A brief survey of the LAMMPS particle simulation code: introduction, case studies, and future development

Paul S. Crozier

Sandia National Laboratories

March 28, 2014

$(SICM)^2$ Parallel Computing Workshop

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.

LOCKHEED MARTIN

Sandia National Laboratories

Acknowledgements

Steve Plimpton

Aidan Thompson

Christian Trott

(all from Sandia)

Many other contributors to LAMMPS

(see: http://lammps.sandia.gov/authors.html)

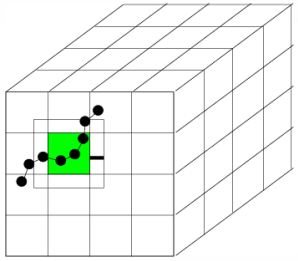
Sandia National Laboratories

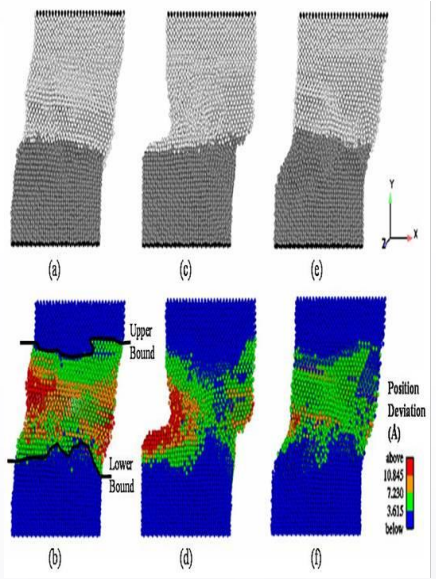
What is LAMMPS?

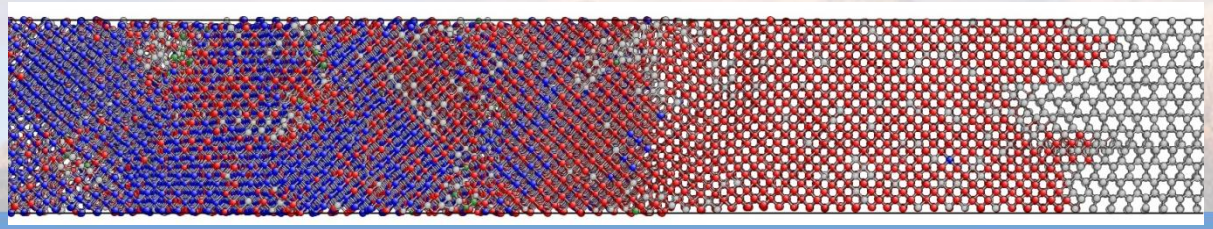
(Large-scale Atomic/Molecular Massively Parallel Simulator)

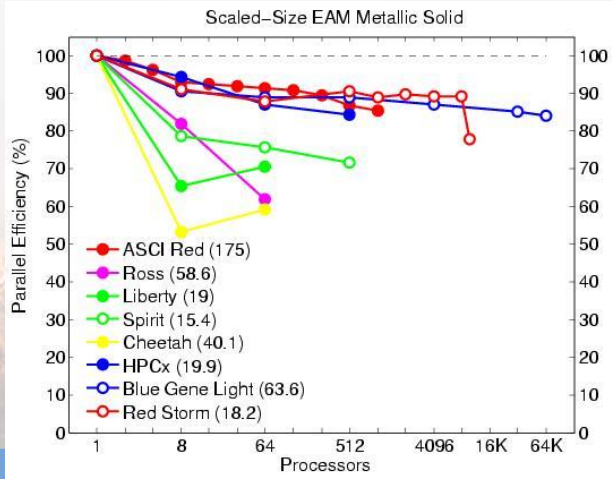
http://lammps.sandia.gov

- Classical MD code.
- Open source, highly portable C++.
- Freely available for download under GPL.
- Easy to download, install, and run.
- Well documented.
- Easy to modify or extend with new features and functionality.
- Active user's e-mail list with over 2000 subscribers.
- Users' workshops: Feb 2010, Aug 2011, Aug 2013, March 2014.
- Spatial-decomposition of simulation domain for parallelism.
- Energy minimization via conjugate-gradient relaxation.
- Atomistic, mesoscale, and coarse-grain simulations.
- Variety of potentials (including many-body and coarse-grain).
- Variety of boundary conditions, constraints, etc.









Freely Available Parallel MD Codes

- **CHARMM, AMBER:** grand-daddies of MD codes, lots of bio features
- **NAMD:** bio, clever decomposition, very scalable
- **GROMACS:** bio, fastest single processor performance, now scalable
- **DL-POLY:** materials oriented
- **HOOMD:** GPU-based code, fastest on single GPUs

- **LAMMPS**
  - materials oriented, wide range of interatomic potentials
  - many coarse-grained models for mesoscale to continuum
  - scalable for large simulations (1000s of particles/processor)
  - easy to extend

LAMMPS Impact

- Est. 1995
- 75,000 downloads
- 1400 citations
- 20,000 postings to mail list
- 50K → 300K lines of code
- 100 external contributors
- 2-day symposium at TMS in 2011 honoring LAMMPS
- Benchmark code for ORNL, LLNL, DoD, universities

Why Use LAMMPS?

