Exceptional service in the national interest

Sandia National Laboratories

LAMMPS ReaxFF Benchmark Deep Dive

Stan Moore

February 24, 2018

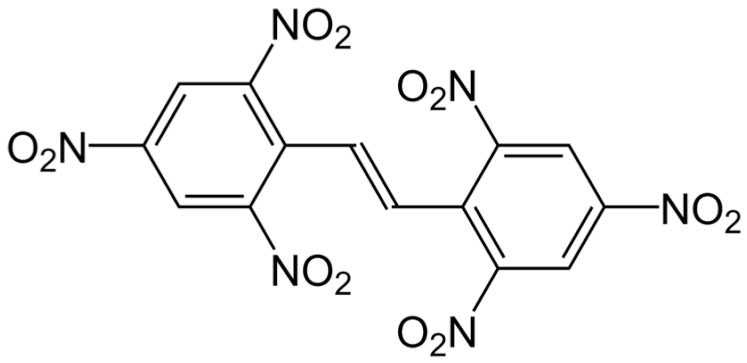
U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

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. SAND NO. 2014-19378 C

LAMMPS ReaxFF Benchmark

- Models the reaction of crystalline Hexanitrostilbene (HNS) energetic material at the atomic scale
- Uses the reactive forcefield (ReaxFF) in LAMMPS



LAMMPS ReaxFF Code

Sandia National Laboratories

- The ReaxFF code has two main parts
  1. Computationally expensive ReaxFF potential, which consists of several deeply nested loops that compute the forces, energy, and pressure of chemically reacting systems
  2. Dynamic charge equilibration (QEq) that computes variable charges on atoms by solving a sparse matrix equation

Kokkos Library

Sandia National Laboratories

- Kokkos is a templated C++ library that provides abstractions to allow a single implementation of an application kernel (e.g. a pair style) to run efficiently on different kinds of hardware such as GPUs
- Kokkos maps the C++ kernel onto different backend languages such as CUDA
- Also provides data abstractions to adjust (at compile time) the memory layout of data structures to optimize performance on different hardware
- For more information on Kokkos, see https://github.com/kokkos

LAMMPS KOKKOS Package

Sandia National Laboratories

- The LAMMPS KOKKOS package contains versions of pair, fix, and atom styles that use data structures and macros provided by the Kokkos library, which is included with LAMMPS in /lib/kokkos
- Currently only supports double precision, no mixed or single precision (on to-do list)

Compiling LAMMPS

Sandia National Laboratories

- To compile the Kokkos CUDA version (uses src/MAKE/OPTIONS/Makefile.kokkos_cuda_mpi):
  - `cd src`
  - `make yes-user-reaxc`
  - `make yes-kokkos`
  - `make -j kokkos_cuda_mpi KOKKOS_ARCH=Power8,Pascal60`
- For more information on building and running with the LAMMPS KOKKOS package, see http://lammps.sandia.gov/doc/accelerate_kokkos.html

