

Thermodynamic output

Look for blow-ups or NaNs, print every step if necessary

```
WARNING: Temperature for thermo pressure is not
         for group all (../thermo.cpp:436)
Setting up run ...
Memory usage per processor = 2.23494 Mbytes
Step Temp E_pair E_mol TotEng Press Volume
0 1.0004177 0 0 0.68689281 0.46210058 1143.0857
1000 1 -0.32494012 0 0.36166587 1.2240503 1282.5239
2000 1 -0.37815616 0 0.30844982 1.0642877 1312.5691
...
...
...
25000 1 -0.36649381 0 0.32011217 0.98366691 1451.5444
25000 1 -0.38890426 0 0.29770172 0.95284427 1455.9361
Loop time of 1.76555 on 4 procs for
         25000 steps with 769 atoms
```

Timing info

```
Loop time of 1.76555 on 4 procs for
        25000 steps with 769 atoms

Pair time (%) = 0.14617 (8.27903)
Neigh time (%) = 0.0467809 (2.64966)
Comm time (%) = 0.307951 (17.4422)
Outpt time (%) = 0.674575 (38.2078)
Other time (%) = 0.590069 (33.4213)
```

Run statistics

Per-processor values at end of run

```
Nlocal:  192.25 ave 242 max 159 min
Histogram:  2 0 0 0 0 1 0 0 0 1
Nghost:  43 ave 45 max 39 min
Histogram:  1 0 0 0 0 0 0 0 2 1
Neighs:  414 ave 588 max 284 min
Histogram:  2 0 0 0 0 0 1 0 0 1

Total # of neighbors = 1656
Ave neighs/atom = 2.15345
Neighbor list builds = 1641
Dangerous builds = 1
```

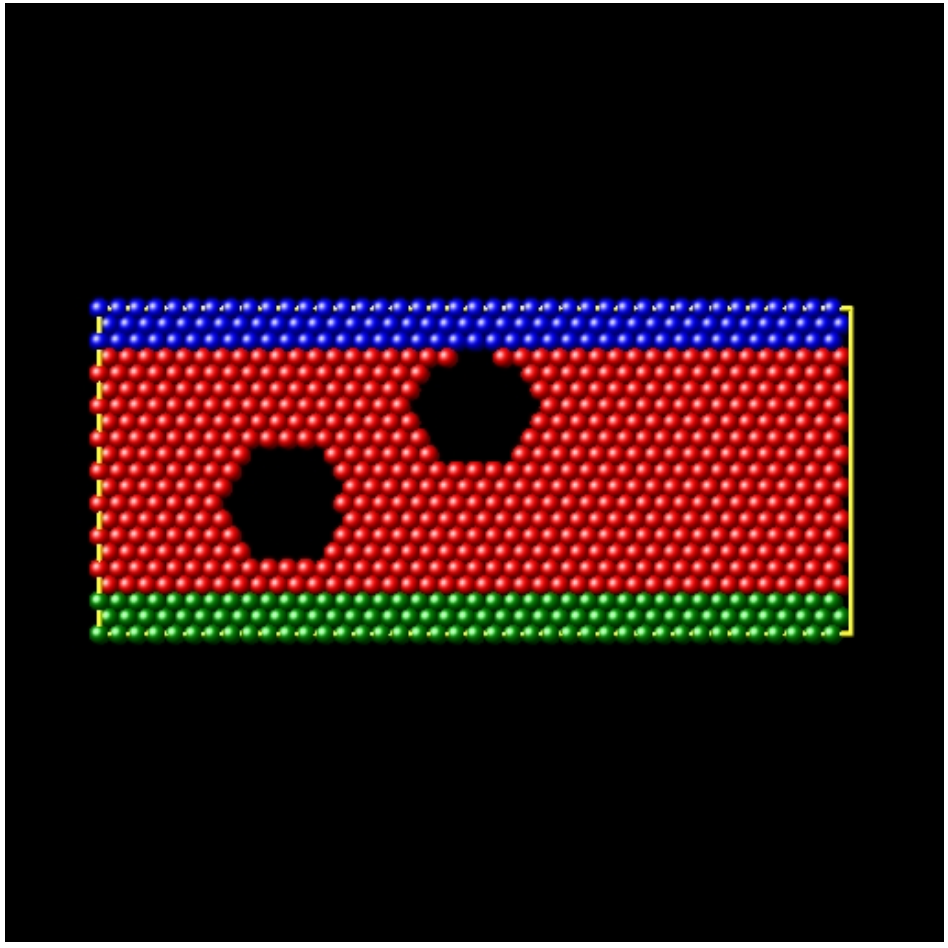
Debug by visualization - what does your system do?

Dump image for instant JPGs

- image.16500.jpg
- ImageMagick display
- Mac Preview

Make/view a movie

- ImageMagick
  convert *.jpg image.gif
- open in browser
  open -a Safari image.gif
- Mac QuickTime
  open image sequence
- Windows Media Player
- VMD, AtomEye, ...



Defining variables in input scripts

- Styles: index, loop, equal, atom, ...
  - variable x index run1 run2 run3 run4
  - variable x loop 100
  - variable x equal trap(f_JJ[3])*${scale}
  - variable x atom -(c_p[1]+c_p[2]+c_p[3])/(3*vol)

Defining variables in input scripts

- Styles: index, loop, equal, atom, ...
  - variable x index run1 run2 run3 run4
  - variable x loop 100
  - variable x equal trap(f_JJ[3])*${scale}
  - variable x atom -(c_p[1]+c_p[2]+c_p[3])/(3*vol)
- Formulas can be complex
  - see doc/variable.html
  - thermo keywords (temp, press, ...)
  - math operators & functions (sqrt, log, cos, ...)
  - group and region functions (count, xcm, fcm, ...)
  - various special functions (min, ave, trap, stride, stagger, ...)
  - per-atom vectors (x, vx, fx, ...)
  - output from computes, fixes, other variables
- Formulas can be time- and/or spatially-dependent

Using variables in input scripts

- Substitute in any command via $x or ${myVar}
- Can define them as command-line arguments
  - % lmp_linux -v myTemp 350.0 < in.polymer
- Use in next command to increment a variable
  - with jump command to create loops
- Many commands allow them as arguments
  - fix addforce 0.0 v_fy 1.0
  - dump_modify every v_count
  - region sphere 0.0 0.0 0.0 v_radius
- Allows time- and spatially-dependent commands

Power tools for input scripts

- Filename options:
  - dump.*.% for per-snapshot or per-processor output
  - read_data data.protein.gz
  - read_restart old.restart.*
- If/then/else via if command
- Insert another script via include command
  - useful for long list of parameters

Power tools for input scripts

- Filename options:
  - dump.*.% for per-snapshot or per-processor output
  - read_data data.protein.gz
  - read_restart old.restart.*
- If/then/else via if command
- Insert another script via include command
  - useful for long list of parameters
- Looping via next and jump commands
  - easy to run incrementally and stop when condition met
  - see examples on jump command doc page
- Invoke a shell command or external program
  - shell cd subdir1
  - shell my_analyze out.file $n ${param}
- Various ways to run multiple simulations from one script
  - see Section_howto 6.4 of manual

Example script for multiple runs

```
variable r equal random(1,1000000000,58798)
variable a loop 8
variable t index 0.8 0.85 0.9 0.95 1.0 1.05 1.1 1.15
log log.$a
read data.polymer
velocity all create $t $r
fix 1 all nvt $t $t 1.0
dump 1 all atom 1000 dump.$a.*
run 100000
next t
next a
jump in.polymer
```

Example script for multiple runs

```
variable r equal random(1,1000000000,58798)
variable a loop 8
variable t index 0.8 0.85 0.9 0.95 1.0 1.05 1.1 1.15
log log.$a
read data.polymer
velocity all create $t $r
fix 1 all nvt $t $t 1.0
dump 1 all atom 1000 dump.$a.*
run 100000
next t
next a
jump in.polymer
```

Run 8 simulations on 3 partitions until finished:

- change a,t to universe-style variables
- % mpirun -np 12 lmp_linux -p 3x4 -in in.polymer

Building systems: a pre-processing task

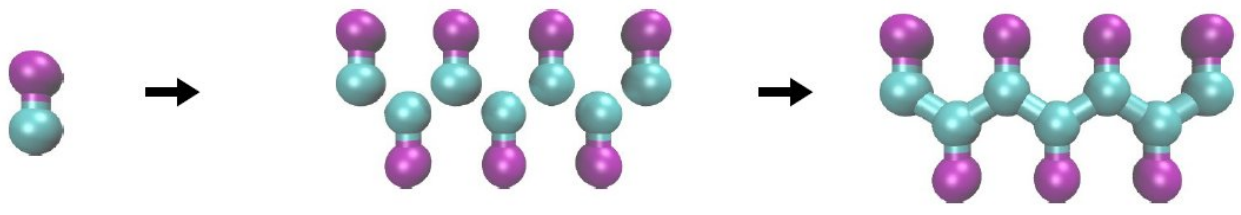
- In general, can be a hard problem!
- Molecular topology is an input to LAMMPS
  - get it from a builder, massage into LAMMPS format
  - auto-magical assignment of force-fields is also hard
- LAMMPS includes some basic pre-processors (tools dir)
  - bead-spring chain builder
  - ch2lmp = PDB to LAMMPS converter
  - amber2lmp = AMBER to LAMMPS converter
  - msi2lmp = Accelrys to LAMMPS converter

Building systems: a pre-processing task

- In general, can be a hard problem!
- Molecular topology is an input to LAMMPS
  - get it from a builder, massage into LAMMPS format
  - auto-magical assignment of force-fields is also hard
- LAMMPS includes some basic pre-processors (tools dir)
  - bead-spring chain builder
  - ch2lmp = PDB to LAMMPS converter
  - amber2lmp = AMBER to LAMMPS converter
  - msi2lmp = Accelrys to LAMMPS converter
- 3rd party builders and force-field generators
  - VMD TopoTools, Avogodro, PackMol, Moltemplate
  - Votca for CG force-field tabulation
  - http://lammps.sandia.gov/prepost.html
- Monte Carlo builders and force-field assignment
  - Towhee (configurational bias) & others
- Be willing to write system-building scripts yourself