Answer 1: Good parallel performance

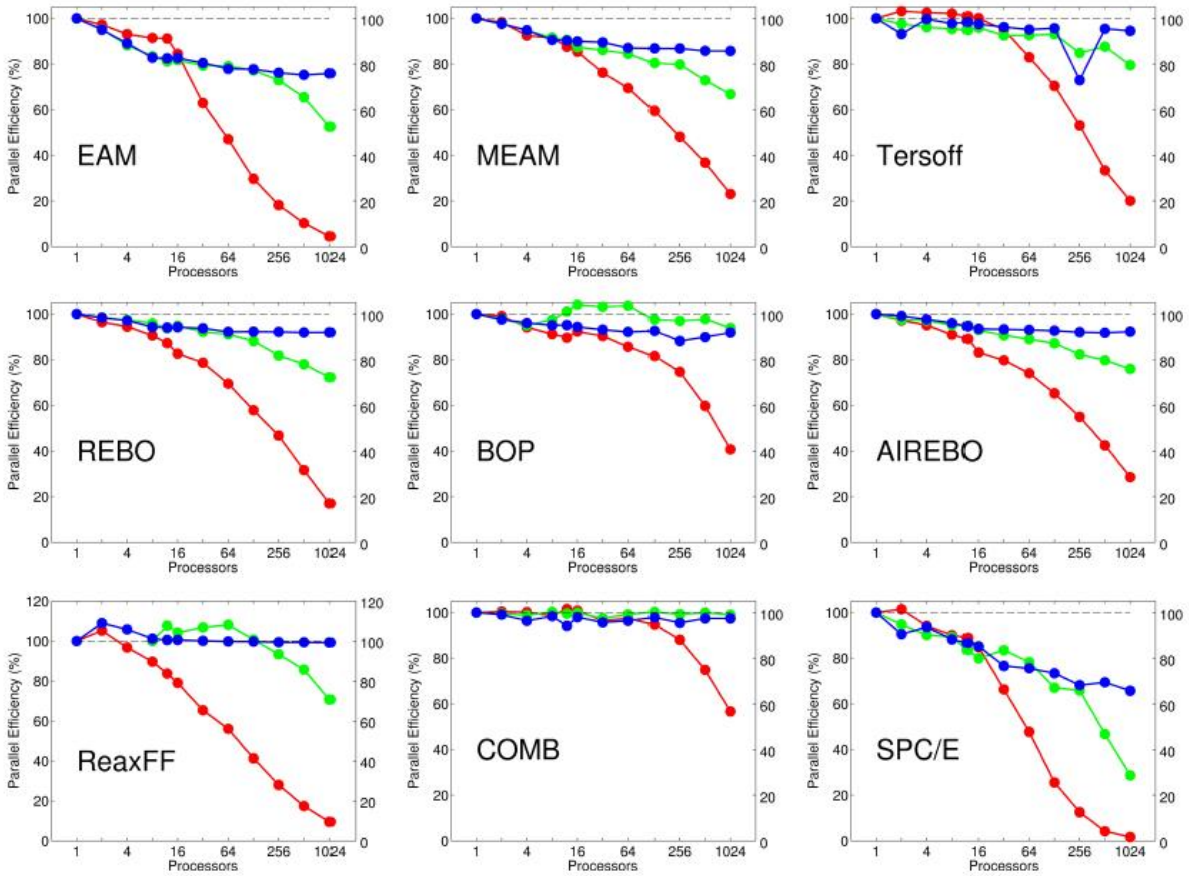


Figure 1: *Performance of 8 many-body potentials and an SPC/E water potential on varying numbers of cores of a Cray XT5 machine, as implemented in LAMMPS. For each potential, efficiency is defined as the one-processor timing divided by the P-processor timing, multiplied by 100/P. The red curves are for 32K atom systems, the green curves are for 1M atom systems; the blue curves are for scaled systems with 32K atoms per processor. The single-core CPU times per-atom per-timestep are listed in Table I.*

Red = 32k atoms
small fixed-size
strong scaling

Green = 1M atoms
large fixed-size
strong scaling

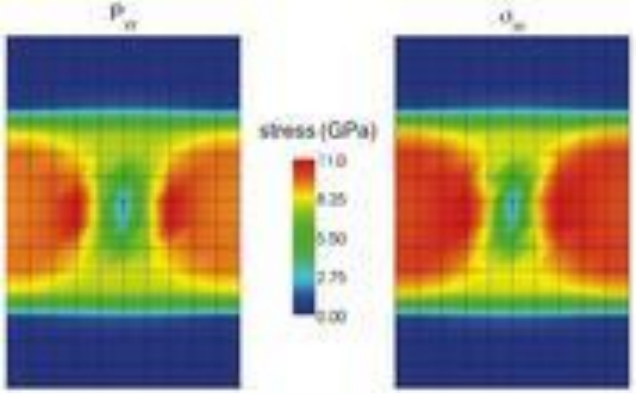
Blue = 32k atoms/proc
scaled size
weak scaling

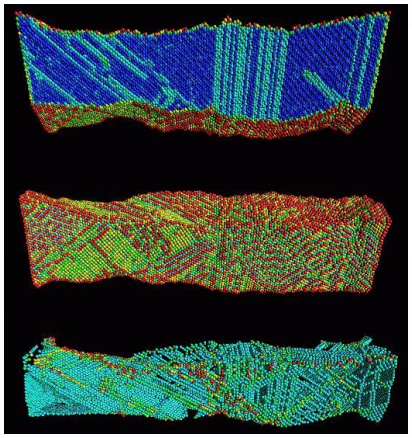
MRS Bulletin,
May 2012,
37, 513-521.

Why Use LAMMPS?

