A Practical Demonstration of Running a First LAMMPS Simulation on ISAAC

David Keffer

Department of Materials Science & Engineering

University of Tennessee, Knoxville

date begun: January 27, 2016

date last updated: February 1, 2024

Table of Contents

I. Purpose of Document

The purpose of this document is to provide a complete and unambiguous demonstration of running a simple LAMMPS simulation on the ISAAC cluster at the University of Tennessee. This document includes specific instructions and sample files.

II. Accessing the ISAAC Open Enclave HPSC cluster

II.A. Request an Account

1. Use your web browser to go to https://oit.utk.edu/hpsc/isaac-open-enclave-new-kpb/
2. On the top left of the page click on "Request an ISAAC Account"
3. You will be redirected to a new page. Click on "I have a <UT NetID>".
4. Use your <UT netID>. Your <UT netID> is the first part of your utk email address. Example: my utk email address is dkeffer@utk.edu, so my <UT netID> is dkeffer.
5. Login using your <UT netID> and password. (Same as your email password.)
6. You will get a DUO push on your registered phone number. Approve it.
7. Complete the form and submit the form.
8. You will get an email when your account is approved and created.

II.B. Authentication

ISAAC requires a two-step authentication to log on.

If you are already using other UT services such as UT email, you may already have DUO push enabled and are using it. If you have not or you got a new phone and set up new phone for DUO push, refer to the link: Duo 2FA: Mobile Phone Setup

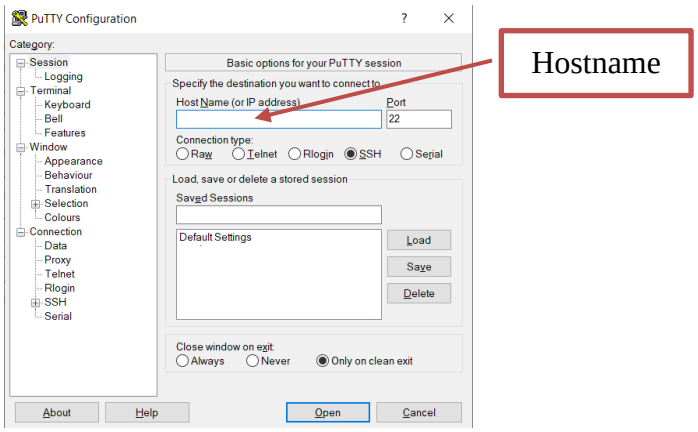
II.C. Logging on to the ISAAC

Mac or Linux Operating System

- Open the Terminal.
- Type ssh <UT NetID>@<hostname>, see Table 1 to find the hostname for the cluster.

Windows Machines

- To access the cluster from Windows machines, please download the PuTTY software from www.putty.org and install it.
- Open the PuTTY application and enter the hostname mentioned in Table 1.



- In putty, Click Open.
- You will be prompted to enter your username. Enter your <UT NetID> and press "Enter".
- Provide your password (same as the one which you use to log in to your UTK email).
- Press 1 or 2 to finish DUO authorization.

Hostname	login.isaac.utk.edu
Username	<UT NetID>
Password	Your <UT NetID> password + "Duo push"
Port	22

Table 1. Login information for ISAAC.

The complete details of how to access and login to ISAAC HPSC cluster can be found on https://oit.utk.edu/hpsc/isaac-open-enclave-new-kpb/access-and-login-isaac-ng/

II.D. Access to class project (ISAAC-UTK0271)

Anyone with an account on ISAAC can use ISAAC in "opportunistic mode", which is a low priority mode. Some computational resources have been reserved for this class. In order to have priority on these nodes, you must access the class project, which is designated: "ACF-UTK0271".

Please visit https://oit.utk.edu/hpsc/overview/user-portal/ for step-by-step instructions to add yourself to an existing project. These instructions are implemented on the user portal page at https://portal.nics.utk.edu/.

III. File Storage on ISAAC

III.A. Directories

You will have a home directory on ISAAC located at

/nfs/home/$USER

Here $USER is your username on ISAAC, which is same as your UTK netID. This has a quota of 50 GB and is backed up.

Often you will require more than 50 GB to run a job. In that case you should change to your scratch directory. Your scratch directory is located at

/lustre/isaac/scratch/$USER

You can change to scratch directory by launching the command **cd $SCRATCHDIR**. On ISAAC NG, each user gets a default quota of 10 TB for scratch storage.

The scratch directory is temporary storage. Old files are deleted after 30 days. **Don't store important data here!** If you have files larger than your quota and you need to keep it, transfer it to an external storage, such as the hard drive of your personal computer or an external hard drive attached to your personal computer.

