Strong scaling of general-purpose molecular dynamics simulations on GPUs

Duncan Stewart

The issue

- The introduction and proliferation of increasingly powerful Gpu's is a huge opportunity to improve the speed of molecular dynamics (MD) simulations

  - MD benefits greatly from parallel computation because computing Newtons equations of motion can be done independently for each molecule in the simulation.

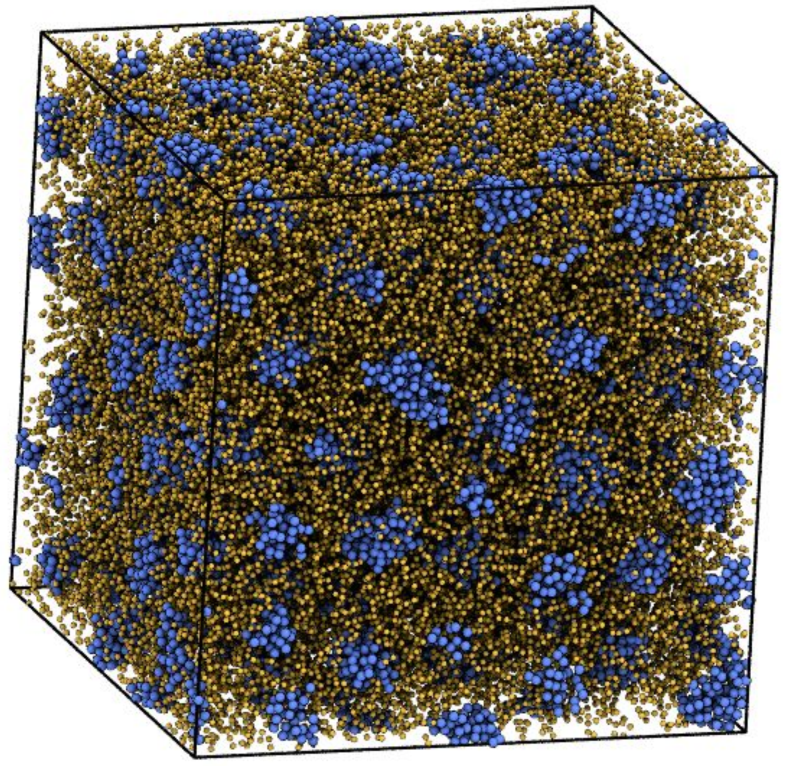
  - As supercomputing clusters such as Titan and Blue Waters are built with thousands of gpu's each, MD codes need to be able to efficiently use many Gpu's at the same time

Current state of the art

- Most major MD codes such as Gromacs, Lammps, or Charmm, already include support for single-Gpu acceleration
  - Since they were written for Cpu's they can't take full advantage of the Gpu
    - These implementations only use the gpu for the compute-intensive step, communicating back and forth with the Cpu on every time-step
    - Data structures used are optimized for the cpu not the gpu, resulting in poor scaling as the number of Gpu's increases
  - Codes such as Fen-Zi and HALMD are designed to fully utilize gpu resources
    - These codes have severely limited features making them specialty tools for specific applications only

What’s new

- HOOMD-blue was built from the beginning to fully utilize Gpu’s, using the Cpu as a driver.
  - New version extends the code to support multiple Gpu’s
  - Retains the robust feature set expected of a general-purpose MD code
  - Introduces automatic tools to ensure optimal performance on multiple Gpu’s


Simulation of a triblock copolymer done with HOOMD-blue

Communication scheme

- Communication between Gpu's is achieved via domain decomposition, and communicating only along the borders of those domains
  - Domain decomposition is accomplished by splitting the simulation box into a 1, 2, or 3 dimensional grid, particles on the boundaries are communicated to compute short-range attractive and repulsive forces
  - Communication is done by each Gpu sending information (position, velocity, index, destination rank) about particles on the edges of its domain to the host, the host then sorts the data and sends it to the appropriate destinations
    - Doesn't directly communicate across domain borders like Lammps does as that introduces data dependency between exchanges
      - Information received in one step may need to be sent in the next, so communication needs to be blocking