Answer 2: Versatility

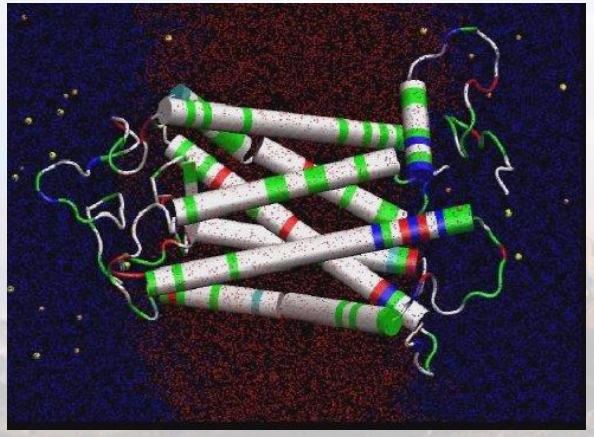
Solid Mechanics

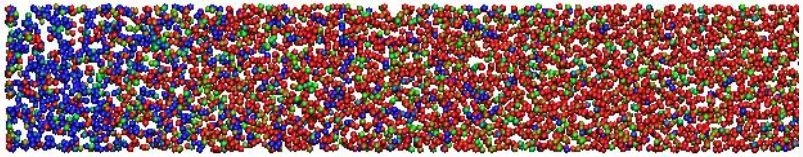




Materials Science

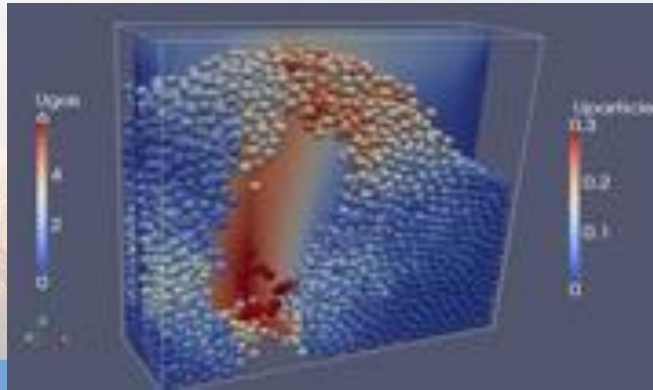
Biophysics





Chemistry

Granular Flow



Why Use LAMMPS?

Answer 3: Modularity

LAMMPS Objects

atom styles: atom, charge, colloid, ellipsoid, point dipole

pair styles: LJ, Coulomb, Tersoff, ReaxFF, AI-REBO, COMB, MEAM, EAM, Stillinger-Weber,

fix styles: NVE dynamics, Nose-Hoover, Berendsen, Langevin, SLLOD, Indentation,...

compute styles: temperatures, pressures, per-atom energy, pair correlation function, mean square displacements, spatial and time averages

Goal: All computes works with all fixes work with all pair styles work with all atom styles

Why Use LAMMPS?

Answer 4: Potential Coverage

LAMMPS Potentials

Biomolecules: CHARMM, AMBER, OPLS, COMPASS (class 2), long-range Coulombics via PPPM, point dipoles, ...

Polymers: all-atom, united-atom, coarse-grain (bead-spring FENE), bond-breaking, …

Materials: EAM and MEAM for metals, Buckingham, Morse, Yukawa, Stillinger-Weber, Tersoff, AI-REBO, ReaxFF, COMB, eFF...

Mesoscale: granular, DPD, Gay-Berne, colloidal, peri-dynamics, DSMC...

Hybrid: can use combinations of potentials for hybrid systems: water on metal, polymers/semiconductor interface, colloids in solution, …

Why Use LAMMPS?

Answer 4: Potential Coverage (cont.)

LAMMPS Potentials

pairwise potentials: Lennard-Jones, Buckingham, ...
charged pairwise potentials: Coulombic, point-dipole
manybody potentials: EAM, Finnis/Sinclair, modified EAM (MEAM), embedded ion method (EIM), Stillinger-Weber, Tersoff, AI-REBO, ReaxFF, COMB
electron force field (eFF)
coarse-grained potentials: DPD, GayBerne, ...
mesoscopic potentials: granular, peridynamics
long-range Coulombics and dispersion: Ewald, PPPM (similar to particle-mesh Ewald)

Why Use LAMMPS?

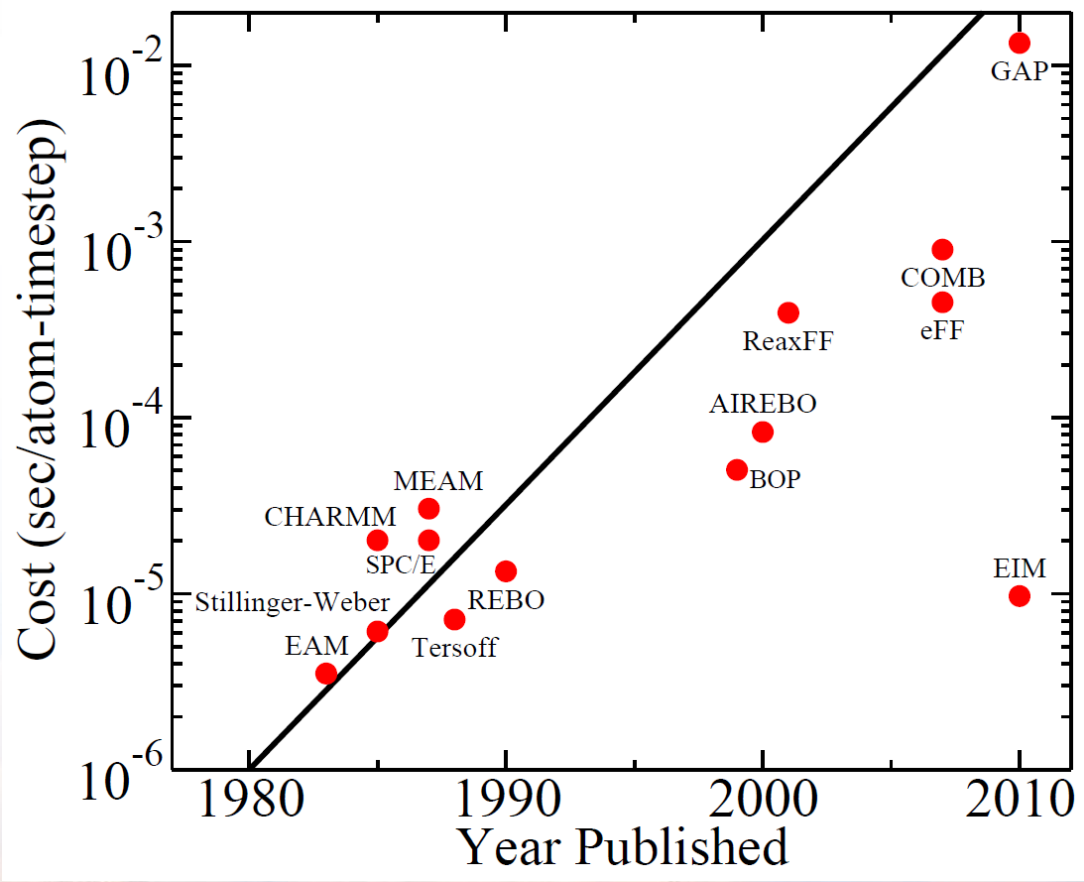
Answer 4: Potential Coverage (cont.)

