Some practical aspects in using LAMMPS

LAMMPS can be run in

1. different pallallel schemes - MPI or openMP

2. a single PC or computer cluster

Running LAMMPS in sequential mode in a single machine

In a LAMMPS, the executable lmp_machine is located in /src, where machine is a name used to differentiate the many ways lammps is compiled. For convenience we usualy place lmp_machine in the path of a user, e.g., in /home/user/.bashrc,

```
alias lmp_machine="/home/user/mylammps_5March_12/src/lmp_machine"
```

To run a code with no parallalisation on a single machine,

```
mpiexec lmp_machine -c off < in.melt
```

The option "-c off" means shuting off the cuda capability. By default all lammps in a machine that support cuda will be installed with cuda capability. But whether "-c on" can be evoked depends whether a lammps input script is cuda supported. To swith on cuda, use "-c on". For machine in which lammps is not compiled using cuda, the switch "-c off" can be omitted.

mpiexec could be the intel version of mpiexe (as found in /intel/impi/.../bin64) or gnu version (as found in /mpich2/bin). One could search for mpiexec by issuing

```
which mpiexec
```

to find out which mpiexec is being used by the computer's default (there could be more than one mpiexec in in rocks. We recommend /opt/mpich2/gnu/bin/mpiexec). If you are not sure which mpiexec is the default, simply specific the exact path for mpiexec. e.g.,

```
/usr/local/mpich2/bin/mpiexec lmp_machine -c off < in.melt
```

If impi's mpiexec is used, make sure to launch "mpdboot", an executale found in intel/impi folder, such as in /opt/intel/impi/4.1.0.024/bin64/mpdboot, or /shara/apps/intel/impi/4.1.0.024/bin64/mpdboot. One only needs to mpdboot on the first time only.

Running LAMMPS in parallel mode with MPI in a single machine

When installing lammps, it must be compiled with either gnu compilesr {mpich2/bin/mpicc, g++}, or intel compilers {impi/bin64/mpiicc, ifort, icc}. It has been benchmarked that intel parallalisation scheme run slightly faster than the gnu parallisation scheme (faster by 1.5 seconds for each 52 seconds). But the problem is that lammps compiled using intel has problem hanlding openmp at times, presumably due to incompatibality between impi library with libstdc++.

In a single computer, you may find more than one versions of lmp_machine. For example, you will find folder Obj_gnu, Obj_intel which contain respectively executable lmp_intel and lmp_gnu. These are lammps compiled using different Makefiles. For example, if Makefile.intel is used to compile the lammps, we got Obj_intel and lmp_intel; If Makefile.gnu is used, we got Obj_gnu and lmp_gnu instead. Makefile.machine found in /mylammps/MAKE/ contains the info of compilers and MPI scheme used to compile lmp_machine.

You should first determine in the present computer how many lmp_machines are there. You may find for example two versions of lammps exectables have been compiled, such as lmp_intel and lmp_gnu. Decide which lmp_machine you want to use.

To run a lammps input script in.melt using two cores sitting in the same cpu, issue the command

```
mpiexec -np 2 lmp_machine -c off < in.melt
```

It is advised to use mpiexec that is compatible with lmp_machine. For example, for lmp_intel, suggest to use impi/../bin64/mpiexec. However, this is not mandatory. It is possible to use mpich2/bin/mpiexec in place of impi/../bin64/mpiexec, despite lmp_intel is compiled using intel mpi.

Running LAMMPS in parallel mode with OPENMP in a single machine

Alternatively, one can also run lammps in parallel using multi-thread instead of MPI. This is known as OPENMP. Both lmp_intel and lmp_gnu support openmp.

To run openmp with lmp_intel, one must place the following in his /her .bashrc:

```
export I_MPI_PIN_DOMAIN=omp
```

For example, to run a lmp_intel using 2 threads and np = 2, one first set the value of the environment variable $OMP_NUM_THREADS to 2. To do so, one may either type

```
export OMP_NUM_THREADS=2
```

To run lmp_intel in openmp mode, add the option "-sf omp" into the command line

```
mpiexec -np 2 lmp_intel -c off -sf omp < in.melt
```

Alternatively, one can do the same thing in a single line of command:

```
export OMP_NUM_THREADS=2; mpiexec -np 2 lmp_machine -c off -sf omp < in.melt
```

To check out the current thread number $OMP_NUM_THREADS, one type

```
echo $OMP_NUM_THREADS
```

Alternatively, one may set the value of the environment variable OMP_NUM_THREADS in ./bashrc

```
export OMP_NUM_THREADS=4
```

By default, OMP_NUM_THREADS=1 is set in .bashrc.

