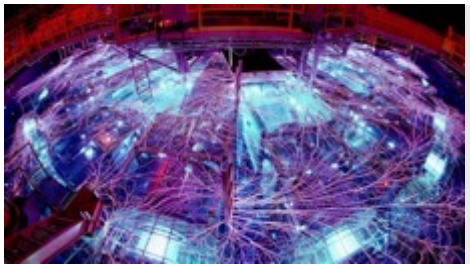
SAND2023-07578C

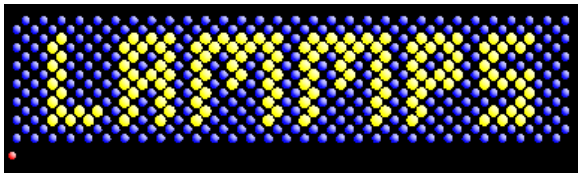
Exceptional service in the national interest

Sandia National Laboratories



CCR
Center for Computing Research

Accelerator Packages and Performance



Stan Moore

2023 LAMMPS Beginners Tutorial

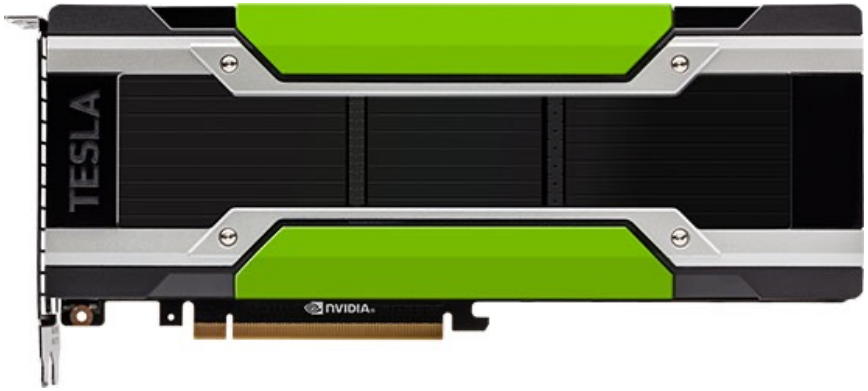
U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories
U.S. DEPARTMENT OF ENERGY NNSA
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Supercomputer Hardware Trends

- Currently 7 out of the top ten supercomputers use GPUs (Graphics Processing Units), according to the June 2023 Top500 List (https://www.top500.org)
- All 3 US exascale machines will have GPUs: OLCF Frontier (AMD), ALCF Aurora (Intel), NNSA El Capitan (AMD)
- LAMMPS accelerator packages: special code (beyond regular C++ and MPI in LAMMPS) that is required to run well on GPUs and many-core CPUs (e.g. CUDA, OpenMP)



http://www.nvidia.com/object/tesla-p100.html

Accelerator Packages

Sandia National Laboratories

- LAMMPS has 5 accelerator packages:
  - OPENMP
  - INTEL
  - OPT
  - GPU
  - KOKKOS
- https://docs.lammps.org/Speed_packages.html

Running OPT Package

- OPT package includes optimized versions of some styles
- Compile LAMMPS with OPT package
- Run with 8 MPI: `mpiexec -np 8 ./lmp_exe -in in.lj -sf opt`
- `-sf opt` is the *suffix* command: automatically appends `/opt` onto anything it can
- For example, `pair_style lj/cut` automatically becomes `pair_style lj/cut/opt` (no changes to input file needed)
- https://docs.lammps.org/suffix.html

OPENMP Package

Sandia National Laboratories

- https://docs.lammps.org/Speed_omp.html
- Uses OpenMP to enable multithreading on CPUs
- **MPI parallelization in LAMMPS is almost always more effective than OpenMP on CPUs**
- When running with MPI across multi-core nodes, MPI often suffers from communication bottlenecks, so using MPI+OpenMP per node *could* be faster
- The more nodes per job and the more cores per node, the more pronounced the bottleneck and the larger the benefit from MPI+OpenMP
- OPENMP package may vectorize (SIMD) better than vanilla LAMMPS styles

Running OPENMP Package

- Compile LAMMPS with OPENMP package
- Run with 2 MPI and 2 OpenMP threads:

```
export OMP_NUM_THREADS=2
mpiexec -np 2 ./lmp_exe -in in.lj -sf omp
```

INTEL Package

- https://docs.lammps.org/Speed_intel.html
- Allows code to vectorize and run well on Intel (and other) CPUs, also supports OpenMP threading
- Can also be used with the OPENMP package
- Normally best performance out of all accelerator packages for CPUs
- Supports reduced precision: mixed FP64+FP32 or pure single FP32

Running INTEL Package

- Compile LAMMPS with INTEL package
- To run using 2 MPI and 2 threads on a Intel CPU:

```
mpiexec -np 2 ./lmp_exe -in in.lj -pk intel
0 omp 2 mode double -sf intel
```

- `-pk` is the package command that sets package options, see https://docs.lammps.org/package.html