LAMMPS Potentials (contd.)

bond potentials: harmonic, FENE,...
angle potentials: harmonic, CHARMM, ...
dihedral potentials: harmonic, CHARMM,...
improper potentials: harmonic, cvff, class 2 (COMPASS)
polymer potentials: all-atom, united-atom, bead-spring, breakable
water potentials: TIP3P, TIP4P, SPC
implicit solvent potentials: hydrodynamic lubrication, Debye
force-field compatibility with common CHARMM, AMBER, OPLS, GROMACS options

More Accurate & Expensive Potentials

Potential	System	LJ Ratio
Granular	chute flow	0.34x
FENE bead/spring	polymer melt	0.36x
Lennard-Jones	LJ liquid	1.0x
DPD	pure solvent	1.46x
EAM	bulk Cu	2.4x
Tersoff	bulk Si	4.1x
Stillinger-Weber	bulk Si	4.1x
EIM	crystalline NaCl	6.5x
SPC/E	liquid water	9.7x
CHARMM + PPPM	solvated protein	13.6x
MEAM	bulk Ni	15.6x
Peridynamics	glass fracture	16.4x
Gay-Berne	ellipsoid mixture	28.3x
AIREBO	polyethylene	54.7x
COMB	crystalline SiO2	284x
eFF	H plasma	306x
ReaxFF	PETN crystal	337x
VASP/small	water	17.7e6
VASP/medium	CO2	170e6
VASP/large	Xe	908e6



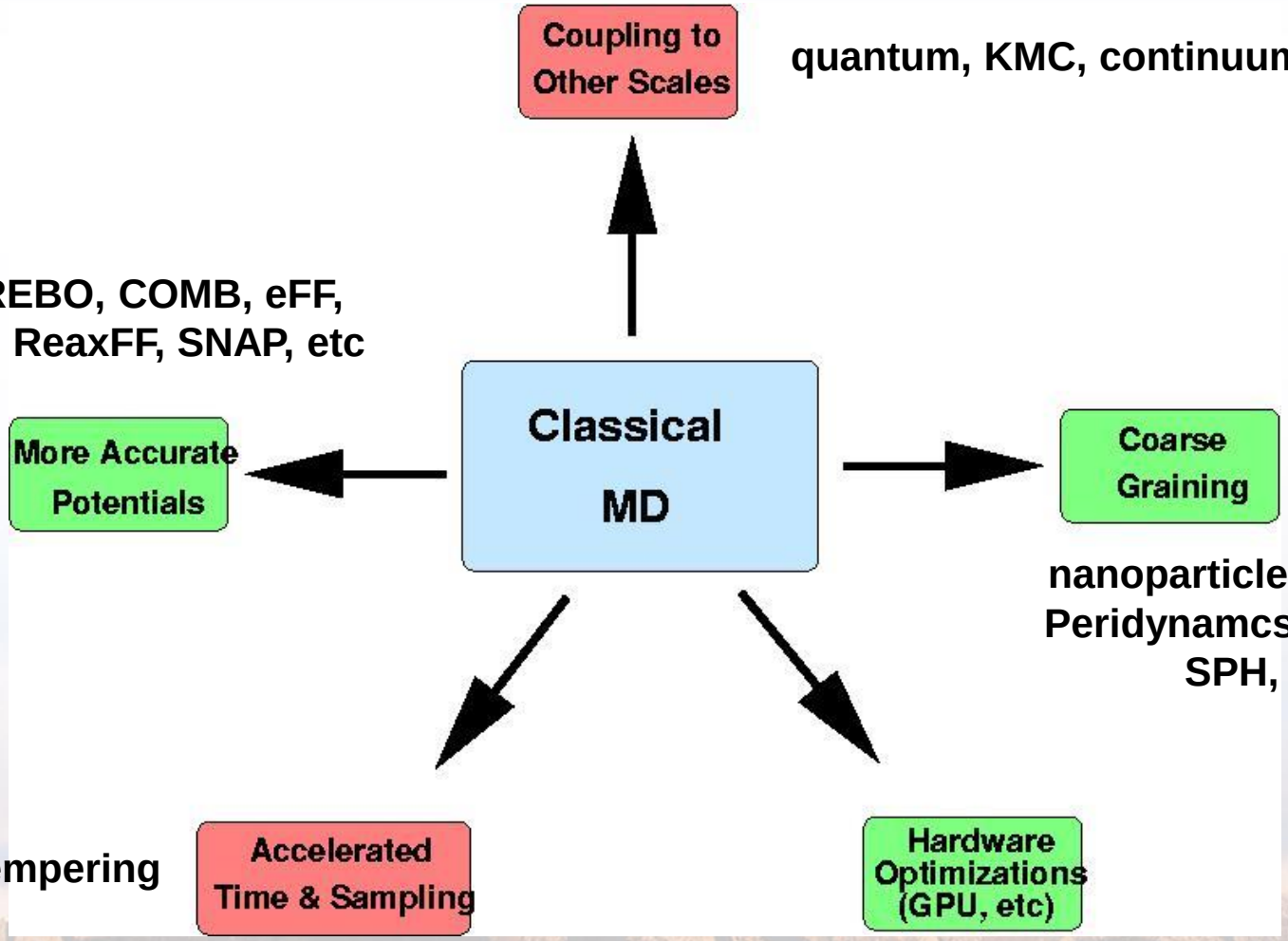
Classical potentials scale as ~O(N)
Quantum scales as ~O(N^3)

Why Use LAMMPS?