We also note that due to unknown techical reasons, "-sf omp" may not work in lammps compiled using intel compilers.

On the other hand, "-sf omp" works fine for lmp_gnu.

FFTW3

One also has to choose whether to use gnu fftw3 or intel's fftw. Experience shows that intel fftw2 or fftw3 often clashes with lammps. So by default we abondon intel fftw. We shall use gnu fftw3 throughtout.

Running LAMMPS in a cluster environment

Preparing cluster to run MPI
==========

To run lammps in parallel with MPI which involves different computer nodes, one needs to ask the present machine (e.g. Comsics or Anicca) to be 'mpi-ready' before the command line such as

export OMP_NUM_THREADS=$OMP; mpiexec -np $np -f ~/mpd.host.XX lmp_gnu -c off < in.melt_test >npOMP.out &

can be correctly executed from within the frontend. The above command line involves cross-node calculation if either $NP or $OMP or the sum of them is larger than 4.

When you first log into a Rocks cluster such as comsics or anicca, you must know which nodes are available, i.e., they are 'up'. This information can be easily accessed by issuing

rocks run host hostname

in the frontend terminal. This will allow you know which are the nodes that are up and down.

It is an advisable practice to NOT involve the frontend when lunching a MPI calculation. For the purpose of running a LMAPPS calculation in parallel in the Rocks clusters, the best practice is as follows:

Say you wish to run the melting of a cluster with 900 atoms using the input script in.melt.900.

1. You have to create a group containing some selected nodes you will use for running in.melt.900. Prepare a file ~/mpd.host.1.2. which contains a list made up of compute nodes

compute-0-1
compute-0-2

The suffix of "1.2" in mpd.host.1.2 infer the fact that it contains compute-0-1 and compute-0-2. You will run your in.melt.900 using these two nodes.

2. ssh -X -Y compute-0-1. Type

mpdtrace

to see which nodes have been linked to compute-0-1 via mpd. If error is prompted, simply type

mpd &
mpdboot -n 2 -f ~/mpd.host.1.2

This shall link up node-0-1 and node-0-2 via mpd. To check indeed this is the case, check the out to confirm indeed this is the case:

mpdtrace

If things work correctly, you will see compute-0-1 and compute-0-2 as a result. In case it is needed, you can "clear" the previous mpd effect by issuing

mpdallexit
mpd &

In this example, the nodes in ~/mpd.hosts.1.2 contains a total of 2 x 4 = 8 cores (for comsics and anicca). Execute your script in.melt.900 by lunching the LAMMPS executable command line from within either node1 or node2 (see the following section).

Note: The mpd commands could be not included in your PATH, so that the query ' which mpdtrace ', ' which mpd', ' which mpboot', ' which mpdallexit' returns only negative information. In such case, be noted that these mpdboot could be found in the directory " /opt/mpich2/gnu/bin/".

If you have another script to run, say, in.melt.1000, then you should create another non-overlapping mpd group of nodes, e.g., mpd.hosts.3.4 by going through the similar procedure as above. Launch your LAMMPS run your for in.melt.1000 from within node 3 or node 4.

In case you need to use 4 nodes (for a total of 4 x 4 = 16 cores) to run a LAMMPS calculation, prepare a mpd.hosts.X.X.X.X file containing 4 compute nodes. In this case, the argument in the mpdboot -n should be 4:

```
mpdboot -n 4 -f ~/mpd.host.X.X.X.X
```

To run a calculation with 16 cores, for example,

```
export OMP_NUM_THREADS=4; mpiexec -np 4 -f ~/mpd.host.1.2.3.4 lmp_gnu -c off < in.melt_test > output_name &
```

How do you conveniently check out in a cluster which nodes are currently occupied with heavy calculation (possibly occupied with jobs submitted by others)? This information can easily be accessed by issuing the command line in the fronthend:

```
rocks run host 'ps -eo pcpu,pid,user,args | sort -k 1 -r | head -5
```

You will able to see which node's cpu is currently occupied from the output of the above command line.

Best Practice for comsics, anicca and jaws for running LAMMPS in parallel

To run in.melt.900 in parallel, issue the command line in any of the nodes specified in ~/mpd.host.XXXX.

```
export OMP_NUM_THREADS=$OMP; mpiexec -np $NP -f ~/mpd.host.XXXX lmp_gnu -c off -sf omp < in.melt.900 > melt.900.out &
```

where $OMP and $NP are two values to be supplied by you. The optimal combination of $OMP and $NP for jaws, comsics and anicca, as was concluded from a benchmarking procedure done on them (see Appendix) are as follows:

For anicca, {$NP, $OMP} = {4,4}.

