Classical MD in a nutshell

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

Initial positions and velocities

Interatomic potential

Positions and velocities at later times
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
⇒ atomistic
⇒ coarse-grained
⇒ 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, ...
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 ⇒ atomistic ⇒ coarse-grained ⇒ 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
  - 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)
LAMMPS tries hard to flag many kinds of errors and warnings

1. If an input command generates the error ...  
   - % Imp_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
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 0.0004177 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])*${scale}$

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
Defining variables in input scripts

**Styles:** index, loop, equal, atom, ...
- variable x index run1 run2 run3 run4
- variable x loop 100
- variable x equal \( \text{trap}(f_{\text{JJ}[3]})*\{\text{scale}\} \)

**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 ${param}`
- **Various ways to run multiple simulations from one script**
  - See Section HOWTO 6.4 of manual
variable r equal random(1,1000000000,58798)
variable a loop 8
variable t index 0.8 0.85 0.9 0.95 1.0 1.05 1.1 1.15
log log.$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 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, Avogadro, PackMol, Moltemplate
  - Votca for CG force-field generation
- Monte Carlo builders and force-field assignment
  - Towhee (configurational bias) & others
- Be willing to write system-building scripts yourself
Moltemplate

- [http://www.moltemplate.org](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](doc/Section_commands.html) for full list