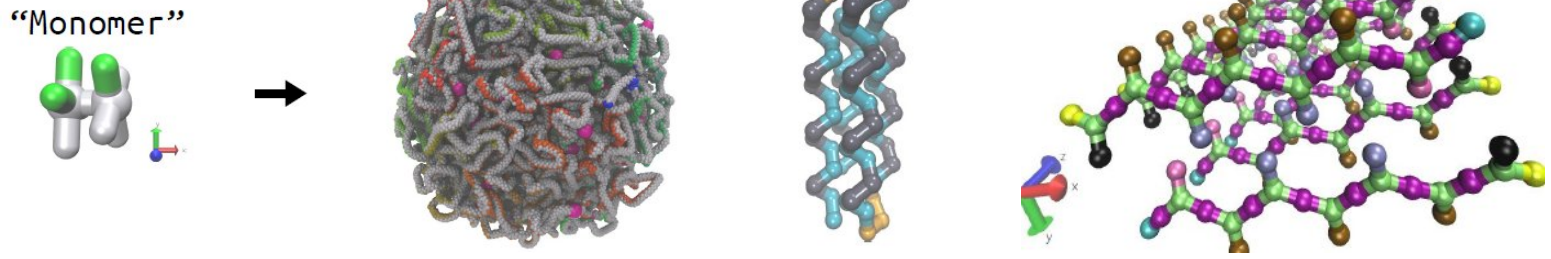
Moltemplate

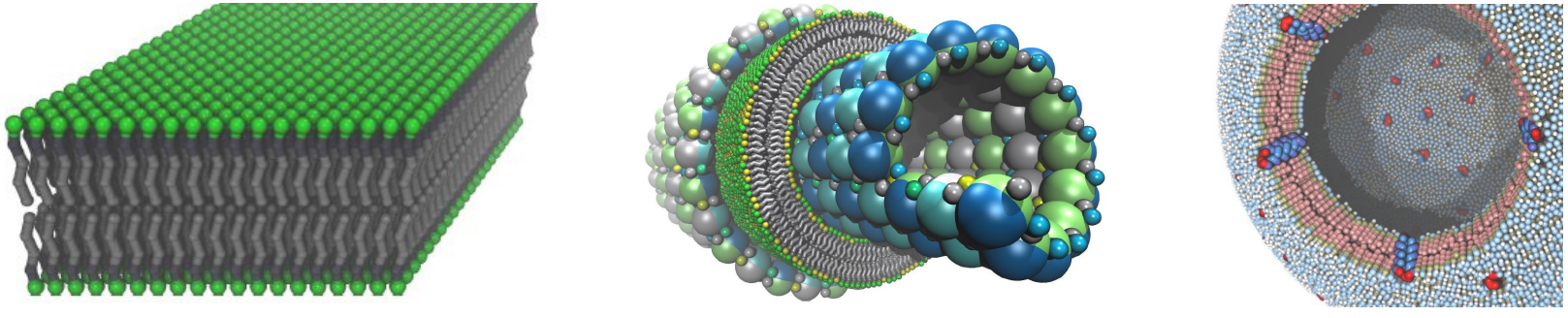
- http://www.moltemplate.org (Andrew Jewett, UCSB)
- Bundled with LAMMPS, designed to work with it
- Scripting language to build monomers/chains/systems hierarchically



- Provide atom charges & bond list
- Moltemplate generates angles, dihedrals, etc
- Also assigns force field params (only OPLS-AA currently)

More complex geometries with Moltemplate





Pair styles

LAMMPS lingo for interaction potentials

Pair styles

LAMMPS lingo for interaction potentials

- A pair style can be true pair-wise or many-body
  - LJ, Coulombic, Buckingham, Morse, Yukawa, ...
  - EAM, Tersoff, REBO, ReaxFF, ...
- Bond/angle/dihedral/improper styles = permanent bonds

Pair styles

LAMMPS lingo for interaction potentials

- A pair style can be true pair-wise or many-body
  - LJ, Coulombic, Buckingham, Morse, Yukawa, ...
  - EAM, Tersoff, REBO, ReaxFF, ...
- Bond/angle/dihedral/improper styles = permanent bonds
- Variants optimized for GPU and many-core
  - GPU, USER-CUDA, USER-OMP packages
  - lj/cut, lj/cut/gpu, lj/cut/cuda, lj/cut/omp
  - see doc/Section_accelerate.html
- Coulomb interactions included in pair style
  - lj/cut, lj/cut/coul/cut, lj/cut/coul/wolf, lj/cut/coul/long
  - done to optimize inner loop

Categories of potentials (pair styles) in LAMMPS

- All-atom: OPLS, CHARMM, AMBER, etc
- Charged systems:
  - pair lj/cut/coul/cut, lj/cut/coul/long + kspace_style
- UA: pair lj, pair coul, bond/angle/dihedral harmonic, etc
- Coarse-grained
  - FENE, DPD, SDK, granular, SPH, peri, colloid, lubricate, brownian, FLD
- Aspherical
  - gayberne, resquared, line, tri
- Tabulated (e.g. force matching)
  - pair table, bond table, angle table, etc
- Reactive: ReaxFF, COMB, AIREBO, other bond-order models
- Hybrid systems: pair hybrid and hybrid/overlay
  - polymers on metal surface
  - polymers with nano-particles
  - solid-solid interface between 2 materials

Pair styles

See doc/Section_commands.html for full list

none	hybrid	hybrid/overlay	adp
airebo	beck	body	bop
born	born/coul/long	born/coul/msm	born/coul/wolf
brownian	brownian/poly	buck	buck/coul/cut
buck/coul/long	buck/coul/msm	buck/long/coul/long	colloid
comb	coul/cut	coul/debye	coul/dsf
coul/long	coul/msm	coul/wolf	dpd
dpd/tstat	dsmc	eam	eam/alloy
eam/fs	eim	gauss	gayberne
gran/hertz/history	gran/hooke	gran/hooke/history	hbond/dreiding/lj
hbond/dreiding/morse	kim	lcbop	line/lj
lj/charmm/coul/charmm	lj/charmm/coul/charmm/implicit	lj/charmm/coul/long	lj/charmm/coul/msm
lj/class2	lj/class2/coul/cut	lj/class2/coul/long	lj/cut
lj/cut/coul/cut	lj/cut/coul/debye	lj/cut/coul/dsf	lj/cut/coul/long
lj/cut/coul/msm	lj/cut/dipole/cut	lj/cut/dipole/long	lj/cut/tip4p/cut
lj/cut/tip4p/long	lj/expand	lj/gromacs	lj/gromacs/coul/gromacs
lj/long/coul/long	lj/long/dipole/long	lj/long/tip4p/long	lj/smooth
lj/smooth/linear	lj96/cut	lubricate	lubricate/poly
lubricateU	lubricateU/poly	meam	mie/cut
morse	peri/lps	peri/pmb	peri/ves
reax	rebo	resquared	soft
sw	table	tersoff	tersoff/mod
tersoff/zbl	tip4p/cut	tip4p/long	tri/lj
yukawa	yukawa/colloid		

ed by users, which can be used if LAMMPS is built with the appropriate package.

awpmd/cut	coul/diel	eam/cd	edip
eff/cut	gauss/cut	list	lj/cut/dipole/sf
lj/sdk	lj/sdk/coul/long	lj/sdk/coul/msm	lj/sf
meam/spline	meam/sw/spline	nb3b/harmonic	reax/c
sph/heatconduction	sph/idealgas	sph/lj	sph/rhosum
sph/taitwater	sph/taitwater/morris	tersoff/table	

Pair styles

And they come in accelerated flavors: omp, gpu, cuda

adp/omp	airebo/omp	beck/omp	born/coul/long/cuda
born/coul/long/gpu	born/coul/long/omp	born/coul/msm/omp	born/coul/wolf/gpu
born/coul/wolf/omp	born/gpu	born/omp	brownian/omp
brownian/poly/omp	buck/coul/cut/cuda	buck/coul/cut/gpu	buck/coul/cut/omp
buck/coul/long/cuda	buck/coul/long/gpu	buck/coul/long/omp	buck/coul/msm/omp
buck/cuda	buck/long/coul/long/omp	buck/gpu	buck/omp
colloid/gpu	colloid/omp	comb/omp	coul/cut/omp
coul/debye/omp	coul/dsf/gpu	coul/long/gpu	coul/long/omp
coul/msm/omp	coul/wolf	dpd/omp	dpd/tstat/omp
eam/alloy/cuda	eam/alloy/gpu	eam/alloy/omp	eam/alloy/opt
eam/cd/omp	eam/cuda	eam/fs/cuda	eam/fs/gpu
eam/fs/omp	eam/fs/opt	eam/gpu	eam/omp
eam/opt	edip/omp	eim/omp	gauss/gpu
gauss/omp	gayberne/gpu	gayberne/omp	gran/hertz/history/omp
gran/hooke/cuda	gran/hooke/history/omp	gran/hooke/omp	hbond/dreiding/lj/omp
hbond/dreiding/morse/omp	line/lj/omp	lj/charmm/coul/charmm/cuda	lj/charmm/coul/charmm/omp
lj/charmm/coul/charmm/implicit/cuda	lj/charmm/coul/charmm/implicit/omp	lj/charmm/coul/long/cuda	lj/charmm/coul/long/gpu
lj/charmm/coul/long/omp	lj/charmm/coul/long/opt	lj/class2/coul/cut/cuda	lj/class2/coul/cut/omp
lj/class2/coul/long/cuda	lj/class2/coul/long/gpu	lj/class2/coul/long/omp	lj/class2/coul/msm/omp
lj/class2/cuda	lj/class2/gpu	lj/class2/omp	lj/long/coul/long/omp
lj/cut/coul/cut/cuda	lj/cut/coul/cut/gpu	lj/cut/coul/cut/omp	lj/cut/coul/debye/cuda
lj/cut/coul/debye/gpu	lj/cut/coul/debye/omp	lj/cut/coul/dsf/gpu	lj/cut/coul/long/cuda
lj/cut/coul/long/gpu	lj/cut/coul/long/omp	lj/cut/coul/long/opt	lj/cut/coul/msm/opt
lj/cut/cuda	lj/cut/dipole/cut/gpu	lj/cut/dipole/cut/omp	lj/cut/dipole/sf/gpu
lj/cut/dipole/sf/omp	lj/cut/experimental/cuda	lj/cut/gpu	lj/cut/omp
lj/cut/opt	lj/cut/tip4p/cut/omp	lj/cut/tip4p/long/omp	lj/cut/tip4p/long/opt
lj/expand/cuda	lj/expand/gpu	lj/expand/omp	lj/gromacs/coul/gromacs/cuda
lj/gromacs/coul/gromacs/omp	lj/gromacs/cuda	lj/gromacs/omp	lj/long/coul/long/opt
lj/sdk/gpu	lj/sdk/omp	lj/sdk/coul/long/gpu	lj/sdk/coul/long/omp
lj/sdk/coul/msm/omp	lj/sf/omp	lj/smooth/cuda	lj/smooth/omp
lj/smooth/linear/omp	lj96/cut/cuda	lj96/cut/gpu	lj96/cut/omp
lubricate/omp	lubricate/poly/omp	meam/spline/omp	morse/cuda
morse/gpu	morse/omp	morse/opt	nb3b/harmonic/omp
peri/lps/omp	peri/pmb/omp	rebo/omp	resquared/gpu
resquared/omp	soft/omp	sw/cuda	sw/omp
table/gpu	table/omp	tersoff/cuda	tersoff/omp
tersoff/table/omp	tersoff/zbl/omp	tip4p/cut/omp	tip4p/long/omp
tri/lj/omp	yukawa/gpu	yukawa/omp	yukawa/colloid/gpu
yukawa/colloid/omp			