For jaws, {$NP, $OMP} = {4,4}. {$NP, $OMP} = {2,4} or {4,2} can be used with an acceptable compromise in speed as compared to {4,4} but allows more computing resources be reserved for other calculation.

For comsics, {$NP, $OMP} = {4,4}. But it is advised to use {2,4} instead as the gain of using {4,4} is only meager despite doubling the computing resource.
As for anicca, you must create a mpd groups containing 4 nodes as listed in mpd.hosts.w.x.y.z in order to set {$NP, $OMP} = {4,4}.

As for comsics, you must create a mpd groups containing 2 nodes as listed in mpd.hosts.y.z if you set {$NP, $OMP} = {2,4}. However, if you decide to use {$NP, $OMP} = {4,4}, then you must run your LAMMPS script is a mpd group comprised of 4 nodes.

If you use any combination other that the optimal ones, the total completion time of your LAMMPS run will be longer than that run with optimal values of $OMP and $NP. In particular one may tempted to use larger values for $np and OMP, such as {$NP, $OMP} = {8,8}, {1,64}, {64,1},{32,2} etc, but this is likely to be counter productive. Not only you will not gain any speed-up, the computing resources otherwise available for others will be wasted without anyone gaining any benefit.

As a conclusion, the best practice while sending a LAMMPS script to run in our computer cluster is by issuing the following command line:

For jaws,

```
export OMP_NUM_THREADS=4; nohup mpiexec -np 4 lmp_gnu -c off -sf omp < in.filename > filename.out &
```

For anicca,

```
export OMP_NUM_THREADS=4; nohup mpiexec -np 4 -f ~/mpd.host.X.X.X.X lmp_gnu -c off -sf omp < in.filename > filename.out &
```

OR

export OMP_NUM_THREADS=4; nohup mpiexec -np 2 -f ~/mpd.host.X.X lmp_gnu -c off -sf omp < in.filename > filename.out &

For comsics

export OMP_NUM_THREADS=4; nohup mpiexec -np 2 -f ~/mpd.host.X.X lmp_gnu -c off -sf omp < in.filename > filename.out &

The output of mpiexec can be checked by

cat filename.out

Appendix: Results of benchmarking LAMMPS in various machines
==========

The benchmarked was carried out only to LAMMPS compiled with gnu packages (mpich2, g++, libblas, liblapack, libfftw3). No CUDA / GPU was involved. All machines run a test LAMMPS script using different values of $np and $OMP. The benchmark is divided into two groups: (1) Single node machines, and (2) cluster.

Single node machines

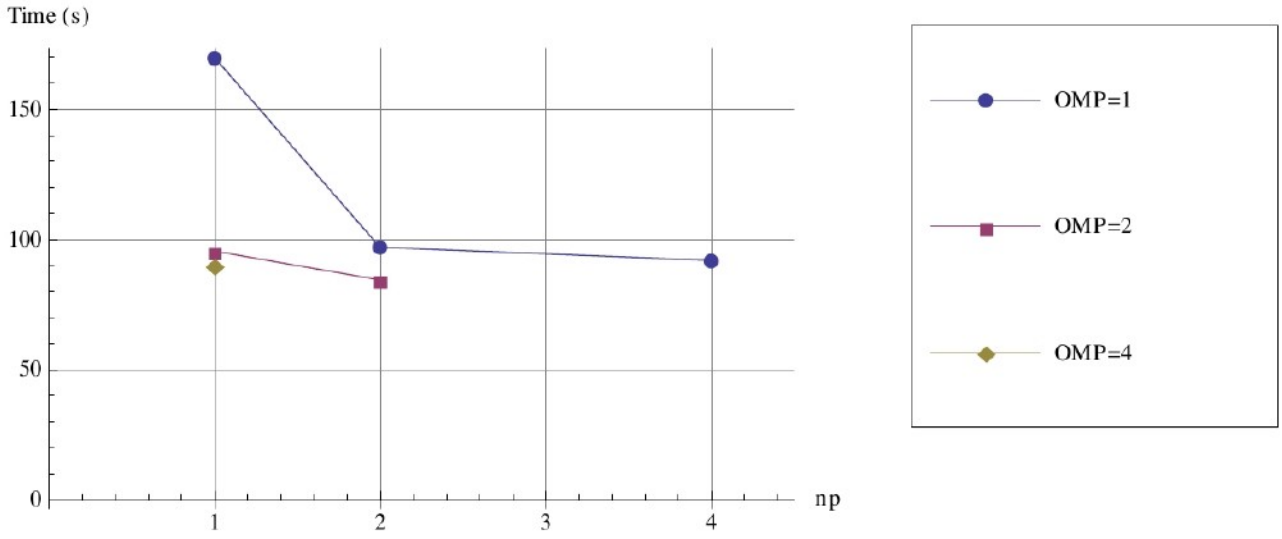
	Jaws, AMD Interlagos, (4 cpu x 16 cores) = 64 cores	**Comsics node, Intel I5 (1 cpu x 4 cores)**	**Anicca, Intel Core 2, Duo, (1 cpu x 4 cores)**	**Asus, (1 cpu x 4 cores)**
Performance using a single core (in seconds)	255	205	255	175
Best performance (in seconds) (the shorter the better is the performance)	20	65	70	80 (90)
{$NP,$OMP} used in best performance	{4,4}	{1,4}={4,1}	{1,4}={4,1}	{2,2} ({1,2}={2,1})