Answer 5: Easily extensible

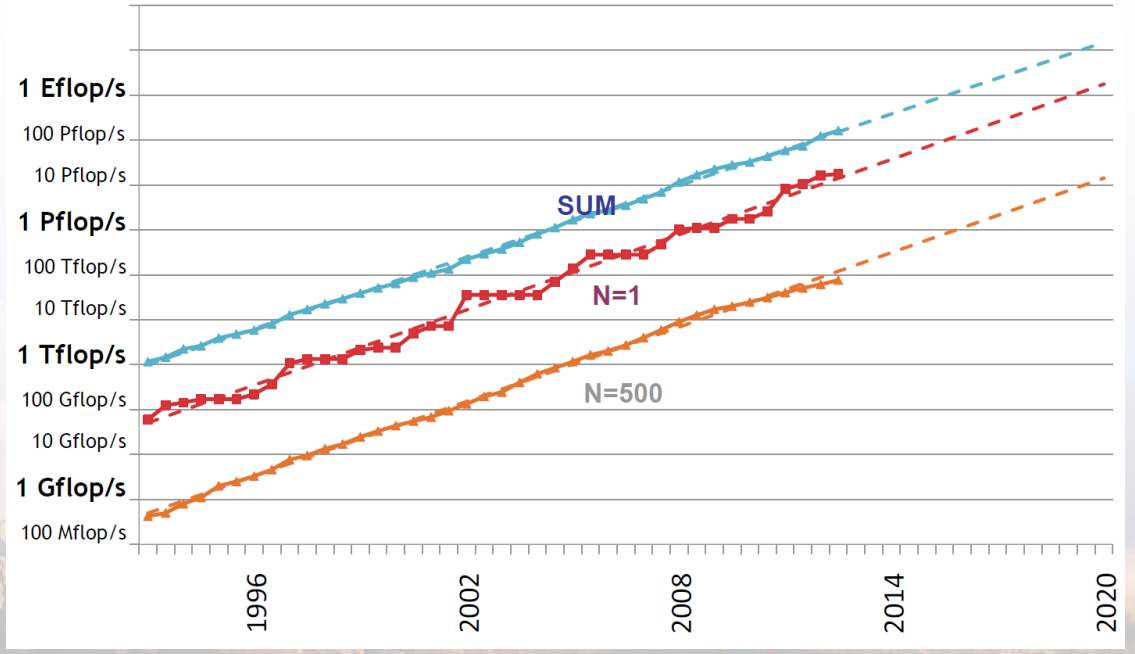
- **One of the best features of LAMMPS**
  - 80% of code is “extensions” via styles
  - only ~35K of 300K lines is core of LAMMPS
- **Easy to add new features via 14 “styles”**
  - new particle types = atom style
  - new force fields = pair style, bond style, angle style, dihedral style, improper style
  - new long range = kspace style
  - new minimizer = min style
  - new geometric region = region style
  - new output = dump style
  - new integrator = integrate style
  - new computations = compute style (global, per-atom, local)
  - new fix = fix style = BC, constraint, time integration, ...
  - new input command = command style = read_data, velocity, run, …
- **Enabled by C++**
  - virtual parent class for all styles, e.g. pair potentials
  - defines interface the feature must provide
  - compute(), init(), coeff(), restart(), etc

LAMMPS Research Directions



Looking ahead to Exascale

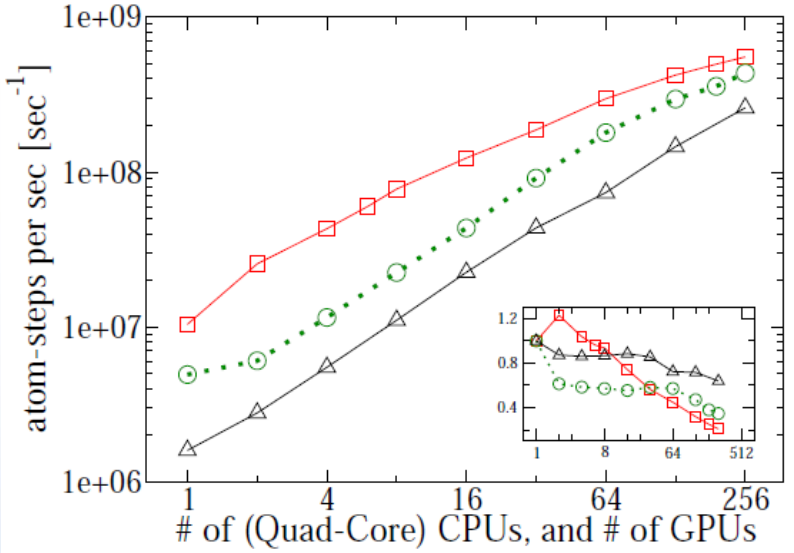
- Next-generation hardware will be heterogeneous and many-core.
- MPI everywhere will be inadequate for good performance.
- Legacy codes (including LAMMPS) will need to be reformulated to expose more parallelism.
- Want software to be portable and efficient across a variety of heterogeneous hardware configurations.