III.B. Nodes

Nodes are not the same thing as directories. Nodes are collections of motherboards (processors and local memory). Any node can access your data stored in your home directory or scratch directory. For our purposes, there are only three types of nodes on ISAAC:

1. login nodes: this is the node you login into.
    - This is the node you will use with PuTTy or your terminal in section II.C. above.
2. data transfer node: this node is for getting files to and from ISAAC.
    - This is the node you will use with FileZilla in section IV below.
3. compute nodes: these nodes are where the calculations are performed.
    - These nodes we will access for interactive and batch jobs as described in section VIII. below.

IV. Data Transfer to and from ISAAC

The straightforward tool for data transfer is the shareware, FileZilla, located at https://filezilla-project.org/. The two-step authentication employed by renders FileZilla requires a minor tweak to FileZilla. There is a detailed set of instructions for getting FileZilla to work properly with an interactive login located on the course website. Here we summarize:

1. Go to File --> Site Manager --> General Tab
    - Host: dtnX.isaac.utk.edu where X = 1 or 2
    - for example: Host: dtn1.isaac.utk.edu
    - Port: 22
    - Protocol: SFTP
    - Logon Type: Interactive

- user: ISAAC-userid (probably your netID)

2. On the Transfer Settings Tab
   - Check "Limit Number of Simultaneous Connections"
   - Set "Maximum Number of Connections" to 1
3. Click "Connect"
4. Enter Password, including duo push

A two window interface opens. The window on the left is your local personal computer. The window on the right is your home directory on ISAAC. FileZilla moves files with drag-and-drop.

There are alternatives to using FileZilla. If you are interested in using Globus, then OIT has a few recorded videos on how to create the endpoint in your personal computer for data transfer. The videos are available under the title Globus Connect Personal Tutorials on Workshops and Trainings page under Overview drop-down menu on our HPSC documentation page.

V. Navigating in a Linux Environment

Once you have logged in via section II.C, here are a few useful Linux commands.

- Locate where you are using the `pwd` command. (print working directory)
- Examine the contents of the current directory, using the `ls -al` command. (list)
- Change your location with the `cd` command (change directory)
  - go down one directory: `cd <subdirectory>`
  - go up one directory: `cd ..`
  - go to your home directory: `cd`
  - go to your scratch directory: `cd /lustre/isaac/user/userid`
  - or go to your scratch directory: `cd $SCRATCHDIR`
- Make a directory for this class, using the `mkdir mse614` command.
- Change to the newly created directory, using the `cd mse614` command.
- Make a directory within `mse614` for this demo, using the `mkdir test_01` command.
- Change to the newly created directory, using the `cd test_01` command

A summary of basic Linux commands is given in Appendix A. Common Linux Commands.

VI. Editing Files

The Linux environment has a default editor, `vi`. It is a line editor, which predates the existence of a computer mouse. While `vi` is a powerful tool, cherished by programmers, many non-programmers find it a very onerous file editor. There are numerous `vi` tutorials online.

Files also can be edited on ISAAC using the `nano` command:

- edit a file in the current directory: `nano filename`

In the age of rapid file transmission, there is an acceptable and more user-friendly alternative, which we will use in this demonstration. We will use the free software, Notepad++. This software can be downloaded from the following URL: https://notepad-plus-plus.org/ .

VII. LAMMPS Input Files

Typically LAMMPS requires two files to run a simulation. The first is the "input file" and the second is the "configuration file". The "input file" contains a sequence of lammps commands. The "configuration file" contains the initial positions of the atoms.

VII.A. Input File

We are going to open Notepad++ and create a new empty file titled `in.lammps`. This will be our LAMMPS input file.

In Appendix B of this document is a sample input file to perform a simulation of a Lennard-Jones fluid. For the purpose of this demonstration, we are going to simply copy the entire contents of Appendix B into our input file `in.lammps`, opened in Notepad++. Once this file is saved, we have our input file.

Note that in the input file, all lines that begin with the "#" sign (formerly called a 'pound' sign and more recently called a 'hash' sign) are comments. They are not required for the code to run. They are included to help make the purpose of each line more clear. All blank lines are also unnecessary. Blank lines are included to make the organization of the input file more apparent to the eye. Any other line begins with a LAMMPS command and is followed by inputs for that particular command.

The LAMMPS manual is online at the following URL: http://lammps.sandia.gov/ . The easiest way to find out what a command means and what options are available to it is google. For example, googling "lammps units" directs us to the appropriate page within the lammps manual, http://lammps.sandia.gov/doc/units.html , with all of the relevant information on the units command.

VII.B. Configuration File

In Appendix C of this document is a sample configuration file for a Lennard-Jones fluid, `config.txt`.