Pair styles

See doc/pair_style.html for one-line descriptions

- pair_style none - turn off pairwise interactions
- pair_style hybrid - multiple styles of pairwise interactions
- pair_style hybrid/overlay - multiple styles of superposed pairwise interactions

- pair_style adp - angular dependent potential (ADP) of Mishin
- pair_style airebo - AIREBO potential of Stuart
- pair_style beck - Beck potential
- pair_style body - interactions between body particles
- pair_style bop - BOP potential of Pettifor
- pair_style born - Born-Mayer-Huggins potential
- pair_style born/coul/long - Born-Mayer-Huggins with long-range Coulombics
- pair_style born/coul/msm - Born-Mayer-Huggins with long-range MSM Coulombics
- pair_style born/coul/wolf - Born-Mayer-Huggins with Coulombics via Wolf potential
- pair_style brownian - Brownian potential for Fast Lubrication Dynamics
- pair_style brownian/poly - Brownian potential for Fast Lubrication Dynamics with polydispersity
- pair_style buck - Buckingham potential
- pair_style buck/coul/cut - Buckingham with cutoff Coulomb
- pair_style buck/coul/long - Buckingham with long-range Coulombics
- pair_style buck/coul/msm - Buckingham long-range MSM Coulombics
- pair_style buck/long/coul/long - long-range Buckingham with long-range Coulombics
- pair_style colloid - integrated colloidal potential
- pair_style comb - charge-optimized many-body (COMB) potential
- pair_style coul/cut - cutoff Coulombic potential
- pair_style coul/debye - cutoff Coulombic potential with Debye screening
- pair_style coul/dsf - Coulombics via damped shifted forces
- pair_style coul/long - long-range Coulombic potential
- pair_style coul/msm - long-range MSM Coulombics
- pair_style coul/wolf - Coulombics via Wolf potential
- pair_style dipole/cut - point dipoles with cutoff
- pair_style dpd - dissipative particle dynamics (DPD)
- pair_style dpd/tstat - DPD thermostatting
- pair_style dsmc - Direct Simulation Monte Carlo (DSMC)
- pair_style eam - embedded atom method (EAM)
- pair_style eam/alloy - alloy EAM
- pair_style eam/fs - Finnis-Sinclair EAM
- pair_style eim - embedded ion method (EIM)
- pair_style gauss - Gaussian potential
- pair_style gayberne - Gay-Berne ellipsoidal potential
- pair_style gran/hertz/history - granular potential with Hertzian interactions
- pair_style gran/hooke - granular potential with history effects
- pair_style gran/hooke/history - granular potential without history effects

Relative computational cost of different potentials

See lammps.sandia.gov/bench.html#potentials

Potential	System	Atoms	Timestep	CPU	LJ Ratio
Granular	chute flow	32000	0.0001 tau	5.08e-7	0.34x
FENE bead/spring	polymer melt	32000	0.012 tau	5.32e-7	0.36x
Lennard-Jones	LJ liquid	32000	0.005 tau	1.48e-6	1.0x
DPD	pure solvent	32000	0.04 tau	2.16e-6	1.46x
EAM	bulk Cu	32000	5 fmsec	3.59e-6	2.4x
Tersoff	bulk Si	32000	1 fmsec	6.01e-6	4.1x
Stillinger-Weber	bulk Si	32000	1 fmsec	6.10e-6	4.1x
EIM	crystalline NaCl	32000	0.5 fmsec	9.69e-6	6.5x
SPC/E	liquid water	36000	2 fmsec	1.43e-5	9.7x
CHARMM + PPPM	solvated protein	32000	2 fmsec	2.01e-5	13.6x
MEAM	bulk Ni	32000	5 fmsec	2.31e-5	15.6x
Peridynamics	glass fracture	32000	22.2 nsec	2.42e-5	16.4x
Gay-Berne	ellipsoid mixture	32768	0.002 tau	4.09e-5	28.3x
AIREBO	polyethylene	32640	0.5 fmsec	8.09e-5	54.7x
COMB	crystalline SiO2	32400	0.2 fmsec	4.19e-4	284x
eFF	H plasma	32000	0.001 fmsec	4.52e-4	306x
ReaxFF	PETN crystal	16240	0.1 fmsec	4.99e-4	337x
ReaxFF/C	PETN crystal	32480	0.1 fmsec	2.73e-4	185x
VASP/small	water	192/512	0.3 fmsec	26.2	17.7e6
VASP/medium	CO2	192/1024	0.8 fmsec	252	170e6
VASP/large	Xe	432/3456	2.0 fmsec	1344	908e6

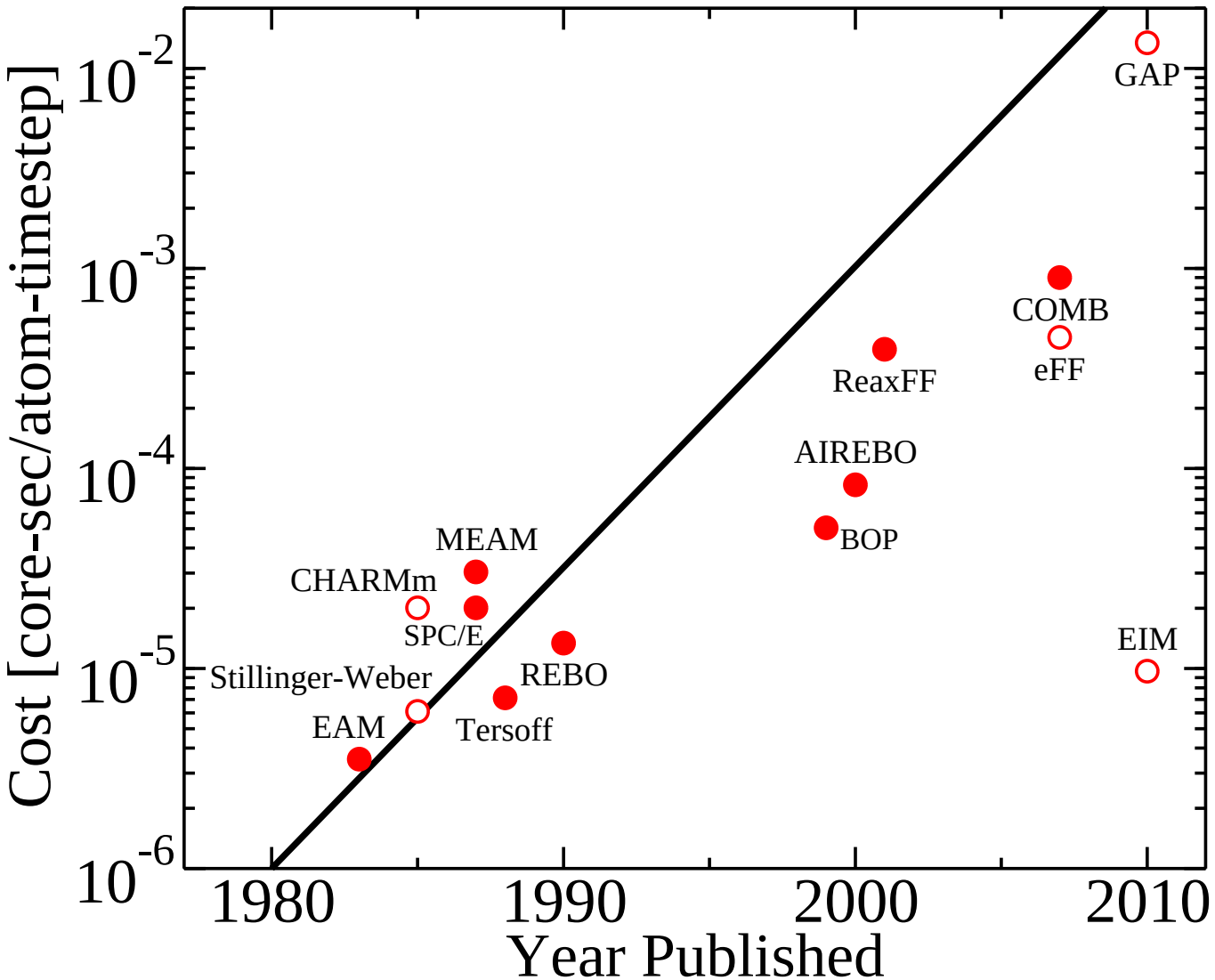
Relative computational cost of different potentials

See lammps.sandia.gov/bench.html#potentials

Potential	System	Atoms	Timestep	CPU	LJ Ratio
Granular	chute flow	32000	0.0001 tau	5.08e-7	0.34x
FENE bead/spring	polymer melt	32000	0.012 tau	5.32e-7	0.36x
Lennard-Jones	LJ liquid	32000	0.005 tau	1.48e-6	1.0x
DPD	pure solvent	32000	0.04 tau	2.16e-6	1.46x
EAM	bulk Cu	32000	5 fmsec	3.59e-6	2.4x
Tersoff	bulk Si	32000	1 fmsec	6.01e-6	4.1x
Stillinger-Weber	bulk Si	32000	1 fmsec	6.10e-6	4.1x
EIM	crystalline NaCl	32000	0.5 fmsec	9.69e-6	6.5x
SPC/E	liquid water	36000	2 fmsec	1.43e-5	9.7x
CHARMM + PPPM	solvated protein	32000	2 fmsec	2.01e-5	13.6x
MEAM	bulk Ni	32000	5 fmsec	2.31e-5	15.6x
Peridynamics	glass fracture	32000	22.2 nsec	2.42e-5	16.4x
Gay-Berne	ellipsoid mixture	32768	0.002 tau	4.09e-5	28.3x
AIREBO	polyethylene	32640	0.5 fmsec	8.09e-5	54.7x
COMB	crystalline SiO2	32400	0.2 fmsec	4.19e-4	284x
eFF	H plasma	32000	0.001 fmsec	4.52e-4	306x
ReaxFF	PETN crystal	16240	0.1 fmsec	4.99e-4	337x
ReaxFF/C	PETN crystal	32480	0.1 fmsec	2.73e-4	185x
VASP/small	water	192/512	0.3 fmsec	26.2	17.7e6
VASP/medium	CO2	192/1024	0.8 fmsec	252	170e6
VASP/large	Xe	432/3456	2.0 fmsec	1344	908e6