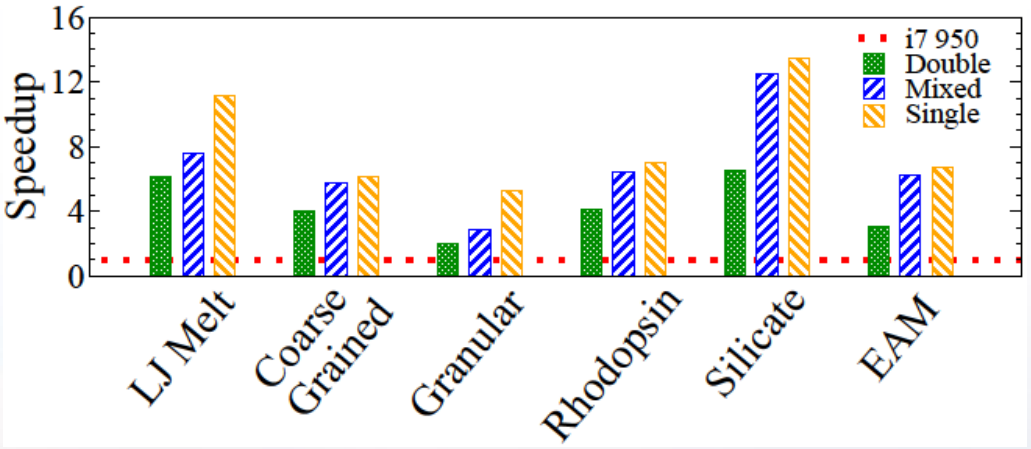


Data from: TOP500 November 2012

Looking ahead to Exascale (cont.)

- Efforts for multi-core, multi-GPU, and hybrid nodes
- Kernels: pair interactions, neighbor list builds, long-range



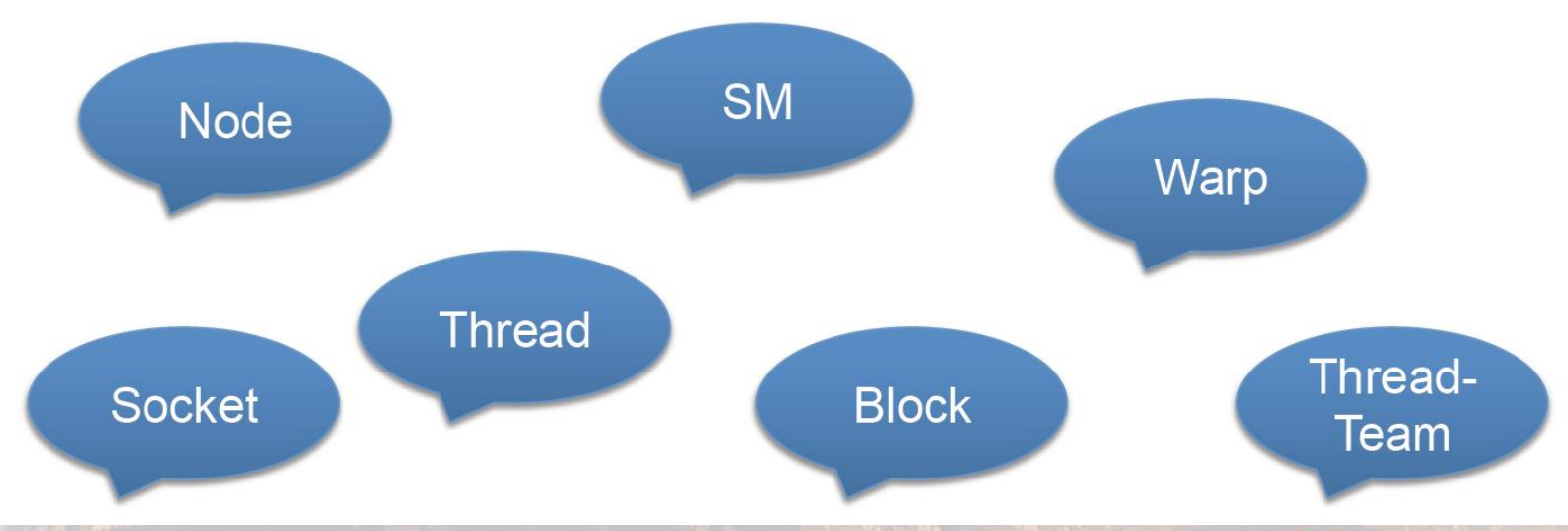


- Challenge to cover wide range of options in LAMMPS
- Collaborations with ORNL, U Tech Ilmenau, NVIDIA