The configuration file provides the initial x, y and z coordinates of each atom along with atom identity. It also stipulates initial simulation volume. The configuration file can include potential information, specifically for molecular systems.

VIII. Running LAMMPS

VIII.A. Transfer Input and Configurations Files to Scratch Directory on ISAAC

Use FileZilla to transfer `in.lammps` and `config.txt` to the subdirectory on your scratch directory where you want to run the simulation.

The combination of Notepad++ and FileZilla allows us to edit any file locally in a user-friendly software and then transfer it immediately to the remote destination, where it can be used.

VIII.B. Interactive Job

There are two types of jobs on ISAAC: interactive and batch. Interactive jobs are short in duration and use only one or two processors. The purpose of an interactive job is for debugging. You want to make sure that you don't have a mistake in your input file or configuration file before you tell the code to run a million steps. (Or before you wait overnight in the queue only to discover your code died before you could execute the first step.) In an interactive job, you have to stay logged in while your job is running, so keep it as short as possible.

You don't want to run any jobs on the login node. Therefore the first thing you do to run an interactive job is to move off login node and onto a computer node:

```
salloc  --account=ISAAC-UTK0271 --partition=condo-mse614 --qos=condo --nodes=1 --ntasks-per-node=1 --time=01:00:00
```

or

```
salloc -A ISAAC-UTK0271 -p condo-mse614 -q condo -N 1 --ntasks-per-node=1 -t 1:00:00
```

Wait approximately thirty seconds for a new prompt. You will have received a message telling you which compute node you now have access to.

```
Nodes clr0826 are ready for job
```

Now ssh to that compute node.

```
ssh clr0826
```

You should now see that your prompt tells you that you are on a compute node.

```
[dkeffer@clr0826 ~]$
```

At this point you load the lammps module

```
module load lammps/02Aug2023
```

You are ready to run the lammps simulation

```
srun -n 1 lmp_mpi < inputfile > outputfile
```

VIII.C. Batch Job

To run a lammps simulation for data production, you will need to submit a "batch job", in which the run command is submitted to the job queue via a job file. For a batch job, you don't have to stay logged in while your job is running. You can disconnect from the machine and go to bed. Hopefully, in the morning, if there were no errors in your file, the job is done.

VIII.C.1. Create Job File for the Queue

We are going to run the job through the queue. In this example we simply present the job file that we are going to use in Table 2. We will call this file `batch_job`. We can get this file onto ISAAC by copying the contents of the appendix into a file in Notepad++ and then transferring it to the cluster using FileZilla, exactly as we did for the input and configuration files.

The first line specifies the shell, bash, the Bourne Again SHell.

Subsequent lines specify the project you are associated with, which will determine your priority in the queue. We also specify the number of nodes and the processors per node (ppn) that you request and the time you request. If you specify many nodes or a long time, you will sit in the queue for a longer time. Therefore, the goal is to conservatively set just the number of nodes that you need and to set the time to be 50% longer than you anticipate. If a batch job exceeds its time before completion, it is killed. Knowing how much time you need is often determined from a shorter run where you can calculate the number of lammps steps per minute that you observe. It is very much system size dependent.

The last line runs the program. It launches MPI (the command that tells the code that you are running a program, lammps, that utilizes MPI). MPI, Message Passing Interface, is a library of subroutines for allowing processors on a multiprocessor machine to communicate with each other. We choose two processors. For this project, our node has 48 processors, so we cannot set the ntasks-per-node greater than 48. The ntasks is the product of the nodes and the ntasks-per-node. For this project, we are only allocated 1 node. Since we are sharing this node, do not use all 48 processors at the same time.

```
#!/bin/bash

#SBATCH --account=ISAAC-UTK0271
#SBATCH --job-name=lmps_test
#SBATCH --partition=condo-mse614
#SBATCH --time=1:00:00  # format is hours:minutes:seconds
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --ntasks=2
#SBATCH --qos=condo
#SBATCH --error=job.e
#SBATCH --output=job.o

############## Load any modules #########
module load lammps/02Aug2023

srun lmp_mpi < inputfile > outputfile
```

Table 2. Job file for cluster queue. This is located in a file named *batch_job*.

VIII.C.2. Submit Job File for the Queue

We next submit the job file to the queue.