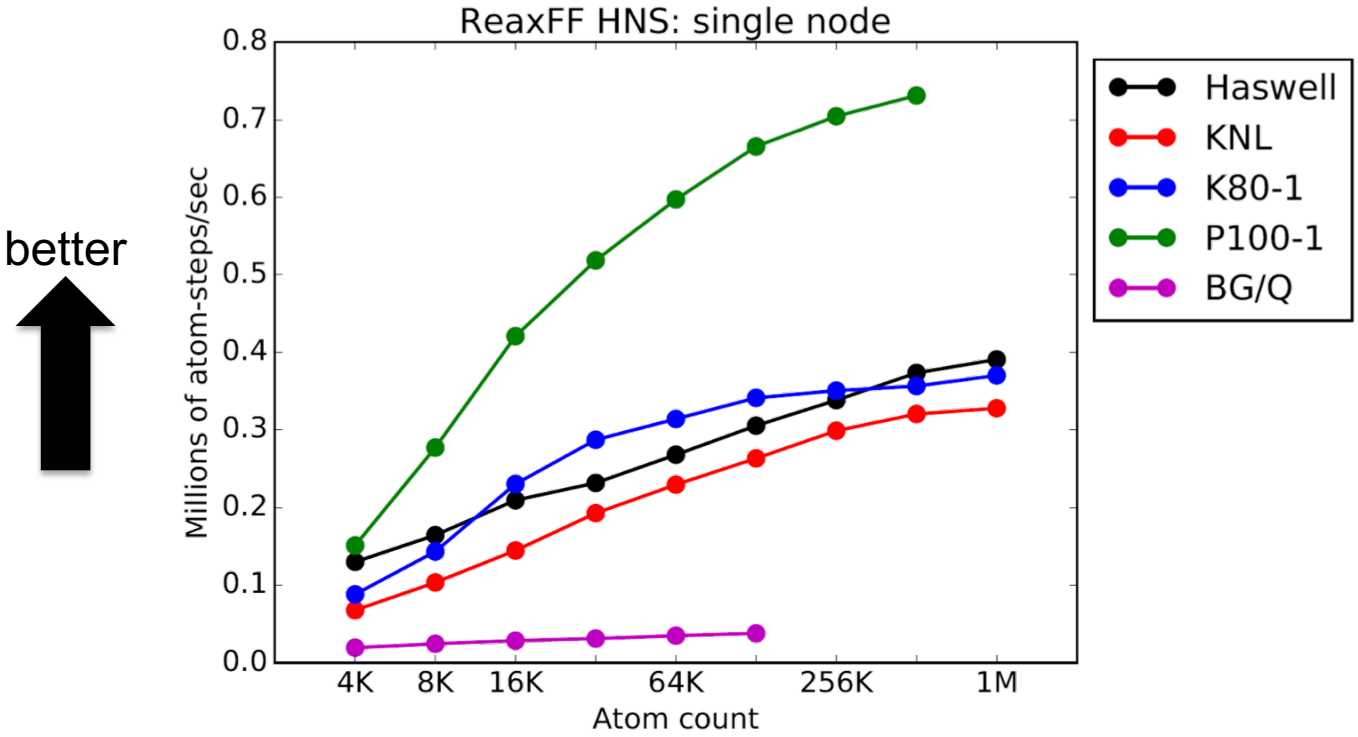
Running the Benchmark

Sandia National Laboratories

- The command "`-v x 1 -v 1 -v z 1`" sets the *x*, *y*, and *z* dimensions of the benchmark. To double the benchmark size (i.e. number of atoms), double the dimension with the lowest value, i.e. use "`-v x 2 -v y 1 -v z 1`".
- To run on 4 P100 GPUs using Kokkos CUDA:
  - `cd reax_benchmark`
  - `mpiexec -np 4 --bind-to core ../src/lmp_kokkos_cuda_mpi -k on g 4 -sf kk -pk kokkos neigh half neigh/qeq full newton on -v x 16 -v y 8 -v z 12 -in in.reaxc.hns -nocite`
- Must use “neigh/qeq full newton” to get good performance. See http://lammps.sandia.gov/doc/package.html

P100 Performance

- Single P100 GPU is 17.6x faster than a single BG/Q node for 116K atoms
- Performance highly dependent on problem size



P100 Performance

Sandia National Laboratories

P100

Profiling application: ../src/lmp_ride100_kokkos_cuda -k on g 1 -sf kk -pk kokkos neigh half neigh/qeq full newton on -v x 8 -v y 8 -v z 8 -v t 100 -in in.reaxc.hns.kokkos_cuda.steps -nocite

Time(%)	Time	Calls	Avg	Min	Max	Name
33.82%	7.55197s	202	37.386ms	3.8966ms	71.156ms	FixQEqReaxKokkos::ComputeHFunctor
16.62%	3.71133s	2272	1.6335ms	1.6259ms	1.6457ms	FixQEqReaxKokkos::SparseMatvec
16.46%	3.67528s	99	37.124ms	36.320ms	37.982ms	PairReaxKokkos::ComputeLJCoulomb
7.28%	1.62674s	186	8.7459ms	1.1944ms	13.837ms	PairReaxKokkos::BuildListsHalf_LessAtomics
7.21%	1.60983s	99	16.261ms	16.147ms	16.429ms	PairReaxKokkos::ComputeTorsion
4.48%	1.00047s	99	10.106ms	9.9881ms	10.252ms	PairReaxKokkos::ComputeAngular
1.88%	419.04ms	7	59.862ms	18.251ms	66.957ms	NPairKokkos::BuildFunctorGhost
1.76%	393.23ms	99	3.9720ms	3.9358ms	4.0529ms	PairReaxKokkos::ComputeBond2
1.29%	287.65ms	3469	82.919us	640ns	520.84us	[CUDA memcpy DtoH]
1.27%	282.58ms	47359	5.9660us	704ns	466.63us	[CUDA memcpy HtoD]

- ComputeHFuntor is part of the charge equilibration, which solves sparse matvec
- Uses parallel scan to generate a CRS graph of a shortened neighbor list
- Only called after neighboring (once every 10 timesteps)

Notes

- Doing host ←→ device data transfer by hand, NOT using UVM
- Only using flat parallelism, not using hierarchal parallelism, thread teams, or shared memory
- May be able to use a different algorithm for sparse matvec (requires constraint that charges sum to zero)