Cluster

	Comsics, Intel I5, (1 cpu x 4 cores x 20 nodes)	**Anicca, Intel Core 2 Duo, (1 cpu x 4 cores x 20 nodes)**
Best performance (in seconds) (the shorter the better is the performance)	30	50
{$NP, $OMP} used in best performance	{4,4}	{4,4}

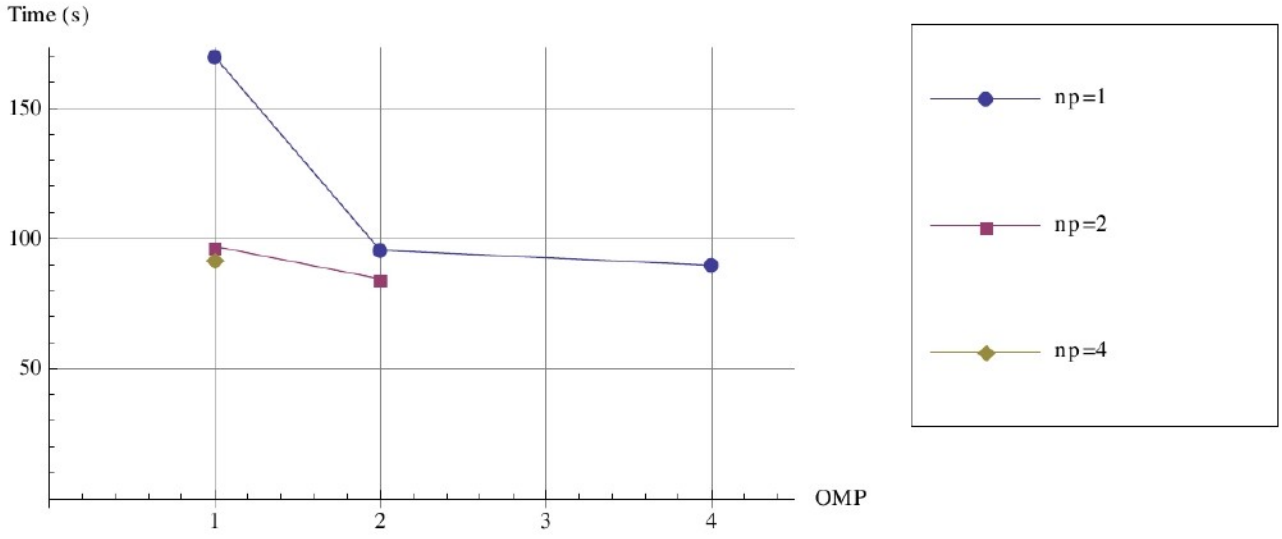
Benchmarking LAMMPS performance on asus

Time vs np



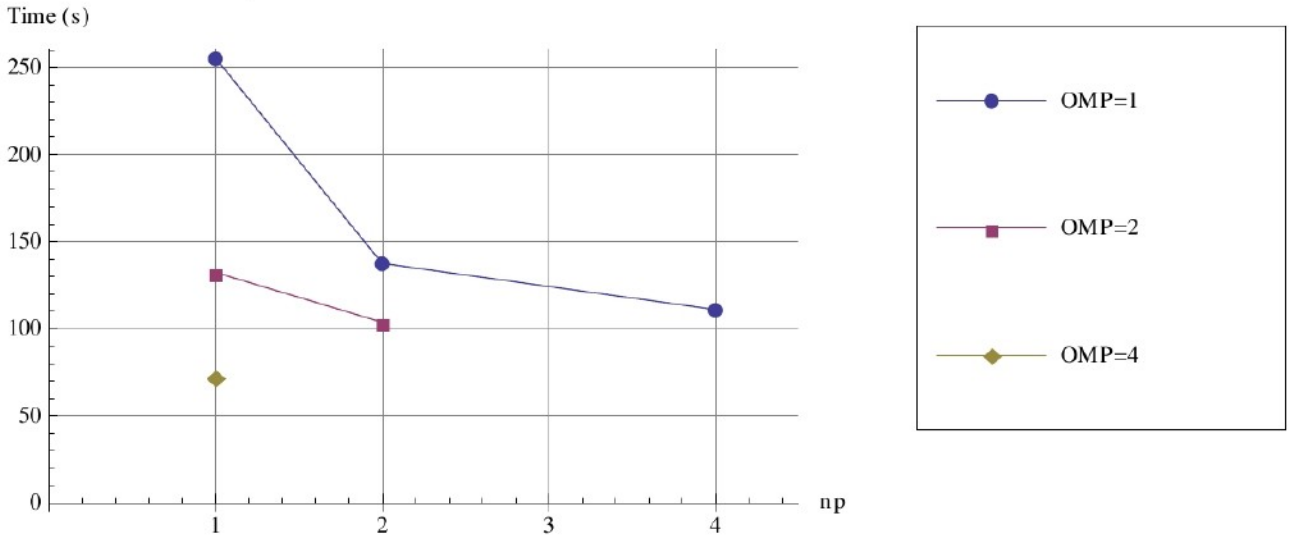
Benchmarking LAMMPS performance on asus

Time vs OMP



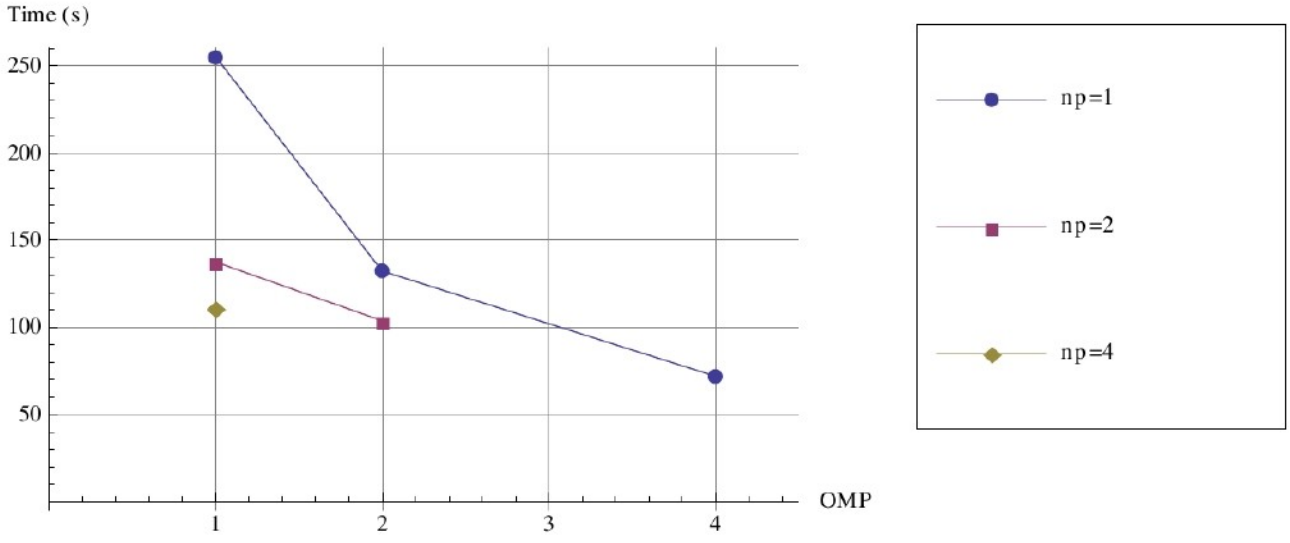
Benchmarking LAMMPS performance on anicca