- The job script is submitted using the command sbatch

  sbatch batch_job

- To check the status of the jobs:

  squeue -u <netid>

- To cancel the job, you need the job id of the job:

  scancel <jobid>

IX. Examining LAMMPS Output

An interactive job generates three files. The first is the output to the screen, which we have redirected to `lammps.out` when we ran lammps at the command prompt. The second is a log file, `lammps.log`. The third is a trajectory file, with a name specified in input file, `output.xyz`. The output file and the log file contain some redundant information, but are not completely the same. If you periodically print out thermodynamic properties, this information will appear in both these files. The log file also includes a restatement of each command in the input file. The trajectory file contains the xyz coordinates of the atoms saved every `n` steps during the simulation. Any sort of post simulation analysis is based on the contents of the trajectory file. The trajectory file is also the source of snapshots (a single frame) and movies (many frames) from the simulation.

A batch job generates two additional files. If the `<process iD>` is, for example, 6397478, then the files `pbs_job.e467589` and `pbs_job.o467589` will appear during the running of the job. The file with the "`o`" is the output file. Since we have redirected our output to `lammps.out`, this file should be empty. The file with the "`e`" is the error file. If there is an error in the execution of lammps, a message (sometimes cryptic) will appear here. In the best case scenario, the program executes without error and the error file is empty. An inconsistency in the lammps input file is the most common cause of error. A problem with ISAAC is also possible, but less common.

The file `lammps.out` contains the redundant output that would have been printed to the screen had it been an interactive job. This file is reproduced in Appendix D.

The most relevant information is the table of thermodynamic properties. There are three such tables corresponding to the NVE equilibration, the NVT equilibration and the NVT data production runs. The most relevant information is the table of thermodynamic properties from the NVT data production. This information is summarized in Table 3. This is a toy simulation. For a longer simulation, the equilibration table would be studied to ensure that the system had actually reached equilibration. Once that was validated the data production run could be used to generate publishable data.

It is important to note that the thermodynamic properties reported in this table are instantaneous values. All analysis of averages and fluctuations must be performed as part of the post-processing. This analysis can be performed in any software of your choice (Excel or Matlab) simply by copying the relevant table into the software.

The file `output.xyz` contains the atomic coordinates in an xyz format. This too can be viewed in Notepad++, at least for a small file such as was generated in this example. This file contains 5140 lines composed of 10 blocks of 514 lines each. The 514 lines compose a

configuration including two header lines (the first reporting the number of atoms in the configuration, 512 in this case, and the second reporting the time associated with the configuration. The last 512 lines are the atomic symbol and coordinates of each atom. There are 10 frames in this file because we ran for 1000 steps and saved every 100 steps, resulting in 10 saved configurations. The additional configuration is the initial configuration at the beginning of the data production run. A sample of the contents of this file are shown in Table 4.

These configurations can be viewed independently or viewed as a movie using the visualization software of your choice. Virtually all visualization software can read this standard xyz format. In this course, we will use the shareware, Ovito.

```
Step PotEng KinEng TotEng Temp Press Density
    2000    -1821.594    756.59549   -1064.9985   0.98707827   -0.3111245          0.5
    2100   -1805.8618     777.9501   -1027.9117    1.0149382   -0.1243004          0.5
    2200   -1801.1839     797.4162   -1003.7677    1.0403342  -0.10290561          0.5
    2300   -1780.5184     757.9926   -1022.5258   0.98890098  -0.19909778          0.5
    2400   -1803.8874    788.00112   -1015.8863     1.028051  -0.30086952          0.5
    2500   -1806.3675      770.407   -1035.9605    1.0050972  -0.23416908          0.5
    2600   -1818.0796    770.41401   -1047.6656    1.0051063  -0.19240302          0.5
    2700   -1821.7818    770.85111   -1050.9307    1.0056766  -0.34659051          0.5
    2800   -1811.3153    782.00793   -1029.3074    1.0202321  -0.36878276          0.5
    2900   -1818.8302    785.93459   -1032.8956     1.025355  -0.28715067          0.5
    3000   -1841.4016    807.48686   -1033.9147    1.0534727  -0.24091218          0.5
    3100   -1838.2069    843.90237   -994.30449    1.1009816  -0.04616304          0.5
    3200   -1838.4076     774.9594   -1063.4482    1.0110364  -0.31697307          0.5
    3300    -1842.646    772.30297    -1070.343    1.0075707  -0.15439713          0.5
    3400    -1834.814    767.92509    -1066.889    1.0018592  -0.12814917          0.5
    3500    -1850.819    793.94239   -1056.8766    1.0358022  -0.33874609          0.5
    3600   -1849.1511    754.52376   -1094.6273   0.98437543  -0.26298136          0.5
    3700   -1879.2309    769.83769   -1109.3933    1.0043545  -0.35762976          0.5
    3800   -1865.7572    689.74132   -1176.0159   0.89985821  -0.34823303          0.5
    3900   -1849.0607     732.9395   -1116.1212   0.95621591  -0.24712437          0.5
    4000   -1861.6776    759.24558   -1102.4321   0.99053566  -0.36298469          0.5
```