<table>
<thead>
<tr>
<th>none</th>
<th>hybrid</th>
<th>hybrid/overlay</th>
<th>adp</th>
</tr>
</thead>
<tbody>
<tr>
<td>airko</td>
<td>berth</td>
<td>berth</td>
<td>step</td>
</tr>
<tr>
<td>born</td>
<td>born/coul/long</td>
<td>born/coul/msm</td>
<td>born/coul/wolf</td>
</tr>
<tr>
<td>brownian</td>
<td>brownian/poly</td>
<td>buck/long/coul/long</td>
<td>colloid</td>
</tr>
<tr>
<td>buck/coul/long</td>
<td>buck/coul/msm</td>
<td>buck/long/coul/long</td>
<td>colloid</td>
</tr>
<tr>
<td>comb</td>
<td>coul/cut</td>
<td>coul/debye</td>
<td>coul/dsf</td>
</tr>
<tr>
<td>coul/long</td>
<td>coul/msm</td>
<td>coul/wolf</td>
<td>dpd</td>
</tr>
<tr>
<td>dpd/kstat</td>
<td>dsnc</td>
<td>eam</td>
<td>eam/alloy</td>
</tr>
<tr>
<td>eam/fs</td>
<td>eim</td>
<td>gauss</td>
<td>gyberne</td>
</tr>
<tr>
<td>gran/hertz/history</td>
<td>gran/hooke</td>
<td>gran/hooke/history</td>
<td>hbond/dreiding/ij</td>
</tr>
<tr>
<td>hbond/dreiding/morse</td>
<td>kim</td>
<td>lcbop</td>
<td>line/ij</td>
</tr>
<tr>
<td>l/j/charmm/coul/charmm</td>
<td>l/j/charmm/coul/charmm/implicit</td>
<td>l/j/charmm/coul/long</td>
<td>l/j/charmm/coul/msm</td>
</tr>
<tr>
<td>l/j/class2</td>
<td>l/j/class2/coul/cut</td>
<td>l/j/class2/coul/long</td>
<td>l/j/class2/coul/long</td>
</tr>
<tr>
<td>l/j/cut/coul/cut</td>
<td>l/j/cut/coul/debye</td>
<td>l/j/cut/coul/dsf</td>
<td>l/j/cut/coul/long</td>
</tr>
<tr>
<td>l/j/cut/tip4p/long</td>
<td>l/j/expand</td>
<td>l/j/gromacs</td>
<td>l/j/gromacs/coul/gromacs</td>
</tr>
<tr>
<td>l/j/long/coul/long</td>
<td>l/j/long/dipole/long</td>
<td>l/j/long/tip4p/long</td>
<td>l/j/smooth</td>
</tr>
<tr>
<td>l/j/smooth/linear</td>
<td>l/j/lub</td>
<td>lubricate</td>
<td>lubricate/poly</td>
</tr>
<tr>
<td>lubricateU</td>
<td>lubricateU/poly</td>
<td>mean</td>
<td>me/cut</td>
</tr>
<tr>
<td>morse</td>
<td>per/ps</td>
<td>per/psb</td>
<td>per/ves</td>
</tr>
<tr>
<td>reax</td>
<td>rebo</td>
<td>resquared</td>
<td>soft</td>
</tr>
<tr>
<td>sw</td>
<td>table</td>
<td>tersoff</td>
<td>tersoff/mod</td>
</tr>
<tr>
<td>tersoff/zbl</td>
<td>tip4p/cut</td>
<td>tip4p/long</td>
<td>tri/ij</td>
</tr>
<tr>
<td>yukawa</td>
<td>yukawa/colloid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Pair styles</th>
<th>Pair styles</th>
<th>Pair styles</th>
<th>Pair styles</th>
</tr>
</thead>
<tbody>
<tr>
<td>adp/omp</td>
<td>altreco/omp</td>
<td>beck/omp</td>
<td>born/coul/long/cuda</td>
</tr>
<tr>
<td>born/coul/omp/gpu</td>
<td>born/coul/omp</td>
<td>born/coul/long/omp</td>
<td>born/coul/long/cuda</td>
</tr>
<tr>
<td>born/coul/wolf/omp</td>
<td>born/gpu</td>
<td>born/omp</td>
<td>brownian/omp</td>
</tr>
<tr>
<td>brownian/poly/omp</td>
<td>back/coul/cut/cuda</td>
<td>buck/coul/cut/gpu</td>
<td>buck/coul/cut/omp</td>
</tr>
<tr>
<td>buck/coul/long/cuda</td>
<td>back/buck/long/omp</td>
<td>buck/gpu</td>
<td>buck/coul/long/omp</td>
</tr>
<tr>
<td>cache/gpu</td>
<td>collision/omp</td>
<td>comb/omp</td>
<td>color/cut/omp</td>
</tr>
<tr>
<td>coul/debye/omp</td>
<td>coul/dsf/omp</td>
<td>coul/long/omp</td>
<td>coul/long/omp</td>
</tr>
<tr>
<td>coul/membr/omp</td>
<td>coul/wolf</td>
<td>dpd/omp</td>
<td>dpd/stat/omp</td>
</tr>
<tr>
<td>coul/long/cuda</td>
<td>eam/cuda</td>
<td>eam/long/omp</td>
<td>eam/long/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/cuda</td>
<td>eam/long/omp</td>
<td>eam/long/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/cuda</td>
<td>eam/fs/cuda</td>
<td>eam/fs/cuda</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/fs/cuda</td>
<td>eam/fs/cuda</td>
<td>eam/fs/cuda</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/fs/cuda</td>
<td>eam/gpu</td>
<td>eam/gpu</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/gpu</td>
<td>eam/omp</td>
<td>eam/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
<tr>
<td>coul/polar/omp</td>
<td>eam/omp</td>
<td>gauss/omp</td>
<td>gauss/omp</td>
</tr>
</tbody>
</table>
| coul/polar/omp       | eam/omp              | gauss/omp            | ga...
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 airho` - AIREBO potential of Stuart
- `pair_style beck` - Beck potential
- `pair_style body` - interactions between body particles
- `pair_style hop` - 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 thermostating
- `pair_style dsmc` - Direct Simulation Monte Carlo (DSMC)
- `pair_style eam` - embedded atom method (EAM)
- `pair_style eam/ialloy` - 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 lammmps.sandia.gov/bench.html#potentials

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

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

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

- **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)
  - $R_{\text{neigh}} = R_{\text{force}} + \Delta_{\text{skin}}$
  - 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):
  - grid domain: bins of size $R_{\text{force}}$
  - each step: search 27 bins for neighbors (or 14 bins)
Verlet list is $\sim 6 \times$ savings over bins