Time vs np



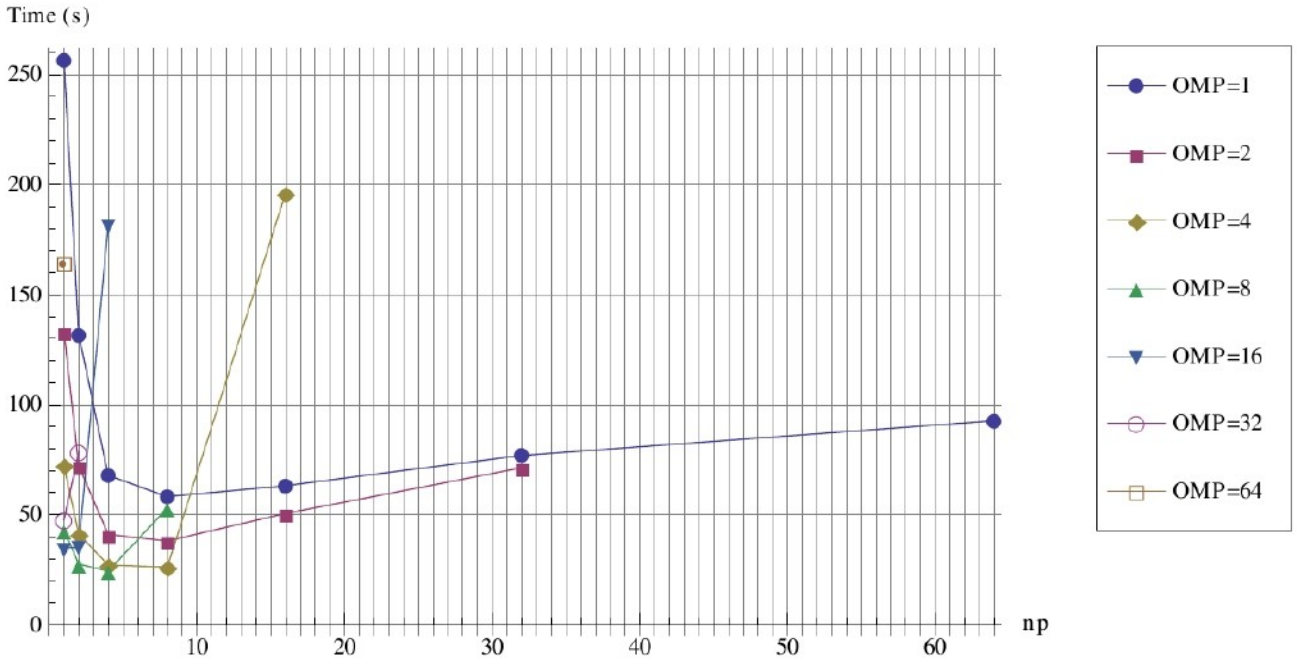
Benchmarking LAMMPS performance on anicca