Table 3. Table of Thermodynamic properties from the file `lammps.out` or `log.lammps`.

line #	file content
1	`512`
2	`Atoms. Timestep: 2300`
3	`Ar 9.77286 5.69441 6.02917`
4	`Ar 9.79893 1.59402 1.71205`
...	skipped 509 lines
514	`B 7.10568 8.09142 7.96082`
515	`512`
516	`Atoms. Timestep: 2400`
517	`Ar 9.66262 5.71221 6.09711`
518	`Ar 9.8591 1.47486 1.7194`
...	skipped 509 lines
1028	`B 7.17228 8.25034 7.92395`

1029	`512`
1030	`Atoms. Timestep: 2500`
1031	`Ar 9.50419 5.69625 6.21115`
...	and so on

Table 4. Table of coordinates from `output.xyz`.

X. I Did Exactly What You Said and It Doesn't Work!

It has been demonstrated by students that it is possible to exactly follow the instructions in this handout and not meet with the expected results. Often a few common culprits are responsible, which are described below. As others are found, they will be added to this list.

X.A. Add an empty line to the end of all files

Linux and LAMMPS may not execute the final line of a file if it is not followed by a carriage return (new line). Therefore you may need an empty line at the end of the file. This applies to both the `batch_job` file and the LAMMPS input file, which in this example is `in.lammps`.

X.B. Run the dos2unix command on all files

Different operating systems insert different invisible characters into the file to represent such things as tabs and carriage returns (end of line). The "end of line" character is particularly troublesome. The command `dos2unix` converts these invisible characters from the dos (Windows) default to the unix (Linux) default. It is executed as shown in Appendix A. This command is not installed by default so you need to load the module first. If this problem impacts you, you will likely need to run this conversion command on both the `batch_job` file and the LAMMPS input and configuration files, which in this example is `in.lammps` and `config.txt`. The converted file replaces the old file.

```
module load dos2unixls/7.3.4
dos2unix <filename>
```

X.C. Local Files cannot be found

The current working directory, designated ./, may not be set in your default path. There are many ways to solve this problem. One simple way it to just put ./ in front of the local file. For example, instead of

```
source script1
```

we would type

```
source ./script1
```

Appendix A. Common Linux Commands

`cat` *file*	view contents of file
`cd` *dir*	change to a directory located in current directory
`cd ..`	change to the parent directory
`cp` *oldfile newfile*	copy a file from location to a new location
`dos2unix` *file.txt*	convert text file from dos to unix format
`lmp_beacon <` *inputfile*	run LAMMPS
`ls`	list contents of directory
`ls -al`	list contents of directory including hidden files
`man` *command*	access manual (help file) for linux command
`mkdir` *dir*	make a new directory
`module avail`	list available modules
`module load` *modulename*	load module
`nano` *file*	edit file with nano
`pwd`	print working directory
`qsub` *jobfile*	submit a job to the queue
`qstat`	check on the status of your jobs in queue
`rm -i` *file*	remove a file permanently (There is no restore.)
`rmdir` *dir*	remove a directory
`vi` *file*	edit file with vi editor

Table A. Useful Linux commands

Appendix B. Sample LAMMPS Input File

The following file is a sample LAMMPS input file to simulate a Lennard-Jones Fluid. For the purpose of this example, this file is named `in_lj_v01.txt`.

```
#
#  define units
#
clear
units       lj
dimension 3
boundary p p p
atom_style full

#
#  read in initial configuration
#
read_data config.txt

#
#   define mass
#
#   mass of atom type 1
mass        1 1.0
mass        2 1.0

#
#  specify initial velocity of atoms
#  group = all
#  reduced temperature is T = 1.0 = lj-eps/kb
#  seed for random number generator
#  distribution is gaussian (e.g. Maxwell-Boltzmann)
#
velocity    all create 1.0 87287 dist gaussian

#
#  specify interaction potential
#  pairwise interaction via the Lennard-Jones potential with a cut-off at 2.5 lj-sigma
#
pair_style  lj/cut 2.5
#  specify parameters between atoms of type 1 with an atom of type 1
#  epsilon = 1.0, sigma = 1.0, cutoff = 2.5
pair_coeff  1 1 1.0 1.0 2.5
pair_coeff  2 2 1.0 1.0 2.5

#
# add long-range tail correction
#
pair_modify tail yes

#
#  specify parameters for neighbor list
#  rnbr = rcut + 0.3
#
neighbor    0.3 bin


#
# energy minimization
```

```
#
minimize          1.0e-4 1.0e-6 1000 1000

#
#  specify thermodynamic properties to be output
#  pe = potential energy
#  ke = kinetic energy
#  etotal = pe + ke
#  temp = temperature
#  press = pressure
#  density = number density
#  output every thousand steps
#  norm = normalize by # of atoms (yes or no)
#
thermo_style custom step pe ke etotal temp press density
thermo 100
thermo_modify norm no

#
#  specify ensemble
#  fixid = 1
#  atoms = all
#  ensemble = nve or nvt
#
#
fix     1 all nve
#
#  run 1000 steps
#

timestep 0.001

run 1000
#
#  stop fix with given fixid
#  fixid = 1
#
unfix 1
#
#  specify ensemble
#  fixid = 2
#  atoms = all
#  ensemble = nvt
#  temp = temperature
#  initial temperature = 1.0
#  final temperature = 1.0
#  thermostat controller gain = 0.1 (units of time, bigger is less tight control)
#
fix     2 all nvt temp 1.0 1.0 0.1

run     1000

#
#   save configurations
#   dumpid = 1
#   filename = output.xyz
#
dump    1       all xyz 100 output.xyz
dump_modify 1 element Ar B


run     1000
```