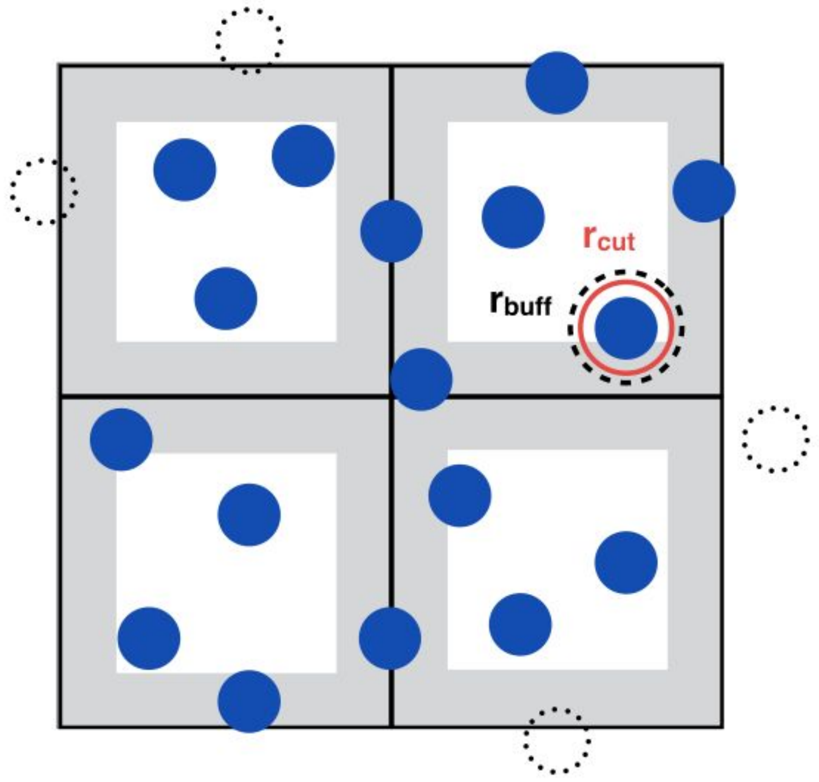


Fig. 1. Domain decomposition with ghost particles and periodic boundary conditions. Shaded regions at the domain boundaries correspond to ghost particle layers. Short-ranged pair forces require a neighbor cut-off distance r_{cut} and a buffer length r_{buff}, and their sum defines the ghost layer width.

Optimizations

- Previous versions of HOOMD have mapped one particle to one thread
  - Using multiple Gpu's and a small system size this becomes inefficient as the Gpu is underutilized
  - In this version each particle is assigned to an array of 2^n threads in order to ensure full utilization
- Introduces an automatic parameter tuning feature
  - It tests all permutations of block and thread array sizes over the first 10-20 thousand time steps of the simulation and uses the best combination
  - Periodically re-checks in case different parameters perform better later in the simulation

Weak scaling results

- Weak scaling was tested against the gpu version of Lammps, simulating up to 108 million particles on up to 3375 gpu's. The simulation were equilibrated for 100,000 time steps, then results were averaged over the next 10,000

  - HOOMD-blue consistently performed around twice as quickly as Lammps for all system sizes tested.

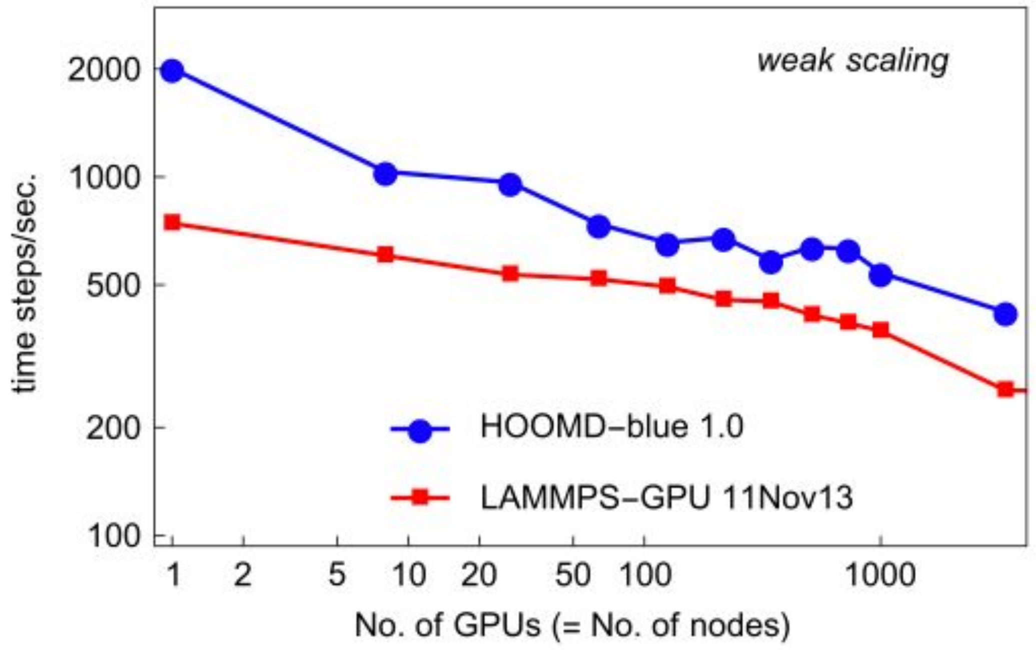


Fig. 6. Weak scaling performance of HOOMD-blue (solid circles) and LAMMPS-GPU (solid squares) on Titan, in number of time steps per second vs. number of GPUs, for a constant number of particles per GPU $N/P = 32{,}000$, for 1 . . . 3375 GPUs.