- Estimate CPU cost for system size & timesteps you need
- Assume good parallel scalability if have 1000+ atoms/core

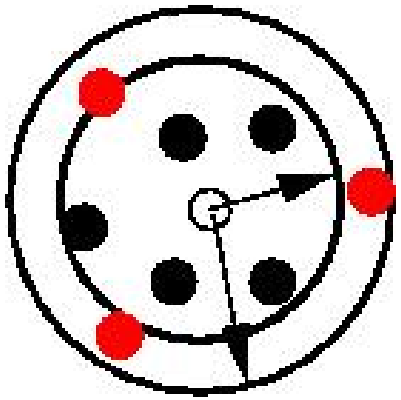
Moore's Law for potentials

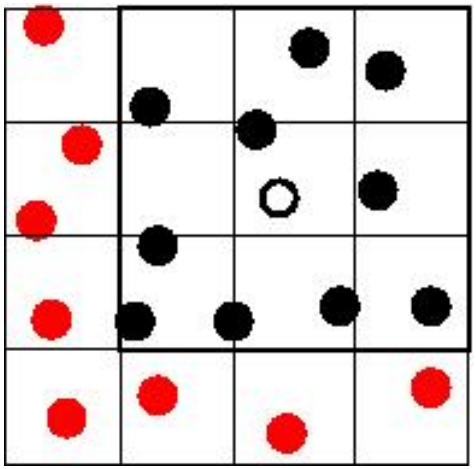


Neighbor lists in LAMMPS

Problem: how to efficiently find neighbors within cutoff?

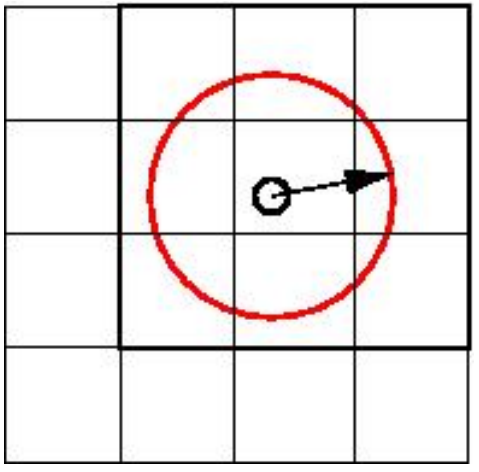
- For each atom, test against all others
  - $O(N^2)$ algorithm
- Verlet lists:
  - *Verlet, Phys Rev, 159, p 98 (1967)*
  - $R_{neigh} = R_{force} + \Delta_{skin}$
  - build list: once every few timesteps
  - other timesteps: scan larger list for neighbors within force cutoff
  - rebuild: any atom moves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





Neighbor lists (continued)

- Verlet list is ~6x savings over bins
  - $V_{sphere} = 4/3\ \pi r^3$
  - $V_{cube} = 27\ r^3$



- LAMMPS does both
  - link-cell to build Verlet list
  - use Verlet list on non-build timesteps
  - $O(N)$ in CPU and memory
  - constant-density assumption

Bond styles (also angle, dihedral, improper)

- Used for molecules with fixed bonds
  - Fix bond/break and bond_style quartic can break them
  - Fix bond/create can add them (e.g. cross-linking)
- To learn what bond styles LAMMPS has ...
  where would you look?

Bond styles (also angle, dihedral, improper)

- Used for molecules with fixed bonds
  - Fix bond/break and bond_style quartic can break them
  - Fix bond/create can add them (e.g. cross-linking)
- To learn what bond styles LAMMPS has ...
  where would you look?
- doc/Section_commands.html or doc/bond_style.html

none	hybrid	class2	fene
fene/expand	harmonic	morse	nonlinear
quartic	table		

ich can be used if LAMMPS is built with the appropriate package.

harmonic/shift	harmonic/shift/cut

e used if LAMMPS is built with the appropriate accelerated package.

class2/omp	fene/omp	fene/expand/omp	harmonic/omp
harmonic/shift/omp	harmonic/shift /cut/omp	morse/omp	nonlinear/omp
quartic/omp	table/omp		

- bond_style none - turn off bonded interactions
- bond_style hybrid - define multiple styles of bond interactions

- bond_style class2 - COMPASS (class 2) bond
- bond_style fene - FENE (finite-extensible non-linear elastic) bond
- bond_style fene/expand - FENE bonds with variable size particles
- bond_style harmonic - harmonic bond
- bond_style morse - Morse bond
- bond_style nonlinear - nonlinear bond
- bond_style quartic - breakable quartic bond
- bond_style table - tabulated by bond length

Long-range Coulombics

KSpace style in LAMMPS lingo, see doc/kspace_style.html

- Options:
  - traditional Ewald, scales as $O(N^{3/2})$
  - PPPM (like PME), scales as $O(N \log(N))$
  - MSM, scales as $O(N)$, lj/cut/coul/msm
- Additional options:
  - non-periodic, PPPM (z) vs MSM (xyz)
  - long-range dispersion (LJ)

Long-range Coulombics

KSpace style in LAMMPS lingo, see doc/kspace_style.html

- Options:
  - traditional Ewald, scales as $O(N^{3/2})$
  - PPPM (like PME), scales as $O(N \log(N))$
  - MSM, scales as $O(N)$, lj/cut/coul/msm
- Additional options:
  - non-periodic, PPPM (z) vs MSM (xyz)
  - long-range dispersion (LJ)
- PPPM is fastest choice for most systems
  - FFTs can scale poorly for large processor counts
- MSM can be faster for low-accuracy or large proc counts

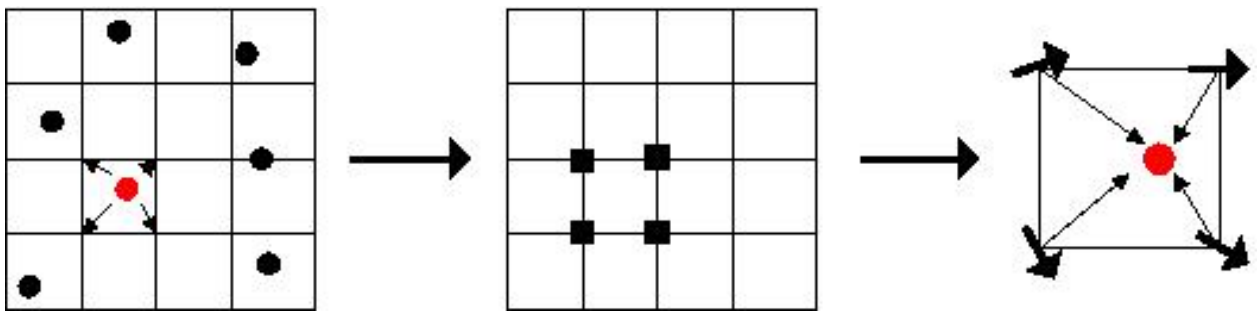
Long-range Coulombics

KSpace style in LAMMPS lingo, see doc/kspace_style.html

- Options:
  - traditional Ewald, scales as $O(N^{3/2})$
  - PPPM (like PME), scales as $O(N \log(N))$
  - MSM, scales as $O(N)$, lj/cut/coul/msm
- Additional options:
  - non-periodic, PPPM (z) vs MSM (xyz)
  - long-range dispersion (LJ)
- PPPM is fastest choice for most systems
  - FFTs can scale poorly for large processor counts
- MSM can be faster for low-accuracy or large proc counts
- Ways to speed-up long-range calculations:
  - see doc/Section_accelerate.html
  - cutoff & accuracy settings adjust Real vs KSpace work
  - kspace_style pppm/stagger for PPPM
  - kspace_modify diff ad for smoothed PPPM
  - run_style verlet/split

PPPM (particle-particle particle-mesh) in LAMMPS

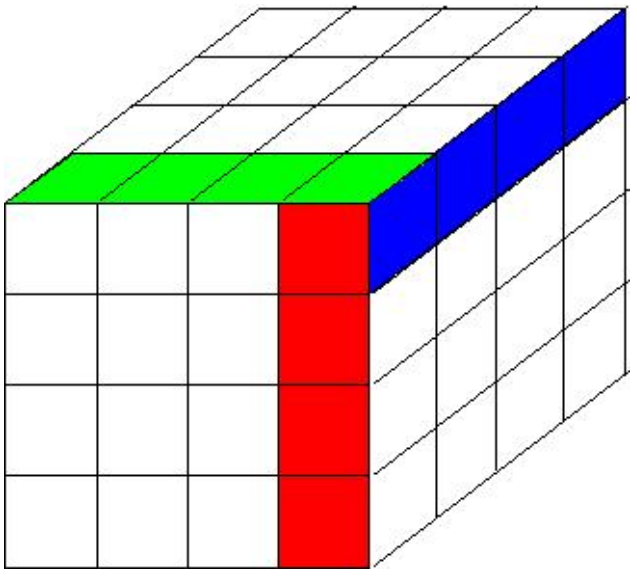
- *Hockney & Eastwood, Comp Sim Using Particles (1988)*
- *Darden, et al, J Chem Phys, 98, p 10089 (1993).*
- Like Ewald, except sum over periodic images evaluated:
  - interpolate atomic charge to 3d mesh
  - solve Poisson's equation on mesh (4 FFTs)
  - interpolate E-fields back to atoms



- User-specified accuracy + cutoff $\Rightarrow$ ewald-G + mesh-size
- Scales as $N\sqrt{\log(N)}$ if grow cutoff with N
- Scales as $N \log(N)$ if cutoff held fixed