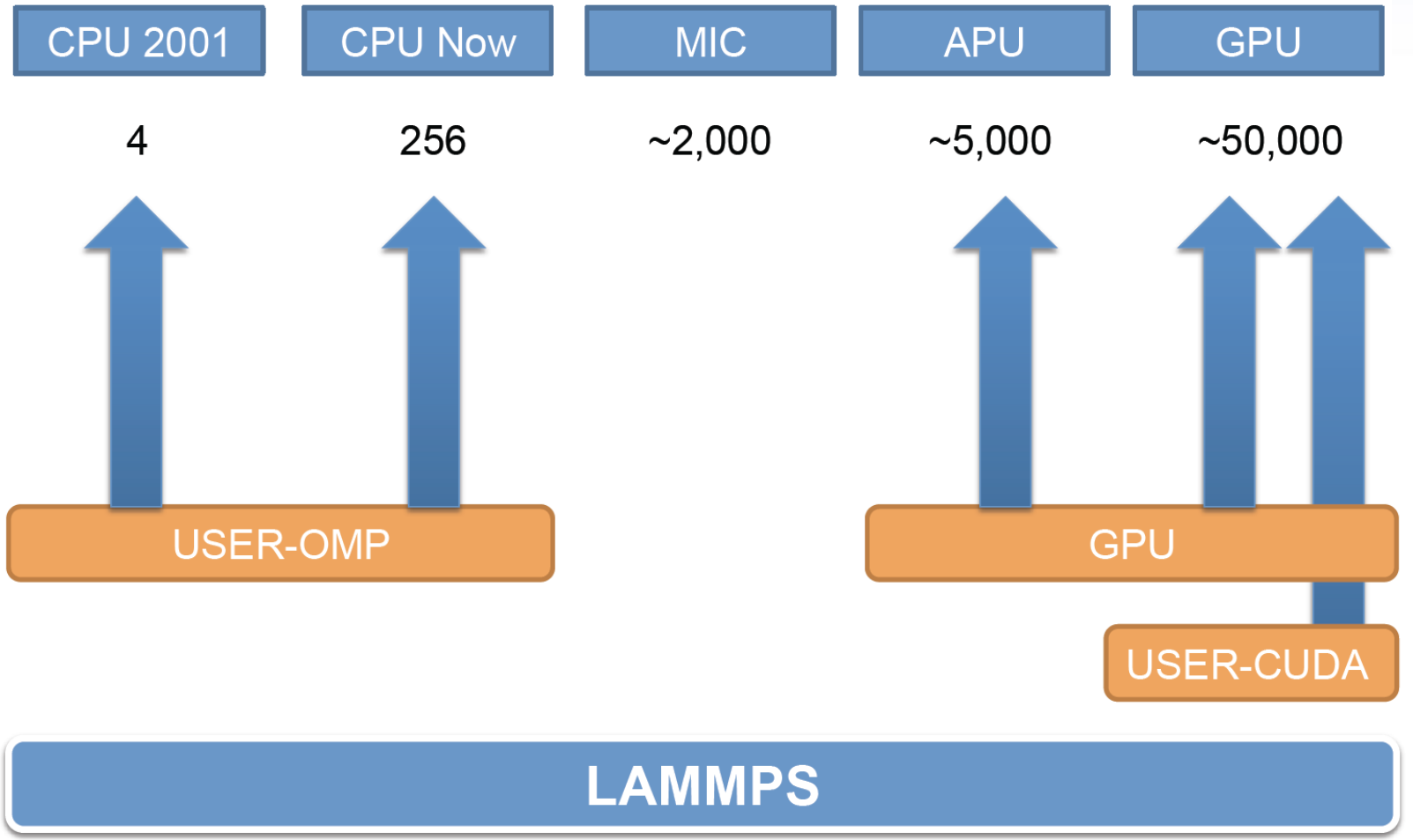
The node parallelism challenge

CPU 2001	CPU Now	MIC	APU	GPU
4	256	~2,000	~5,000	~50,000

MPI everywhere will be inadequate going forward.
We'll need threading.



The node parallelism challenge



Current threading support in LAMMPS

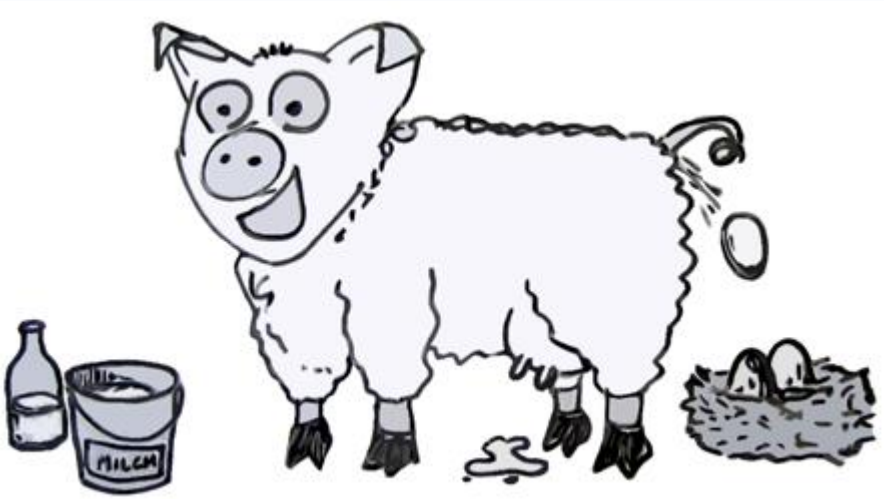
- **LAMMPS threading through optional packages:**
  - USER-OMP, for multicore CPUs
  - USER-CUDA, for Nvidia GPUs via CUDA
  - GPU, for all GPUs via CUDA or OpenCL

- **Advantages of achieving threading through optional packages**
  - Added flexibility --- each optional package can go its own way
  - Users who don't want that flavor of threading don't have to care

- **Disadvantages of achieving threading through optional packages**
  - Optional packages can get out of sync with main LAMMPS code
  - Lots of code repetition
  - Divergent code
  - Maintenance nightmare!

Wish list for LAMMPS threading support

- Single, unified, simple, maintainable code base
- Support for all current and next-gen hardware
- Flexible run configurations
  - MPI-only
  - MPI + threads
  - MPI + GPU
  - MPI + GPU + threads
- Close to optimal performance for all possible run configurations.
- Allow kernel specialization.
- Use vendor-specific compilers.



Eierlegende Wollmilchsau
(egg-laying wool-milk-sow)
http://en.wiktionary.org/wiki/eierlegende_Wollmilchsau

What is Kokkos?

- C++ template library → almost everything is headers.
- Open source, stand alone (no dependencies required).
- Lead developer: Carter Edwards (Sandia).
- First stable release in September 2013.
- Currently being integrated into Trilinos.
- Developed as node level parallelism layer for Trilinos
  - See: http://trilinos.sandia.gov/packages/kokkos
  - Kokkos information: http://trilinos.sandia.gov/packages/docs/r8.0/packages/kokkos/doc/html/index.html
  - "Kokkos: Enabling performance portability across manycore architectures", Carter Edwards and Christian Trott, https://www.xsede.org/documents/271087/586927/Edwards-2013-XSCALE13-Kokkos.pdf
  - "A next generation LAMMPS: preparing for the many-core future with Kokkos", Christian Trott, http://lammps.sandia.gov/workshops/Aug13/Trott/LAMMPS-Kokkos3.pdf
  - "Towards Performance-Portable Applications through Kokkos: A Case Study with LAMMPS", Christian Trott, Carter Edwards, and Simon Hammond, http://on-demand.gputechconf.com/supercomputing/2013/presentation/SC3103_Towards-Performance-Portable-Applications-Kokkos.pdf

