LAMMPS ReaxFF

Purpose of Benchmark

This benchmark models the reaction of crystalline Hexanitrostilbene (HNS) energetic material at the atomic scale. The benchmark uses the reactive forcefield (ReaxFF) in the LAMMPS molecular dynamics code, see http://lammps.sandia.gov.

Characteristics of Benchmark

The ReaxFF code has two main parts. The first is the computationally expensive ReaxFF potential, which consists of several deeply nested loops that compute the forces, energy, and pressure of chemically reacting systems. The second part is a dynamic charge equilibration scheme (QEq) that computes variable charges on atoms by solving a sparse matrix equation.

Mechanics of Building Benchmark

To compile the CPU-only version (uses src/MAKE/Makefile.mpi):

- `cd src`
- `make yes-user-reaxc`
- `make -j mpi`

For more information on building and running LAMMPS, see http://lammps.sandia.gov/doc/Section_start.html.

To compile the native OpenMP version on Vulcan or Sequoia (uses src/MAKE/OPTIONS/Makefile.bgq):

- `export IBM_MAIN_DIR=/opt/ibmcmp/`
- `cd src`
- `make yes-user-reaxc`
- `make yes-user-omp`
- `make -j bgq`

For more information on building and running with the native OpenMP version (USER-OMP package), see http://lammps.sandia.gov/doc/accelerate_omp.html.

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.

Mechanics of Running Benchmark

To run on a single core with the CPU-only version:

- `cd reax_benchmark`
- `mpiexec -np 1 ../src/lmp_mpi -v x 1 -v y 1 -v z 1 -in in.reaxc.hns -nocite`

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`

To run on 96K BG/Q nodes using the native OpenMP version (8 MPI x 8 OpenMP per node):

- `export OMP_PLACES=threads`
- `export OMP_PROC_BIND=true`
- `export OMP_NUM_THREADS=8`
- `cd reax_benchmark`
- `srun -N 98304 --ntasks-per-node 8 ../src/lmp_bgq -sf omp -pk omp 8 -v x 256 -v y 256 -v z 576 -in in.reaxc.hns -nocite`

A CORAL-2 size problem would be several times larger than the above BG/Q example.

Verification of Results

The FOM is given in atom-timesteps/s and can be found by multiplying the "timesteps/s" output from the LAMMPS logfile with the number of atoms in the simulation. For example, if a 1000 atom simulation took 2 seconds to run 100 timesteps, the FOM would be 50,000 atom-steps/s. From an actual LAMMPS logfile:

```
Loop time of 331.426 on 262144 procs for 100 steps with
478150656 atoms
Performance: 0.003 ns/day, 9206.275 hours/ns, 0.302
timesteps/s
```

This would give a FOM of 144.3 million atom-steps/s.

The reference FOM for 96K BG/Q nodes on Sequoia is 3.388 billion atom-steps/s for 11.476 billion atoms (approximately 115K atoms/node). This simulation used 8 MPI x 8 OpenMP threads per node with the native OpenMP (USER-OMP) version.

Most of the benchmark output results are normalized by number of atoms and should remain nearly constant with problem size, except for volume. The output quantities "`Temp PotEng Press E_vdwl E_coul`" should be close (i.e. less than 0.1% different) to an unmodified baseline output (either MPI only, USER-OMP, or KOKKOS) for the same problem (i.e. same geometry, # of MPI, etc).