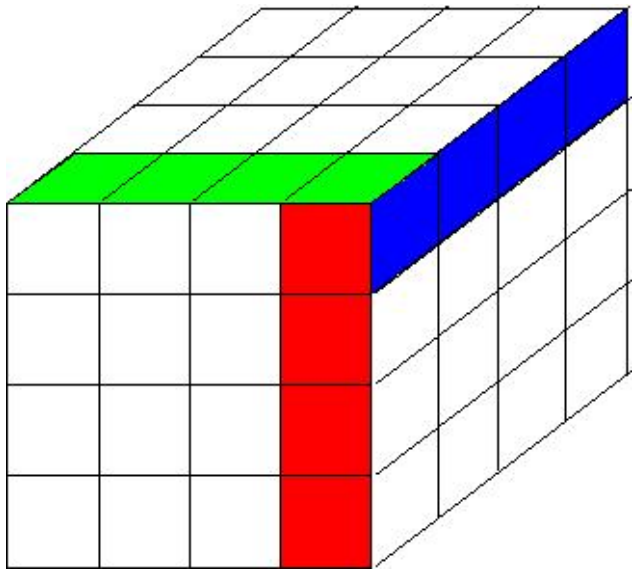
Parallel FFTs in LAMMPS

- 3d FFT is 3 sets of 1d FFTs
  - in parallel, 3d grid is distributed across procs
  - 1d FFTs on-processor
  - native library or FFTW (www.fftw.org)
  - multiple transposes of 3d grid
  - data transfer can be costly
- FFTs for PPPM can scale poorly
  - on large # of procs and on clusters



Parallel FFTs in LAMMPS

- 3d FFT is 3 sets of 1d FFTs
  - in parallel, 3d grid is distributed across procs
  - 1d FFTs on-processor
  - native library or FFTW (www.fftw.org)
  - multiple transposes of 3d grid
  - data transfer can be 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 Ang cutoff

Fixes

Most flexible feature in LAMMPS
Allows control of what happens when within each timestep

Loop over timesteps:

communicate ghost atoms

build neighbor list (once in a while)
compute forces
communicate ghost forces

output to screen and files

Fixes

Most flexible feature in LAMMPS
Allows control of what happens when within each timestep

Loop over timesteps:

- fix initial — NVE, NVT, NPT, rigid-body integration
- communicate ghost atoms
- fix neighbor — insert particles
- build neighbor list (once in a while)
- compute forces
- communicate ghost forces
- fix force — SHAKE, langevin drag, wall, spring, gravity
- fix final — NVE, NVT, NPT, rigid-body integration
- fix end — volume & T rescaling, diagnostics
- output to screen and files

Fixes

- 100+ fixes in LAMMPS
- You choose what group of atoms to apply fix to
- Already saw some in obstacle example:
  - fix 1 all nve
  - fix 2 flow temp/rescale 200 1.0 1.0 0.02 1.0
  - fix 3 lower setforce 0.0 0.0 0.0
  - fix 5 upper aveforce 0.0 -0.5 0.0
  - fix 6 flow addforce 1.0 0.0 0.0

Fixes

- 100+ fixes in LAMMPS
- You choose what group of atoms to apply fix to
- Already saw some in obstacle example:
  - fix 1 all nve
  - fix 2 flow temp/rescale 200 1.0 1.0 0.02 1.0
  - fix 3 lower setforce 0.0 0.0 0.0
  - fix 5 upper aveforce 0.0 -0.5 0.0
  - fix 6 flow addforce 1.0 0.0 0.0
- To learn what fix styles LAMMPS has ...
  where would you look?

Fixes

- 100+ fixes in LAMMPS
- You choose what group of atoms to apply fix to
- Already saw some in obstacle example:
  - fix 1 all nve
  - fix 2 flow temp/rescale 200 1.0 1.0 0.02 1.0
  - fix 3 lower setforce 0.0 0.0 0.0
  - fix 5 upper aveforce 0.0 -0.5 0.0
  - fix 6 flow addforce 1.0 0.0 0.0
- To learn what fix styles LAMMPS has ...
  where would you look?
- doc/Section_commands.html or doc/fix.html

Fixes

- 100+ fixes in LAMMPS
- You choose what group of atoms to apply fix to
- Already saw some in obstacle example:
  - fix 1 all nve
  - fix 2 flow temp/rescale 200 1.0 1.0 0.02 1.0
  - fix 3 lower setforce 0.0 0.0 0.0
  - fix 5 upper aveforce 0.0 -0.5 0.0
  - fix 6 flow addforce 1.0 0.0 0.0
- To learn what fix styles LAMMPS has ...
  where would you look?
- doc/Section_commands.html or doc/fix.html
- If you familiarize yourself with fixes,
  you'll know many things LAMMPS can do
- Many fixes store output accessible by other commands
  - rigid body COM
  - thermostat energy
  - forces before modified

Computes

- ∼75 computes in LAMMPS
- Calculate some property of system, in parallel
- Always for the current timestep
- To learn what compute styles LAMMPS has ...

Computes

- ∼75 computes in LAMMPS
- Calculate some property of system, in parallel
- Always for the current timestep
- To learn what compute styles LAMMPS has ...
  doc/Section_commands.html or doc/compute.html

angle/local	atom/molecule	body/local	bond/local	centro/atom	cluster/atom
cna/atom	com	com/molecule	contact/atom	coord/atom	damage/atom
dihedral/local	displace/atom	erotate/asphere	erotate/sphere	erotate/sphere/atom	event/displace
group/group	gyration	gyration/molecule	heat/flux	improper/local	inertia/molecule
ke	ke/atom	msd	msd/molecule	msd/nongauss	pair
pair/local	pe	pe/atom	pressure	property/atom	property/local
property/molecule	rdf	reduce	reduce/region	slice	stress/atom
temp	temp/asphere	temp/com	temp/deform	temp/partial	temp/profile
temp/ramp	temp/region	temp/sphere	ti	voronoi/atom	

ributed by users, which can be used if LAMMPS is built with the appropriate package.

ackland/atom	basal/atom	ke/eff	ke/atom/eff	meso_e/atom	meso_rho/atom
meso_t/atom	temp/eff	temp/deform/eff	temp/region/eff	temp/rotate	

e styles, which can be used if LAMMPS is built with the appropriate accelerated package.

pe/cuda	pressure/cuda	temp/cuda	temp/partial/cuda

Computes

- Key point:
  - computes store their results
  - other commands invoke them and use the results
  - e.g. thermo output, dumps, fixes
- Output of computes: (discussion in section 6.15 of manual)
  - global vs per-atom vs local
  - scalar vs vector vs array
  - extensive vs intensive values

Computes

- Key point:
  - computes store their results
  - other commands invoke them and use the results
  - e.g. thermo output, dumps, fixes
- Output of computes: (discussion in section 6.15 of manual)
  - global vs per-atom vs local
  - scalar vs vector vs array
  - extensive vs intensive values
- Examples:
  - temp & pressure = global scalar or vector
  - pe/atom = potential energy per atom (vector)
  - displace/atom = displacement per atom (array)
  - pair/local & bond/local = per-neighbor or per-bond info
- Many computes are useful with averaging fixes:
  - fix ave/time, fix ave/spatial, fix ave/atom
  - fix ave/histo, fix ave/correlate

Thermo output

One line of output every N timesteps to screen and log file

- See doc/thermo_style.html

Thermo output

One line of output every N timesteps to screen and log file

- See doc/thermo_style.html
- Any scalar can be output:
  - dozens of keywords: temp, pyy, eangle, lz, cpu
  - any output of a compute or fix: c_ID, f_ID[N], c_ID[N][M]
    - fix ave/time stores time-averaged quantities
  - equal-style variable: v_MyVar
  - one value from atom-style variable: v_xx[N]
  - any property for one atom: q, fx, quat, etc
  - useful for debugging or post-analysis

Thermo output

One line of output every N timesteps to screen and log file

- See doc/thermo_style.html
- Any scalar can be output:
  - dozens of keywords: temp, pyy, eangle, lz, cpu
  - any output of a compute or fix: c_ID, f_ID[N], c_ID[N][M]
    - fix ave/time stores time-averaged quantities
  - equal-style variable: v_MyVar
  - one value from atom-style variable: v_xx[N]
  - any property for one atom: q, fx, quat, etc
  - useful for debugging or post-analysis
- Post-process via:
  - tools/python/logplot.py log.lammps X Y (via GnuPlot)
  - tools/python/log2txt.py log.lammps data.txt X Y ...
  - Pizza.py log tool
    - can read thermo output across multiple runs
  - tools/xmgrace/README and one-liners and auto-plotter

Dump output

Snapshot of per-atom values every N timesteps

- See doc/dump.html

Dump output

Snapshot of per-atom values every N timesteps

- See doc/dump.html
- Styles
  - atom, custom (both native LAMMPS)
    - VMD will auto-read if file named *.lammpstraj
  - xyz for coords only
  - cfg for AtomEye
  - DCD, XTC for CHARMM, NAMD, GROMACS
    - useful for back-and-forth runs and analysis

Dump output

Snapshot of per-atom values every N timesteps

- See doc/dump.html
- Styles
  - atom, custom (both native LAMMPS)
    - VMD will auto-read if file named *.lammpstraj
  - xyz for coords only
  - cfg for AtomEye
  - DCD, XTC for CHARMM, NAMD, GROMACS
    - useful for back-and-forth runs and analysis
- Two additional styles
  - local: per-neighbor, per-bond, etc info
  - image: instant JPG/PPM picture, rendered in parallel

Dump output

- Any per-atom quantity can be output
  - dozens of keywords: id, type, x, xs, xu, mux, omegax, ...
  - any output of a compute or fix: f_ID, c_ID[M]
  - atom-style variable: v_foo

Dump output

- Any per-atom quantity can be output
  - dozens of keywords: id, type, x, xs, xu, mux, omegax, ...
  - any output of a compute or fix: f_ID, c_ID[M]
  - atom-style variable: v_foo
- Additional options:
  - control which atoms by group or region
  - control which atoms by threshold
    - dump_modify thresh c_pe $>$ 3.0
  - text or binary or gzipped
  - one big file or per snapshot or per proc
  - see dump_modify fileper or nfile