- $V_{\text{sphere}} = \frac{4}{3} \pi r^3$
- $V_{\text{cube}} = 27 \ r^3$

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

<table>
<thead>
<tr>
<th>none</th>
<th>hybrid</th>
<th>class2</th>
<th>fene</th>
</tr>
</thead>
<tbody>
<tr>
<td>fene/expand</td>
<td>harmonic</td>
<td>morse</td>
<td>nonlinear</td>
</tr>
<tr>
<td>quartic</td>
<td>table</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- used if LAMMPS is built with the **appropriate accelerated package**.

| class2/omp | fene/omp | fene/expand/omp | harmonic/omp |
| harmonic/shift | harmonic/shift | 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
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)
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
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)**
- 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 ⇒ 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](http://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 $\sim 2x$ 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
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
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
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?
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
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

<table>
<thead>
<tr>
<th>angle/local</th>
<th>atom/molecule</th>
<th>body/local</th>
<th>bond/local</th>
<th>centro/atom</th>
<th>cluster/atom</th>
</tr>
</thead>
<tbody>
<tr>
<td>cna/atom</td>
<td>com</td>
<td>com/molecule</td>
<td>contact/atom</td>
<td>coord/atom</td>
<td>damage/atom</td>
</tr>
<tr>
<td>dihedral/local</td>
<td>displace/atom</td>
<td>erotate/asphere</td>
<td>erotate/sphere</td>
<td>erotate/sphere/atom</td>
<td>event/displace</td>
</tr>
<tr>
<td>group/group</td>
<td>gyration</td>
<td>gyration/molecule</td>
<td>heat/flux</td>
<td>improper/local</td>
<td>inertia/molecule</td>
</tr>
<tr>
<td>ke</td>
<td>ke/atom</td>
<td>msd</td>
<td>msd/molecule</td>
<td>msd/nongauss</td>
<td>pair</td>
</tr>
<tr>
<td>pair/local</td>
<td>pressure</td>
<td>property/atom</td>
<td>property/local</td>
<td>slice</td>
<td>stress/atom</td>
</tr>
<tr>
<td>property/molecule</td>
<td>rdf</td>
<td>reduce</td>
<td>reduce/region</td>
<td>temp/normal</td>
<td>temp/def</td>
</tr>
<tr>
<td>temp</td>
<td>temp/asphere</td>
<td>temp/com</td>
<td>temp/deform</td>
<td>temp/normal</td>
<td>temp/def</td>
</tr>
<tr>
<td>temp/ramp</td>
<td>temp/region</td>
<td>temp/sphere</td>
<td>ti</td>
<td>voronoi/atom</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ackland/atom</th>
<th>basal/atom</th>
<th>ke/eff</th>
<th>ke/atom/eff</th>
<th>meso_e/atom</th>
<th>meso_rho/atom</th>
</tr>
</thead>
<tbody>
<tr>
<td>meso_t/atom</td>
<td>temp/eff</td>
<td>temp/deform/eff</td>
<td>temp/region/eff</td>
<td>temp/rotate</td>
<td></td>
</tr>
</tbody>
</table>

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

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
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
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, ppy, 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
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
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`
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 ⇒ CPU cost scales as $O(N/P)$
Parallel performance

Parallel performance


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

Many ideas for long-range Coulombics

1. PPPM with 2 vs 4 FFTs
2. PPPM with staggered grid
3. run_style verlet/split command
4. 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**
     - 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/Sectionhowto.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)

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
Mechanical coupling with AtC package

Elasto-dynamic response:
What have people done with LAMMPS?

- **Pictures:** [http://lammps.sandia.gov/pictures.html](http://lammps.sandia.gov/pictures.html)
- **Movies:** [http://lammps.sandia.gov/movies.html](http://lammps.sandia.gov/movies.html)

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

- **Pictures:** [http://lammps.sandia.gov/pictures.html](http://lammps.sandia.gov/pictures.html)
- **Movies:** [http://lammps.sandia.gov/movies.html](http://lammps.sandia.gov/movies.html)

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

- **Papers:** [http://lammps.sandia.gov/papers.html](http://lammps.sandia.gov/papers.html)
  - authors, titles, abstracts for ~3600 papers
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

- 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