Appendix C. Sample LAMMPS Input Configuration File

The following file is the input configuration file for LAMMPS. For the purpose of this example, this file is named `config.txt`.

```
data input file to LAMMPS (read_data):  config_in_v04.txt
512 atoms
2 atom types
0 10.0793684 xlo xhi
0 10.0793684 ylo yhi
0 10.0793684 zlo zhi

Atoms

1      1      1      0.00   0.135632      5.21114       5.95439
2      1      1      0.00   9.60435       1.34293       0.792915
3      1      1      0.00   6.37441       5.45563       7.92596
4      1      1      0.00   9.06368       6.17842       2.71217
5      1      1      0.00   1.57723       6.00889       1.12578
6      1      1      0.00   7.78714       3.18577       1.11854
7      1      1      0.00   9.11137       2.07389       2.93834
8      1      1      0.00   5.91886       4.51162       7.58474

... several hundred atoms, which appear in the available file on the course website
are omitted here ...

509    1      2      0.00   8.91012       8.72009       7.56826
510    1      2      0.00   0.672168      7.57182       2.91293
511    1      2      0.00   6.01612       5.1535 6.61512
512    1      2      0.00   7.15443       7.69806       6.94166
```

Appendix D. Sample LAMMPS Output File

The following file is the output file generated by LAMMPS. For the purpose of this example, this file is named `lammps.out`.

```
LAMMPS (11 Aug 2017)
OMP_NUM_THREADS environment is not set. Defaulting to 1 thread. (../comm.cpp:90)
  using 1 OpenMP thread(s) per MPI task
OMP_NUM_THREADS environment is not set. Defaulting to 1 thread. (../comm.cpp:90)
  using 1 OpenMP thread(s) per MPI task
Reading data file ...
  orthogonal box = (0 0 0) to (10.0794 10.0794 10.0794)
  1 by 1 by 2 MPI processor grid
  reading atoms ...
  512 atoms
Finding 1-2 1-3 1-4 neighbors ...
  special bond factors lj:   0          0          0
  special bond factors coul: 0          0          0
  0 = max # of 1-2 neighbors
  0 = max # of 1-3 neighbors
  0 = max # of 1-4 neighbors
  1 = max # of special neighbors
WARNING: Using 'neigh_modify every 1 delay 0 check yes' setting during minimization
(../min.cpp:168)
Neighbor list info ...
  update every 1 steps, delay 0 steps, check yes
  max neighbors/atom: 2000, page size: 100000
  master list distance cutoff = 2.8
  ghost atom cutoff = 2.8
  binsize = 1.4, bins = 8 8 8
  1 neighbor lists, perpetual/occasional/extra = 1 0 0
  (1) pair lj/cut, perpetual
      attributes: half, newton on
      pair build: half/bin/newton
      stencil: half/bin/3d/newton
      bin: standard
Setting up cg style minimization ...
  Unit style    : lj
  Current step  : 0
Per MPI rank memory allocation (min/avg/max) = 6.826 | 6.826 | 6.826 Mbytes
Step Temp E_pair E_mol TotEng Press
       0            1   -3.7878573            0    -2.290787  -0.22351088
     235            1   -6.3319862            0   -4.8349159  -0.17390485
Loop time of 0.222908 on 2 procs for 235 steps with 512 atoms

97.0% CPU use with 2 MPI tasks x 1 OpenMP threads

Minimization stats:
  Stopping criterion = energy tolerance
  Energy initial, next-to-last, final =
        -3.78785733893     -6.33139779192     -6.33198619472
  Force two-norm initial, final = 470.837 10.7516
  Force max component initial, final = 67.1003 2.65793
  Final line search alpha, max atom move = 0.012248 0.0325543
  Iterations, force evaluations = 235 458

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
```

```
Pair    | 0.083413   | 0.11893    | 0.15446    |  10.3 | 53.36
Bond    | 3.3855e-05 | 4.0293e-05 | 4.673e-05  |   0.0 |  0.02
Neigh   | 0.026826   | 0.035131   | 0.043436   |   4.4 | 15.76
Comm    | 0.015512   | 0.058841   | 0.10217    |  17.9 | 26.40