Dump output

- Any per-atom quantity can be output
  - dozens of keywords: id, type, x, xs, xu, mux, omegax, ...
  - any output of a compute or fix: f_ID, c_ID[M]
  - atom-style variable: v_foo
- Additional options:
  - control which atoms by group or region
  - control which atoms by threshold
    - dump_modify thresh c_pe $>$ 3.0
  - text or binary or gzipped
  - one big file or per snapshot or per proc
  - see dump_modify fileper or nfile
- Post-run conversion
  - tools/python/dump2cfg.py, dump2pdb.py, dump2xyz.py
  - Pizza.py dump, cfg, ensight, pdb, svg, vtk, xyz tools

Classical MD in parallel

- MD is inherently parallel
  - forces on each atom can be computed simultaneously
  - X and V can be updated simultaneously
- Nearly all MD codes are parallelized
  - distributed-memory message-passing (MPI) between nodes
  - MPI or threads (OpenMP, GPU) within node

Classical MD in parallel

- MD is inherently parallel
  - forces on each atom can be computed simultaneously
  - X and V can be updated simultaneously
- Nearly all MD codes are parallelized
  - distributed-memory message-passing (MPI) between nodes
  - MPI or threads (OpenMP, GPU) within node
- MPI = message-passing interface
  - MPICH or OpenMPI
  - assembly-language of parallel computing
  - lowest-common denominator
  - most portable
  - runs on all parallel machines, even on multi- and many-core
  - more scalable than shared-memory parallel

Goals for parallel algorithms

- Scalable
  - short-range MD scales as N
  - optimal parallel scaling is N/P
  - even on clusters with higher communication costs
- Good for short-range forces
  - 80% of CPU
  - long-range Coulombics have short-range component
- Fast for small systems, not just large
  - nano, polymer, bio systems require long timescales
  - 1M steps of 10K atoms is more useful than 10K steps of 1M atoms
- Efficient at finding neighbors
  - liquid state, polymer melts, small-molecule diffusion
  - neighbors change rapidly
  - atoms on a fixed lattice is simpler to parallelize

Parallel algorithms for MD

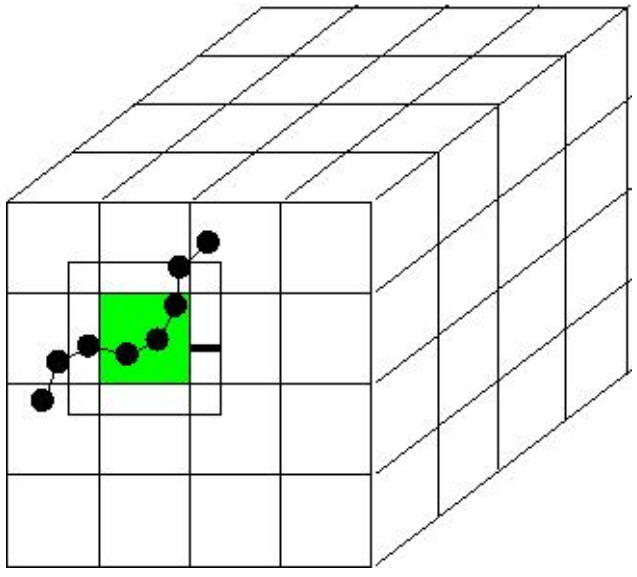
- *Plimpton, J Comp Phys, 117, p 1 (1995)*
- 3 classes of algorithms, used by all MD codes
  1. atom-decomposition = split and replicate atoms
  2. force-decomposition = partition forces
  3. spatial-decomposition = geometric split of simulation box

Parallel algorithms for MD

- *Plimpton, J Comp Phys, 117, p 1 (1995)*
- 3 classes of algorithms, used by all MD codes
  1. atom-decomposition = split and replicate atoms
  2. force-decomposition = partition forces
  3. spatial-decomposition = geometric split of simulation box
- All 3 methods balance computation optimally as N/P
- Differ in organization of inter-particle force computation, other tasks can be done within any of 3 algorithms
  - molecular forces
  - time integration (NVE/NVT/NPT)
  - thermodynamics, diagnostics, ...
- Differ in issues affecting parallel scalability
  - communication costs
  - load-balance

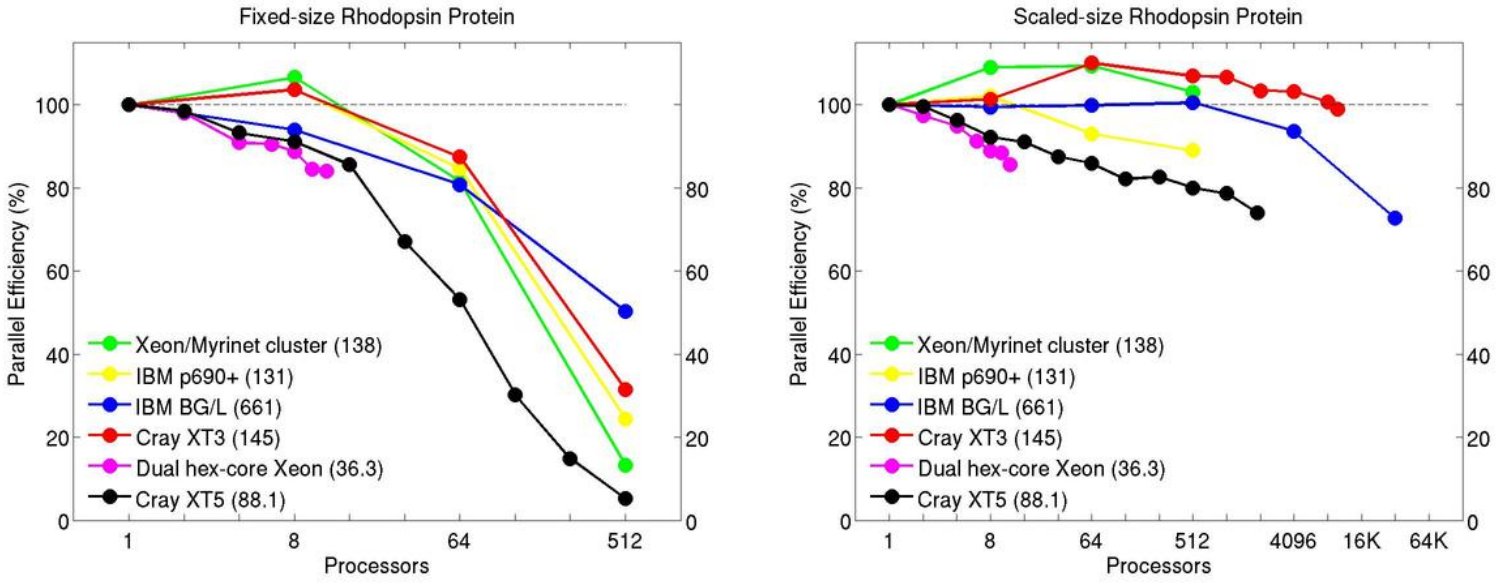
LAMMPS is parallelized via spatial-decomposition

- Physical domain divided into 3d bricks
- One brick per MPI task
- Compute forces on atoms in box
  using ghost info from nearby bricks
- Atoms carry properties &
  topology as they migrate
- Comm of ghost atoms within cutoff
  - 6-way local stencil
- Short-range forces $\Rightarrow$
  CPU cost scales as $O(N/P)$



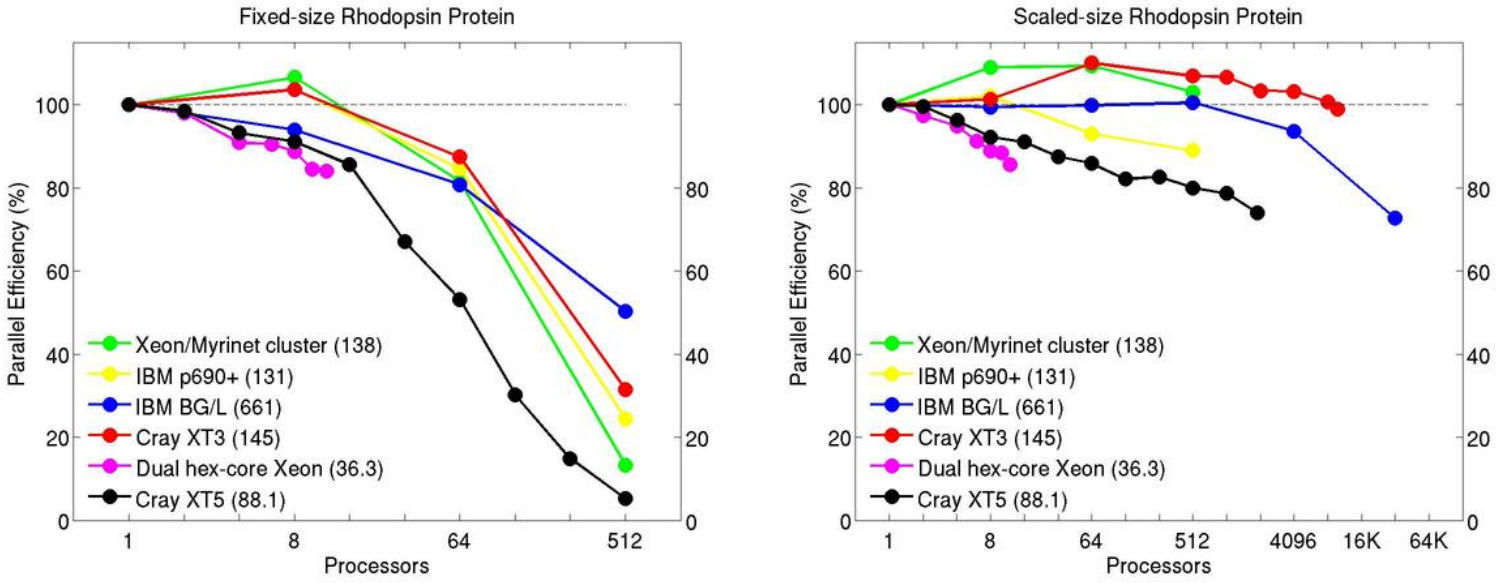
Parallel performance

See http://lammps.sandia.gov/bench.html



Parallel performance

See http://lammps.sandia.gov/bench.html



Useful exercise:

- run bench/in.lj, change N and P, is it $O(N/P)$?
- % mpirun -np 2 lmp_linux < in.lj
- % lmp_linux -v x 2 -v y 2 -v z 2 < in.lj

How to speed-up your simulations

See doc/Section_accelerate.html of manual

1. Many ideas for long-range Coulombics
   - PPPM with 2 vs 4 FFTs
   - PPPM with staggered grid
   - run_style verlet/split command
   - adjust processor layout via processors command