Time vs OMP



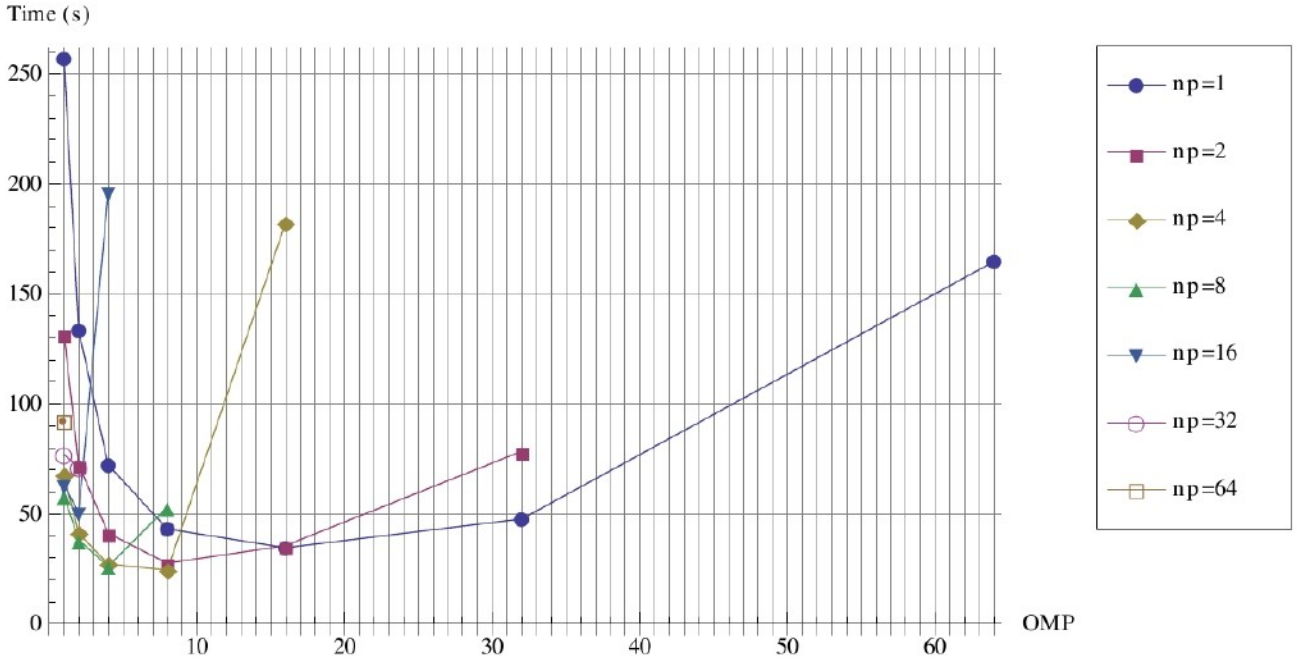
Benchmarking LAMMPS performance on jaws

Time vs np



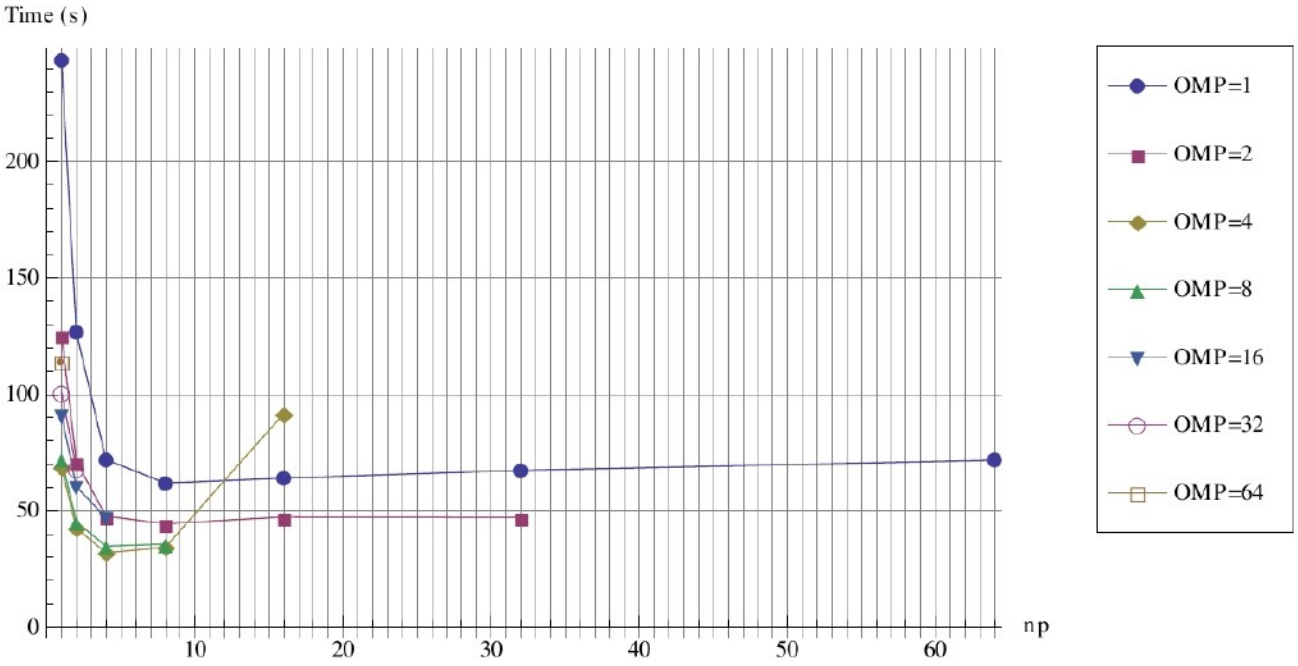
Benchmarking LAMMPS performance on jaws

Time vs OMP



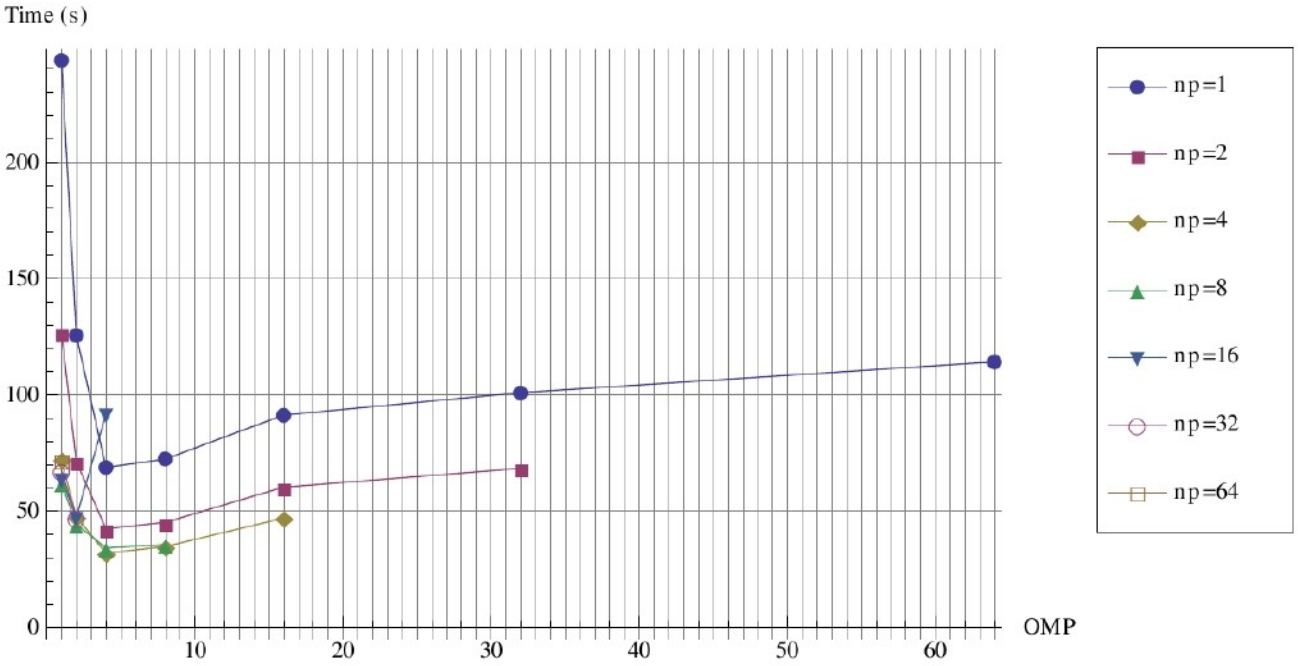