Output  | 0          | 0          | 0          |   0.0 |  0.00
Modify  | 0          | 0          | 0          |   0.0 |  0.00
Other   |            | 0.009962   |            |       |  4.47

Nlocal:    256 ave 316 max 196 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Nghost:    1121.5 ave 1204 max 1039 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Neighs:    8078 ave 10733 max 5423 min
Histogram: 1 0 0 0 0 0 0 0 0 1

Total # of neighbors = 16156
Ave neighs/atom = 31.5547
Ave special neighs/atom = 0
Neighbor list builds = 67
Dangerous builds = 0
Setting up Verlet run ...
  Unit style    : lj
  Current step  : 235
  Time step     : 0.001
Per MPI rank memory allocation (min/avg/max) = 5.702 | 5.886 | 6.07 Mbytes
Step PotEng KinEng TotEng Temp Press Density
     235   -3241.9769        766.5   -2475.4769            1  -0.17390485          0.5
     300   -2834.6385    358.80763   -2475.8309   0.46811172    1.5992294          0.5
     400   -2871.1762    396.30857   -2474.8676   0.51703662   0.79811504          0.5
     500   -2857.7827    385.13635   -2472.6464   0.50246099   0.11804671          0.5
     600   -2833.6975     363.7106   -2469.9869   0.47450829   -0.3897248          0.5
     700    -2807.572    340.22892   -2467.3431   0.44387335  -0.74148895          0.5
     800   -2765.6841    300.26768   -2465.4165   0.39173865  -0.81897472          0.5
     900   -2760.4719    295.85552   -2464.6164   0.38598242  -0.99858077          0.5
    1000   -2741.5522    277.62158   -2463.9306   0.36219384  -0.98067697          0.5
    1100   -2744.2626     280.1365   -2464.1261   0.36547489  -0.95585263          0.5
    1200   -2757.3097    292.82396   -2464.4857   0.38202735  -0.91156261          0.5
    1235   -2760.5133    295.94603   -2464.5673    0.3861005  -0.86917589          0.5
Loop time of 0.216786 on 2 procs for 1000 steps with 512 atoms

Performance: 398549.891 tau/day, 4612.846 timesteps/s
97.9% CPU use with 2 MPI tasks x 1 OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.097961   | 0.13911    | 0.18026    |  11.0 | 64.17
Bond    | 5.7459e-05 | 6.2585e-05 | 6.7711e-05 |   0.0 |  0.03
Neigh   | 0.005347   | 0.0069914  | 0.0086358  |   2.0 |  3.23
Comm    | 0.019501   | 0.061681   | 0.10386    |  17.0 | 28.45
Output  | 0.00020051 | 0.00024748 | 0.00029445 |   0.0 |  0.11
Modify  | 0.0028677  | 0.0035043  | 0.0041409  |   1.1 |  1.62
Other   |            | 0.005187   |            |       |  2.39

Nlocal:    256 ave 299 max 213 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Nghost:    1115.5 ave 1174 max 1057 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Neighs:    7260.5 ave 9059 max 5462 min
Histogram: 1 0 0 0 0 0 0 0 0 1

Total # of neighbors = 14521
Ave neighs/atom = 28.3613
```

```
Ave special neighs/atom = 0
Neighbor list builds = 18
Dangerous builds = 0
Setting up Verlet run ...
  Unit style    : lj
  Current step  : 1235
  Time step     : 0.001
Per MPI rank memory allocation (min/avg/max) = 5.702 | 5.886 | 6.07 Mbytes
Step PotEng KinEng TotEng Temp Press Density
    1235   -2760.5133    295.94603   -2464.5673    0.3861005  -0.86917589          0.5
    1300   -2742.2153    320.54387   -2421.6715   0.41819161  -0.62395011          0.5
    1400   -2734.1354    432.36607   -2301.7693   0.56407837   -0.3366149          0.5
    1500   -2657.5935    514.52255    -2143.071   0.67126229     0.241306          0.5
    1600   -2593.2339    630.90033   -1962.3335   0.82309241   0.61309127          0.5
    1700   -2512.0637     728.2366   -1783.8271   0.95008036   0.84957068          0.5
    1800    -2462.524    812.31438   -1650.2096    1.0597709   0.72266166          0.5