How to speed-up your simulations

2. GPU and USER-CUDA and USER-OMP packages
   - GPU:
     - pair style and neighbor list build on GPU
     - can use multiple cores per GPU
     - 39 supported pair styles, PPPM
   - USER-CUDA:
     - fixes and computes onto GPU (many timesteps)
     - one core per GPU
     - 30 pair styles, 15 fixes, 4 computes, PPPM
   - USER-OMP:
     - threading via OpenMP, run 1 or 2 MPI tasks/node
     - 95 pair styles, 29 fixes, many PPPM variants
   - GPU benchmark data at http://lammps.sandia.gov/bench.html
     - desktop and Titan (ORNL)

How to speed-up your simulations

3. Increase time scale via timestep size
   - fix shake for rigid bonds (2 fs)
   - run_style respa for hierarchical steps (4 fs)

4. Increase length scale via coarse graining
   - all-atom vs united-atom vs bead-spring
   - also increases time scale
   - mesoscale models:
     - ASPHERE, BODY, COLLOID, FLD packages
     - GRANULAR, PERI, RIGID, SRD packages
     - see doc/Section_packages.html for details

Quick tour of more advanced topics

See http://lammps.sandia.gov/features.html

1. Units
   - see doc/units.html
   - LJ, real, metal, cgs, si, micro, nano
   - all input/output in one unit system
2. Ensembles
   - see doc/Section_howto.html 6.16
   - one or more thermostats (by group)
   - single barostat
   - rigid body dynamics (RIGID package)
3. Hybrid models
   - pair_style hybrid and hybrid/overlay
   - atom_style hybrid sphere bond ...

Quick tour of more advanced topics

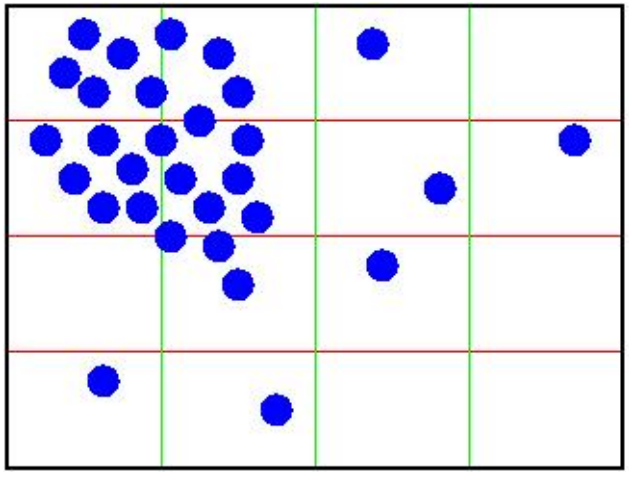
4. Aspherical particles
   - see doc/Section_howto.html 6.14
   - ellipsoidal, lines, triangles, rigid bodies
   - ASPHERE package
5. Mesoscale and continuum models
   - COLLOID, FLD, SRD packages for NPs and colloids
   - PERI package for Peridynamics
   - USER-ATC package for atom-to-continuum (FE)
   - GRANULAR package for granular media
   - add-on LIGGGHTS package for DEM
     - www.liggghts.com and www.cfdem.com

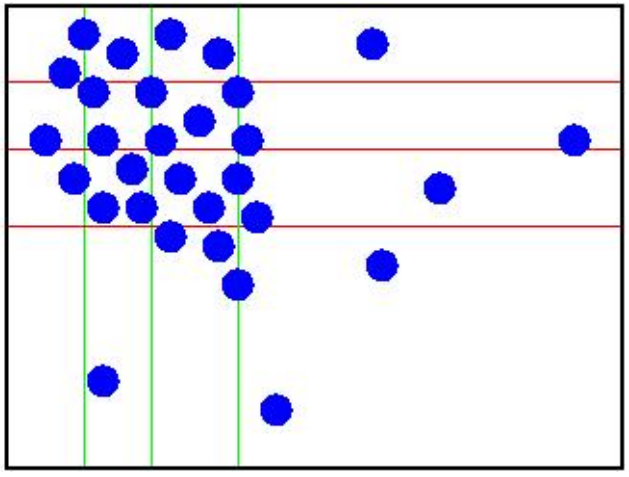
Quick tour of more advanced topics

6. Multi-replica modeling
   - see doc/Section_howto.html 6.14
   - parallel tempering via temper command
   - PRD, TAD, NEB in REPLICA package

Quick tour of more advanced topics

6. Multi-replica modeling
   - see doc/Section_howto.html 6.14
   - parallel tempering via temper command
   - PRD, TAD, NEB in REPLICA package
7. Load balancing
   - balance command for static LB
   - fix balance command for dynamic LB
   - work by adjusting proc dividers in 3d brick grid



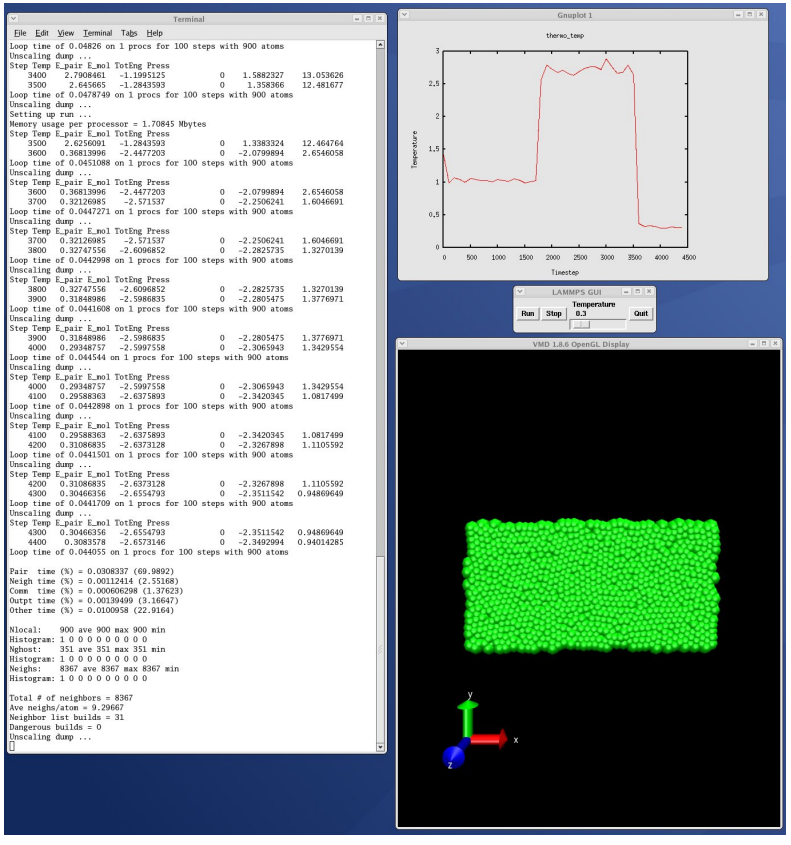


Quick tour of more advanced topics

8. Energy minimization
   - Via dynamics to un-overlap particles
     - pair_style soft with time-dependent push-off
     - fix nve/limit and fix viscous
   - Via gradient-based minimization
     - min_style cg, hftn, sd
   - Via damped-dynamics minimization
     - min_style quickmin and fire
     - used for nudged-elastic band (NEB)

Use LAMMPS as a library or from Python

- doc/Section_howto.html 6.10 and 6.19
- C-style interface (C, C++, Fortran, Python)
- examples/COUPLE dir
- python and python/examples directories



Coupling MD to other scales

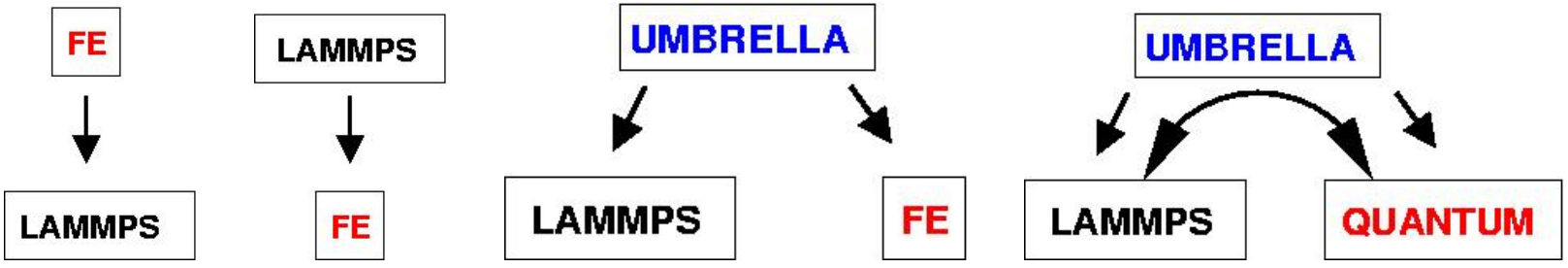
Multi-physics or multi-scale models often lead to
numeric or coupling interface between two methods or two codes

Coupling MD to other scales

Multi-physics or multi-scale models often lead to
numeric or coupling interface between two methods or two codes

- LAMMPS can call other codes as libraries
  - write a simple fix to wrap the library
- Another code can instantiate LAMMPS (one or more times)
  - LAMMPS is really a library (single C++ class)
  - C interface also provided
  - enables LAMMPS to be called from C, Fortran, Python

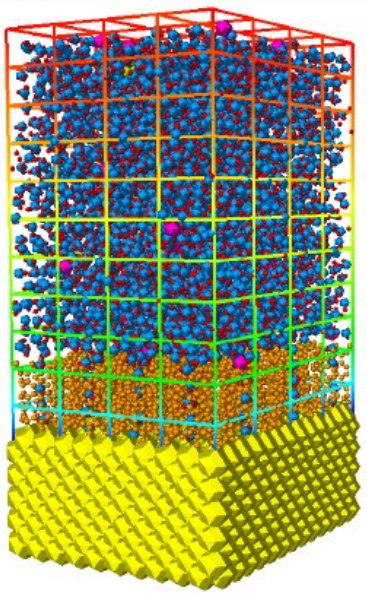
Examples of MD in multi-scale context



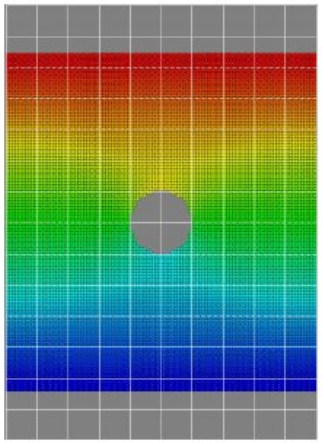
- MD + DFT: dynamics with quantum forces
- MD + on-lattice kinetic MC: stress-driven grain growth
- MD + FE: thermal/mechanical coupling to continuum
- MD + CFD (OpenFoam): fluidized granular bed
- MD + Navier-Stokes: flowing biomolecules