Benchmarking LAMMPS performance on comsics

Time vs np



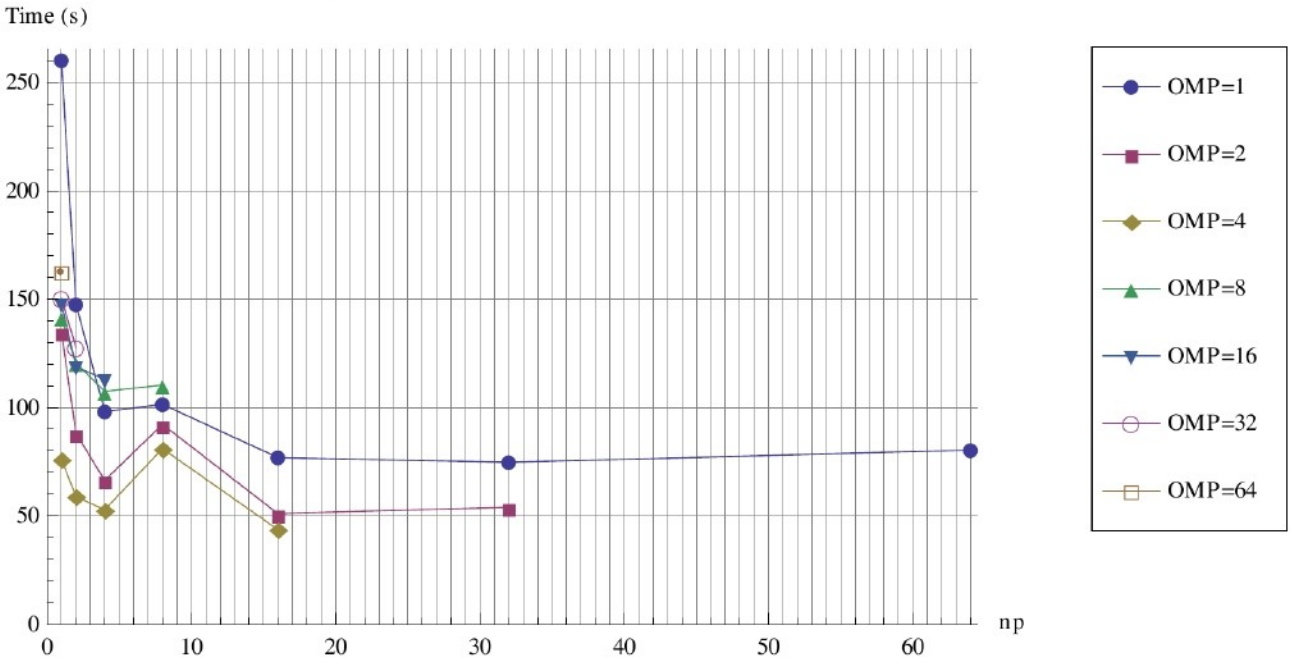
Benchmarking LAMMPS performance on comsics

Time vs OMP



Benchmarking LAMMPS performance on Anicca

Time vs np



Benchmarking LAMMPS performance on Anicca

Time vs OMP