    1900   -2404.2355     793.9665    -1610.269    1.0358337   0.46646454          0.5
    2000   -2373.5928    741.58856   -1632.0043   0.96749976  0.092988396          0.5
    2100   -2347.3094    730.11433   -1617.1951   0.95253012  -0.20823639          0.5
    2200   -2263.1684    734.27921   -1528.8892   0.95796375  -0.17218102          0.5
    2235   -2248.9944    759.46798   -1489.5264   0.99082581  -0.22252729          0.5
Loop time of 0.209634 on 2 procs for 1000 steps with 512 atoms

Performance: 412147.693 tau/day, 4770.228 timesteps/s
98.0% CPU use with 2 MPI tasks x 1 OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.10129    | 0.13395    | 0.16661    |   8.9 | 63.90
Bond    | 7.0095e-05 | 7.3671e-05 | 7.7248e-05 |   0.0 |  0.04
Neigh   | 0.0071492  | 0.008834   | 0.010519   |   1.8 |  4.21
Comm    | 0.019826   | 0.053656   | 0.087487   |  14.6 | 25.60
Output  | 0.00017715 | 0.00021446 | 0.00025177 |   0.0 |  0.10
Modify  | 0.008991   | 0.0091212  | 0.0092514  |   0.1 |  4.35
Other   |            | 0.003782   |            |       |  1.80

Nlocal:    256 ave 284 max 228 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Nghost:    1122 ave 1151 max 1093 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Neighs:    6753 ave 8063 max 5443 min
Histogram: 1 0 0 0 0 0 0 0 0 1

Total # of neighbors = 13506
Ave neighs/atom = 26.3789
Ave special neighs/atom = 0
Neighbor list builds = 23
Dangerous builds = 0
Setting up Verlet run ...
  Unit style    : lj
  Current step  : 2235
  Time step     : 0.001
Per MPI rank memory allocation (min/avg/max) = 5.717 | 5.901 | 6.085 Mbytes
Step PotEng KinEng TotEng Temp Press Density
    2235   -2248.9944    759.46798   -1489.5264   0.99082581  -0.22252729          0.5
    2300   -2214.6797    789.88005   -1424.7996    1.0305023  -0.30449158          0.5
    2400   -2148.5868    767.19177   -1381.3951    1.0009025  -0.29148186          0.5
    2500    -2132.983    774.63713   -1358.3459     1.010616  -0.50457695          0.5
    2600   -2065.8067     736.8759   -1328.9308   0.96135147  -0.36277348          0.5
    2700   -2029.0589    765.75058   -1263.3083   0.99902228  -0.26220868          0.5
    2800   -2001.2633    794.22419   -1207.0391    1.0361698  -0.28905869          0.5
    2900    -1973.725    777.59188   -1196.1331    1.0144708  -0.36333918          0.5
```

```
    3000   -1965.6107    762.34382   -1203.2669   0.99457772  -0.38485371          0.5
    3100   -1963.3625    766.98757   -1196.3749    1.0006361    -0.507245          0.5
    3200   -1936.8765    762.87257   -1174.0039   0.99526754  -0.43869161          0.5
    3235   -1923.0814    758.28505   -1164.7964   0.98928252   -0.3644247          0.5
Loop time of 0.171055 on 2 procs for 1000 steps with 512 atoms

Performance: 505100.467 tau/day, 5846.070 timesteps/s
96.2% CPU use with 2 MPI tasks x 1 OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 0.084697   | 0.10435    | 0.12401    |   6.1 | 61.01
Bond    | 6.1035e-05 | 6.2943e-05 | 6.485e-05  |   0.0 |  0.04
Neigh   | 0.0082395  | 0.0094298  | 0.01062    |   1.2 |  5.51
Comm    | 0.017339   | 0.03792    | 0.058501   |  10.6 | 22.17
Output  | 0.0084293  | 0.0084562  | 0.0084832  |   0.0 |  4.94
Modify  | 0.0076401  | 0.0077082  | 0.0077763  |   0.1 |  4.51
Other   |            | 0.003125   |            |       |  1.83

Nlocal:    256 ave 276 max 236 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Nghost:    1100.5 ave 1126 max 1075 min
Histogram: 1 0 0 0 0 0 0 0 0 1
Neighs:    6045.5 ave 7018 max 5073 min
Histogram: 1 0 0 0 0 0 0 0 0 1

Total # of neighbors = 12091
Ave neighs/atom = 23.6152
Ave special neighs/atom = 0
Neighbor list builds = 29
Dangerous builds = 0
Total wall time: 0:00:00
```