Strong scaling results

- Strong scaling was tested against gpu Lammps again, simulating about 11 million particles on an ever increasing amount of gpu's
  - Nearly ideal scaling is seen up to 32 gpu's, then scaling continues at decreased efficiency up to 1024 gpu's. As before, HOOMD performs about 2x as well as Lammps until more than 1024 gpu's are used and both run in to architecture-inherent communication bottlenecks and perform similarly.

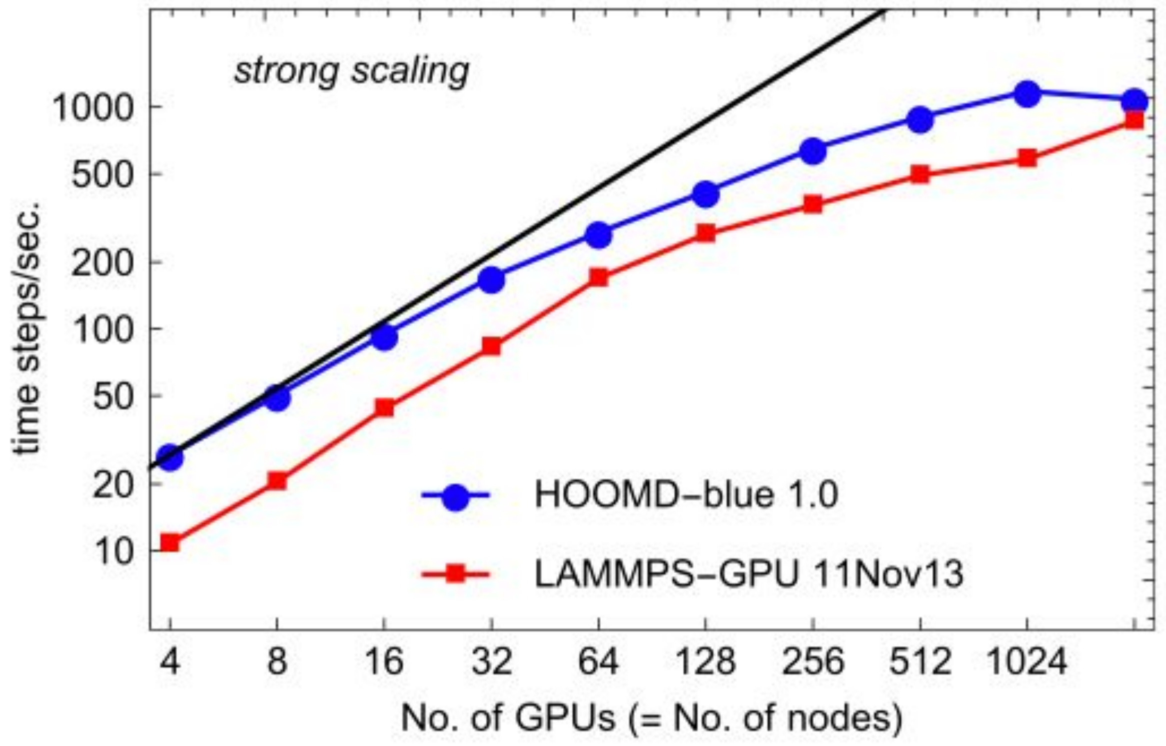


Fig. 7. Strong scaling of a LJ liquid benchmark with $N = 10976{,}000$ particles (single precision) on the Titan supercomputer (Cray XK7, Oak Ridge National Laboratories). Shown is the performance in terms of number of time steps per second (TPS) for HOOMD-blue (circles) and LAMMPS-GPU (squares), running on $P = 4 \ldots 2048$ GPUs. HOOMD-blue performance is optimal with 1 MPI rank per node/GPU, for LAMMPS-GPU up to 16 CPU cores are assigned to one device. The solid line shows ideal linear scaling.

Opinion

- Based on what is shown HOOMD-blue accomplished what it set out to do
  - It demonstrates impressive results for large system sizes
  - scales well with increasing numbers of Gpu’s
- It is strange that the authors only report HOOMD’s performance against Lammps and not any of the other popular simulation packages
  - Lammps is widely considered one of the most robust MD packages, but also one of the slowest
  - Gpu accelerated versions of other codes such as Gromacs exist, so why weren’t they tested?