GPU Package

Sandia National Laboratories

- https://docs.lammps.org/Speed_gpu.html
- Designed for one or more GPUs coupled to many CPU cores
- Only pair runs on GPU, fixes/bonds/computes run on CPU
- Atom-based data (e.g. coordinates, forces) move back and forth between the CPU(s) and GPU every timestep
- Asynchronous force computations can be performed simultaneously on the CPU(s) and GPU if using Kspace (e.g. PPPM)
- Provides NVIDIA and more general OpenCL support
- Supports reduced precision: mixed FP64+FP32 or pure single FP32

Running GPU Package

Sandia National Laboratories

- Compile GPU library found in lib/gpu
- Compile LAMMPS with GPU package
- Run with 16 MPI and 4 GPUs: `mpiexec -np 16 ./lmp_exe -in in.lj -sf gpu -pk gpu 4`
- **Important**: use CUDA MPS (Multi-Process Service) if using multiple MPI ranks per GPU
- Automatically overlaps pair-style on GPU with Kspace on CPU

Sandia National Laboratories

- Abstraction layer between programmer and next-generation platforms
- Allows the same C++ code to run on multiple hardware (GPU, CPU)
- Kokkos consists of two main parts:
  1. Parallel dispatch—threaded kernels are launched and mapped onto backend languages such as CUDA or OpenMP
  2. Kokkos views—polymorphic memory layouts that can be optimized for a specific hardware

- Used on top of existing MPI parallelization (MPI + X)
- See https://kokkos.github.io/kokkos-core-wiki for more info

LAMMPS KOKKOS Package

Sandia National Laboratories

- https://docs.lammps.org/Speed_kokkos.html
- **Need C++17 compiler**
- Supports OpenMP and GPUs
- Designed so that everything (pair, fixes, computes, etc.) runs on the GPU, minimal data transfer from GPU to CPU
- Relative performance penalty if kernel isn't ported to Kokkos
- Only double precision FP64 support
- Package options can toggle full and half neighbor list, newton on/off, etc.

```
-pk kokkos newton on neigh half
```

- https://docs.lammps.org/package.html

Running Kokkos Package

- Compile LAMMPS with the KOKKOS package
- Run with 4 MPI and 4 GPUs: `mpiexec -np 4 ./lmp_exe -in in.lj -k on g 4 -sf kk`
- Run with 4 OpenMP threads: `./lmp_exe -in in.lj -k on t 4 -sf kk`

Processor and Thread Affinity

Sandia National Laboratories

- Use mpirun command-line arguments (e.g. `--bind-to core`) to control how MPI tasks and threads are assigned to nodes and cores
- Also use OpenMP variables such as `OMP_PROC_BIND` and `OMP_PLACES`
- One must also pay attention to NUMA bindings between tasks, cores, and GPUs. For example, for a dual-socket system, MPI tasks driving GPUs should be on the same socket as the GPU

Measuring performance

```
Loop time of 0.0174524 on 640 procs for 100 steps with 32000 atoms

Performance: 2475308.243 tau/day, 5729.880 timesteps/s
94.1% CPU use with 640 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.0010798  | 0.0013214  | 0.0016188  |   0.2 |  7.57
Neigh   | 0.00021591 | 0.00024434 | 0.0003079  |   0.0 |  1.40
Comm    | 0.015171   | 0.015479   | 0.01573    |   0.1 | 88.69
Output  | 9.0258e-05 | 0.00011218 | 0.00014501 |   0.0 |  0.64
Modify  | 0.00017915 | 0.00018453 | 0.00020567 |   0.0 |  1.06
Other   |            | 0.0001111  |            |       |  0.64
```

- For KOKKOS package on GPUs, timing breakdown won't be accurate without CUDA_LAUNCH_BLOCKING=1 (but will prevent kernel overlap and could slow down simulation)

Parallel MD Performance

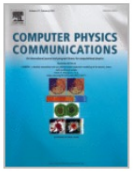
Sandia National Laboratories

- MD parallelizes well: major parts of timestep (forces, neighbor list build, time integration) can be done in parallel through domain decomposition
- High communication overhead when strong scaling to a few 100 atoms/proc (depends on cost of the force-field)
- Strong scaling: hold system size fixed while increasing processor count (# of atoms/processor decreases)
- Weak scaling: increase system size in proportion to increasing processor count (# of atoms/processor remains constant)
- For perfect strong scaling, doubling the processor count cuts the simulation time in half
- For perfect weak scaling, the simulation time stays exactly the same when doubling the processor count
- Harder to maintain parallel efficiency with strong scaling because the compute time decreases relative to the communication time

Performance of Different Potentials

- See plots in LAMMPS reference paper
- https://doi.org/10.1016/j.cpc.2021.108171

Computer Physics Communications
Volume 271, February 2022, 108171



Feature article

LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales ☆, ☆☆

Aidan P. Thompson [a], H. Metin Aktulga [b], Richard Berger [c], Dan S. Bolintineanu [a], W. Michael Brown [d], Paul S. Crozier [a], Pieter J. in 't Veld [e], Axel Kohlmeyer [c], Stan G. Moore [a], Trung Dac Nguyen [f], Ray Shan [g], Mark J. Stevens [a], Julien Tranchida [a], Christian Trott [a], Steven J. Plimpton [a]

Thank You

Sandia National Laboratories

Questions?