Preparing LAMMPS for exascale: the Kokkos approach

Goal: separate the physics code from the hardware details

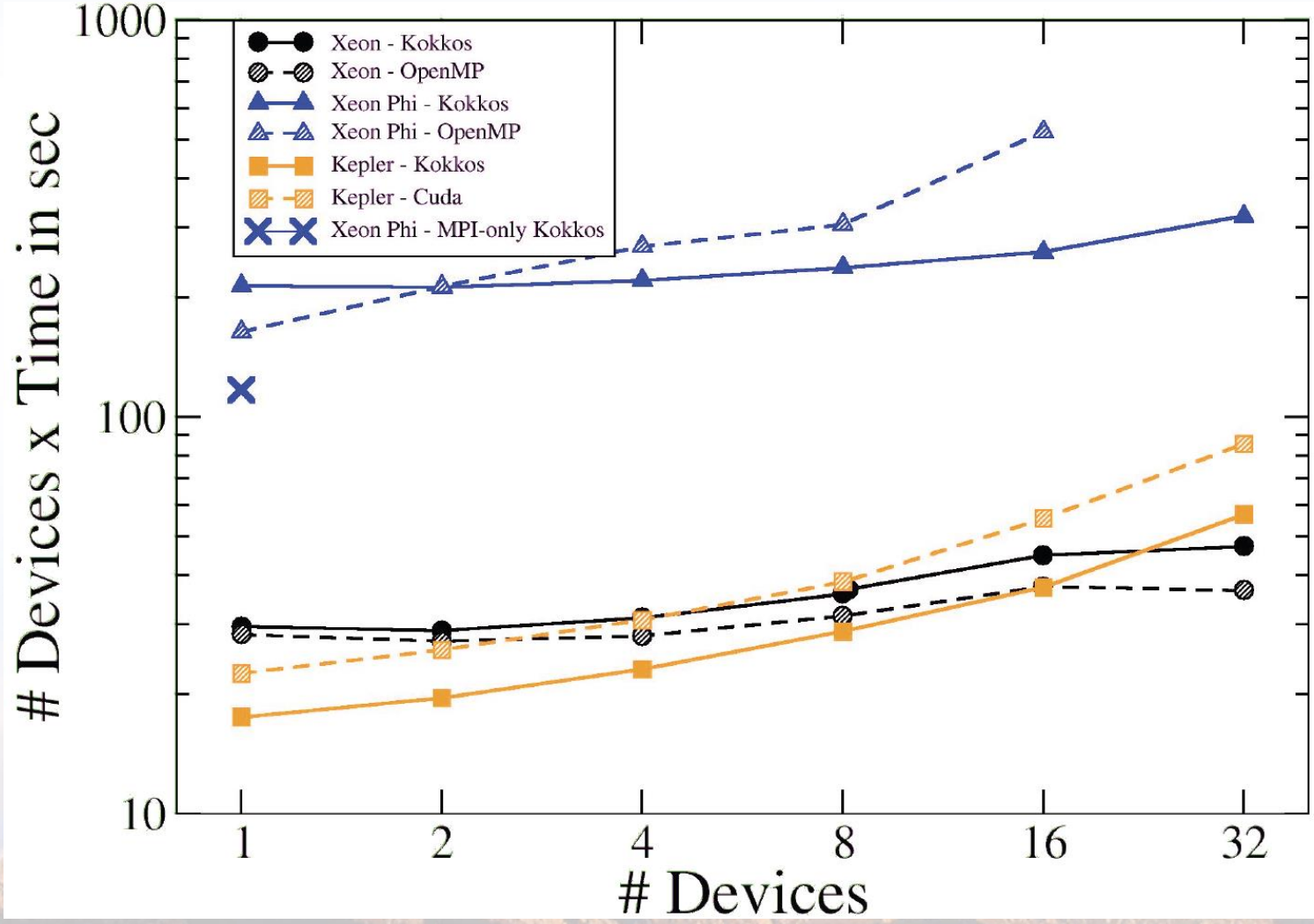
- **Kokkos is a programming model with two major components:**

1. **Data access abstraction**
   - Change data layout at compile time without changing access syntax
     → Optimal access pattern for each device
   - Transparent data padding and alignment
   - Access traits for portable support of hardware specific load/store units

2. **Parallel dispatch**
   - Express algorithms with parallel_for, parallel_reduce, etc.
   - Uses functor concept
   - Transparently mapped onto back-end languages (e.g. OpenMP, CUDA)

LAMMPS-Kokkos results

- 10^6 LJ atoms
- Strong scaling
- Lower is better
- Good Kokkos performance on all three
- Fastest: Kokkos on Kepler



Conclusions

- **Kokkos helping us "future proof" LAMMPS**
- **Unified code base for CPUs, Xeon Phi, GPUs, …**
- **Good performance: >90% of "native" implementations**
- **Extensible: use new back-ends without changing the code**

For more information: Paul Crozier (pscrozi@sandia.gov)

LAMMPS: http://lammps.sandia.gov

Kokkos:

- http://trilinos.sandia.gov/packages/kokkos
- "Kokkos: Enabling performance portability across manycore architectures", Carter Edwards and Christian Trott, https://www.xsede.org/documents/271087/586927/Edwards-2013-XSCALE13-Kokkos.pdf

LAMMPS + Kokkos: "Towards Performance-Portable Applications through Kokkos: A Case Study with LAMMPS", Christian Trott, Carter Edwards, and Simon Hammond, http://on-demand.gputechconf.com/supercomputing/2013/presentation/SC3103_Towards-Performance-Portable-Applications-Kokkos.pdf