AtC package for atomistic to continuum coupling

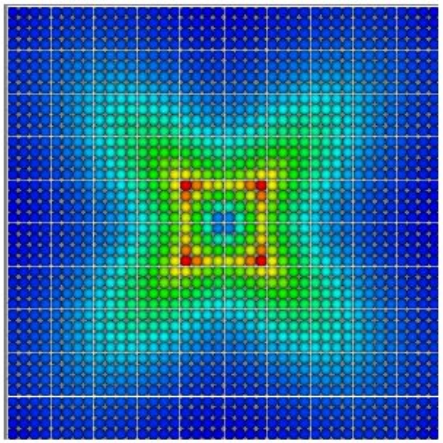
Reese Jones, Jon Zimmerman, Jeremy Templeton,
Greg Wagner (Sandia)



Saltwater-electrode-CNT system: mesh overlaps exactly with water-CNT atom region



Circular hole in plate: mesh overlaps exactly with box, but atom region is subset



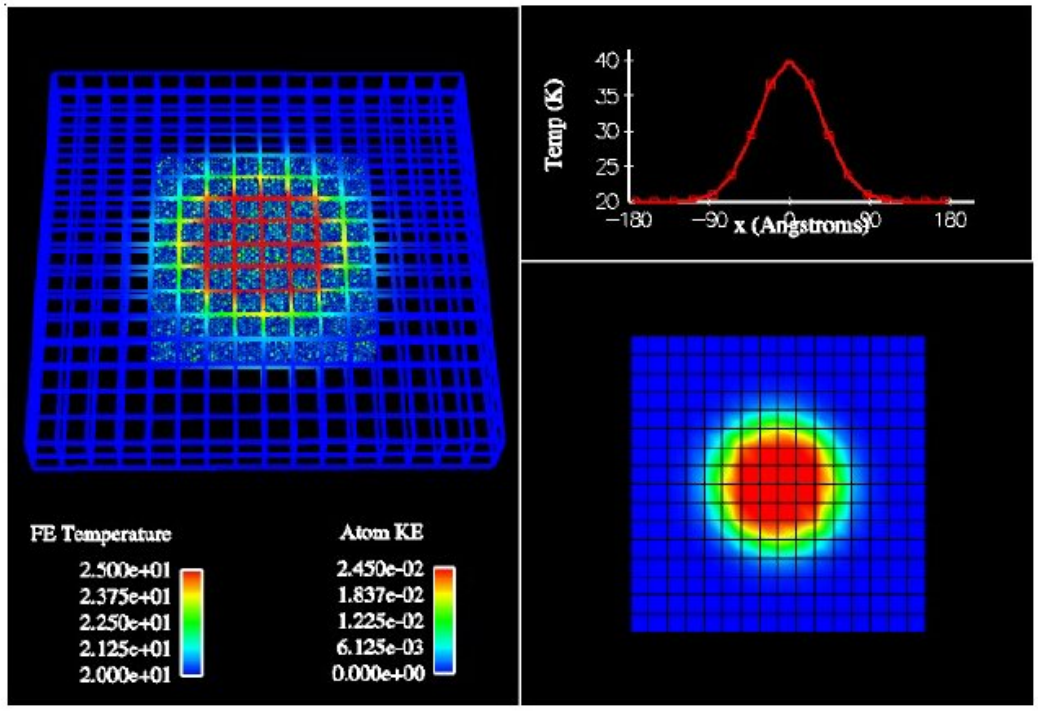
Elastic inclusion problem: mesh overlaps exactly with box and atoms

Particles in parallel, FE solution in serial

Different PDEs can be solved: thermal, deformation, etc

Thermal coupling with AtC package

2D diffusion problem

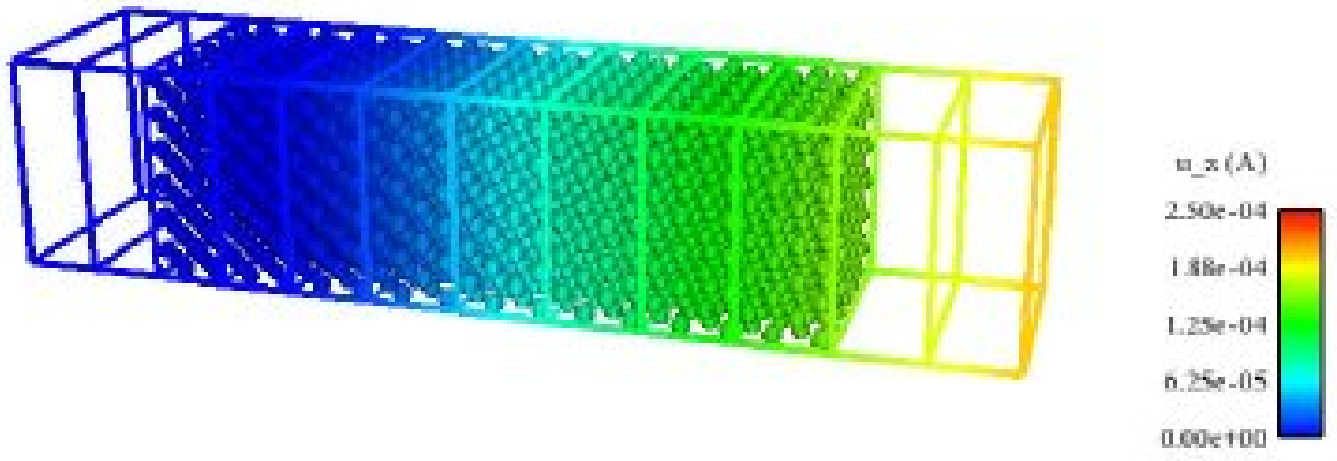


- Plate with embedded MD region (~33,000 atoms)
- Initialized to temperature field with gaussian profile
- Adiabatic boundary conditions at edges

Sandia National Laboratories

Mechanical coupling with AtC package

Elasto-dynamic response:



What have people done with LAMMPS?

- Pictures: http://lammps.sandia.gov/pictures.html
- Movies: http://lammps.sandia.gov/movies.html

	evaporation self-assembly		Compression of nanoparticles
	GCMC model of zeolite occupancy		self-assembling nanofibers from Thiophene-peptide oligomers
	layer-by-layer self assembly		smoothed particle hydrodynamics (SPH) models
	dislocations moving thru grain boundaries		electron force field for non-adiabatic dynamics
	granular particles flowing from hopper		granular Discrete Elememt Method (DEM) models
	fiber dynamics		brazing of two-metal system
	Peridynamics mesoscale modeling of impact fracture		shear faults in a model brittle solid
	stick/slip and polymer flow on rough surfaces		crystallization of polyethylene melt
	melting of polycrystalline metal		deformation and void nucleation under shock loading
	dynamics of an isolated edge dislocation		cavitation in liquid metal
	nanoprecipitates and shock induced plasticity		Brazil nut effect
	ultra-thin Cu nanowire formation		Cu nanowire loading and unloading
	Au nanowire formation and extension		flow of water and ions thru a silica pore
	metal response to He bubble formation		dynamics of rhodopsin protein in lipid membrane
	CO2 escaping from binding pocket of RuBisCO protein		C-terminus of RuBisCO closing over binding pocket
	entropy-driven nano-motor		metal solidification
	liquid crystal conformations		

What have people done with LAMMPS?

- Pictures: http://lammps.sandia.gov/pictures.html
- Movies: http://lammps.sandia.gov/movies.html

	evaporation self-assembly		Compression of nanoparticles
	GCMC model of zeolite occupancy		self-assembling nanofibers from Thiophene-peptide oligomers
	layer-by-layer self assembly		smoothed particle hydrodynamics (SPH) models
	dislocations moving thru grain boundaries		electron force field for non-adiabatic dynamics
	granular particles flowing from hopper		granular Discrete Elememt Method (DEM) models
	fiber dynamics		brazing of two-metal system
	Peridynamics mesoscale modeling of impact fracture		shear faults in a model brittle solid
	stick/slip and polymer flow on rough surfaces		crystallization of polyethylene melt
	melting of polycrystalline metal		deformation and void nucleation under shock loading
	dynamics of an isolated edge dislocation		cavitation in liquid metal
	nanoprecipitates and shock induced plasticity		Brazil nut effect
	ultra-thin Cu nanowire formation		Cu nanowire loading and unloading
	Au nanowire formation and extension		flow of water and ions thru a silica pore
	metal response to He bubble formation		dynamics of rhodopsin protein in lipid membrane
	CO2 escaping from binding pocket of RuBisCO protein		C-terminus of RuBisCO closing over binding pocket
	entropy-driven nano-motor		metal solidification
	liquid crystal conformations		

- Papers: http://lammps.sandia.gov/papers.html
  - authors, titles, abstracts for ∼3600 papers

Modifying LAMMPS (advert for tomorrow)

- LAMMPS is designed to be easy to extend
- 90% of LAMMPS is customized add-on classes, via styles
- Write a new derived class, drop into src, re-compile

Modifying LAMMPS (advert for tomorrow)

- LAMMPS is designed to be easy to extend
- 90% of LAMMPS is customized add-on classes, via styles
- Write a new derived class, drop into src, re-compile

- Tuesday AM - Modifying & Extending LAMMPS
- Wednesday PM - Hands-on: Writing new code for LAMMPS
- Resources:
  - doc/PDF/Developer.pdf
    - class hierarchy & timestep structure
  - doc/Section_modify.html
- Please contribute your code to the LAMMPS distro!

Links and thanks

- LAMMPS: http://lammps.sandia.gov
- post a question: http://lammps.sandia.gov/mail.html
- my email: sjplimp@sandia.gov

- Thanks to LAMMPS developers at Sandia and elsewhere:
  - Aidan Thompson, Paul Crozier
  - Stan Moore, Ray Shan, Christian Trott
  - Axel Kohlmeyer (ICTP & Temple Univ)
  - http://lammps.sandia.gov/authors.html