the inf−norm is the largest component. The last 2 lines are statistics on how many iterations and force−evaluations the minimizer required. Multiple force evaluations are typically done at each iteration to perform a 1d line minimization in the search direction.

If a kspace_style long−range Coulombics solve was performed during the run (PPPM, Ewald), then additional information is printed, e.g.

```
FFT time (% of Kspce) = 0.200313 (8.34477)
FFT Gflps 3d 1d-only = 2.31074 9.19989
```

The first line gives the time spent doing 3d FFTs (4 per timestep) and the fraction it represents of the total KSpace time (listed above). Each 3d FFT requires computation (3 sets of 1d FFTs) and communication (transposes). The total flops performed is 5Nlog_2(N), where N is the number of points in the 3d grid. The FFTs are timed with and without the communication and a Gflop rate is computed. The 3d rate is with communication; the 1d rate is without (just the 1d FFTs). Thus you can estimate what fraction of your FFT time was spent in communication, roughly 75% in the example above.

2.8 Running on GPUs

A few LAMMPS pair styles can be run on graphical processing units (GPUs). We plan to add more over time. Currently, they only support NVIDIA GPU cards. To use them you need to install certain NVIDIA CUDA software on your system:

- Check if you have an NVIDIA card: cat /proc/driver/nvidia/cards/0
- Go to http://www.nvidia.com/object/cuda_get.html
- Install a driver and toolkit appopriate for your system (SDK is not necessary)
- Run make in lammps/lib/gpu, editing a Makefile if necessary
- Run lammps/lib/gpu/nvc_get_devices to list supported devices and properties

GPU hardware

When using GPUs, you are restricted to one physical GPU per LAMMPS process. This can be multiple GPUs on a single node or across multiple nodes. For each GPU pair style, the first two arguments (GPU mode followed by GPU ID) control how GPUs are selected. If you are running on a single node, the mode is "one/node" and the parameter is the ID of the first GPU to select:

```
pair_style lj/cut/gpu one/node 0 2.5
```

The ID is the GPU ID reported by the driver for CUDA enabled graphics cards. For multiple GPU cards on a node, an MPI process should be run for each graphics card. In this case, each process will grab the GPU with ID equal to the process rank plus the GPU parameter.

For multiple nodes with one GPU per node, the mode is "one/gpu" and the parameter is the ID of the GPU used on every node:

```
pair_style lj/cut/gpu one/gpu 1 2.5
```

In this case, MPI should be run with exactly one process per node.

For multiple nodes with multiple GPUs, the mode is "multi/gpu" and the parameter is the number of GPUs per node:

```
pair_style lj/cut/gpu multi/gpu 3 2.5
```

In this case, LAMMPS will attempt to grab 3 GPUs per node and this requires that the number of processes per node be 3. The first GPU selected must have ID zero for this mode (in the example, GPUs 0, 1, and 2 will be selected on every node). An additional constraint is that the MPI processes must be filled by slot on each node such that the process ranks on each node are always sequential. This is a option for the MPI launcher (mpirun/mpiexec) and will be the default on many clusters.

GPU single vs double precision

See the lammps/lib/gpu/README file for instructions on how to build the LAMMPS gpu library for single vs double precision. The latter requires that your GPU card supports double precision. The lj/cut/gpu pair style does not support double precision.

2.9 Tips for users of previous LAMMPS versions

The current C++ began with a complete rewrite of LAMMPS 2001, which was written in F90. Features of earlier versions of LAMMPS are listed in this section. The F90 and F77 versions (2001 and 99) are also freely distributed as open−source codes; check the LAMMPS WWW Site for distribution information if you prefer those versions. The 99 and 2001 versions are no longer under active development; they do not have all the features of C++ LAMMPS.

If you are a previous user of LAMMPS 2001, these are the most significant changes you will notice in C++ LAMMPS:

(1) The names and arguments of many input script commands have changed. All commands are now a single word (e.g. read_data instead of read data).

(2) All the functionality of LAMMPS 2001 is included in C++ LAMMPS, but you may need to specify the relevant commands in different ways.

(3) The format of the data file can be streamlined for some problems. See the read_data command for details. The data file section "Nonbond Coeff" has been renamed to "Pair Coeff" in C++ LAMMPS.

(4) Binary restart files written by LAMMPS 2001 cannot be read by C++ LAMMPS with a read_restart command. This is because they were output by F90 which writes in a different binary format than C or C++ writes or reads. Use the *restart2data* tool provided with LAMMPS 2001 to convert the 2001 restart file to a text data file. Then edit the data file as necessary before using the C++ LAMMPS read_data command to read it in.

(5) There are numerous small numerical changes in C++ LAMMPS that mean you will not get identical answers when comparing to a 2001 run. However, your initial thermodynamic energy and MD trajectory should be close if you have setup the problem for both codes the same.