Using LAMMPS Compiled with GPU for MD Simulations

Audrey Wu

June 26, 2025

1 Basic Script

The script used for LAMMPS with CPU and LAMMPS with GPU is essentially the same. LAMMPS documentation from 2010 says to add a line that says

```
newton off
```

(which turns Newton pairwise flag off), but this is no longer necessary as the default is now set to **newton off**. In fact, LAMMPS does not currently support Newton pairwise flag being turned on.

One of the main differences between CPU and GPU LAMMPS is the potentials. The script for both is the same, but the potentials themselves are different. At the end of a particular potential you are using, you need to add **/gpu**. This will tell LAMMPS that you are intending to run the potential using a GPU accelerator. For example, if your script looked like

```
pair_style tersoff/zbl
pair_coeff * * 2007_SiO_tersoff.zbl Si O NULL
```

it should now read

```
pair_style tersoff/zbl/gpu
pair_coeff * * 2007_SiO_tersoff.zbl Si O NULL
```

Note: not all potentials are supported on the GPU-enabled version of LAMMPS, so you may need to experiment and try different potentials if the one you have been using is not supported.

2 Running the Simulation

You still run the simulation in the terminal as you would for CPU LAMMPS, but the command line is a little different. To run a GPU simulation, the command line is

```
lmp -sf gpu -pk gpu #GPU -in file.txt
```

where #GPU is the number of GPUs you want to run the simulation on.

To run a simulation on both GPU and CPU, the command is

```
mpirun -np #CPU --bind-to core lmpGPU -sf gpu -pk gpu #GPU -in file.txt
```

where #CPU is the number of cores you want to run the simulation on. This will run the script using GPU on the potentials marked with **/gpu** and using CPU on everything else. Note: if you wish to run on a specific number of threads on your cores, make sure to run

```
export OMP_NUM_THREADS=#
```

either in the terminal or at the beginning of your script.

3 Neighbor Lists

You may run into an issue where error given is saying there is a problem with the neighbor lists on the GPU, particularly if you are running a joint GPU/CPU simulation. To fix this, before the line that reads in the crystal file, write

```
package gpu #GPU
neigh_modify binsize 2.0
neigh_modify every 1 delay 2 check yes one 10000 page 1000000
```

The **neigh_modify** commands can be altered, and the one and page commands do not need to be included unless you are running a large crystal. The one command places an upper limit on the amount of neighbors any one atom can have, and the page command places an upper limit on the amount of neighbors that can be included on any one "page" of the neighbor list. The number following page must be at least double the number following one. After you run your potentials, but before any other command that would cause neighbor lists to be generated (run, minimize, etc.), write

```
package gpu 0
neigh_modify binsize 2.0
neigh_modify every 1 delay 2 check yes one 1000000 page 10000000
atom_modify sort 0 0.0
```

to tell LAMMPS that the neighbor lists should now be stored on CPU. This helps prevent running out of memory on the GPU during large runs. If you have lines fixing temperature, such as these

```
fix equil all nvt temp 300.0 300.0 100.0
run 50000 # 5 ps
unfix equil
```

I have found that placing the **package gpu 0** lines directly after the **unfix equil** line works best. For more information on the **package** command, go to `https://docs.lammps.org/package.html`