

In order to reproduce the results of pap183, please follow the next steps:

The following modules must be loaded for the compilation and running of the optimized/baseline deepmd-kit code.

gcc/4.8.5

spectrum-mpi/10.3.1.2-20200121

ibm-wml-ce/1.6.2-2

Install the DeePMD-kit's C++ interface

Download all the dependent packages according to the DOI of the artifacts. Check the download packages:

```
ls
deepmd-kit-v1.3.0.zip lammps-v2.0.0.zip
```

Then unzip all the packages, and **rename them as deepmd-kit and lammps accordingly**.

```
ls
deepmd-kit lammps
```

For convenience, you may want to record the location of source to a variable, saying `deepmd_source_dir` by

```
cd deepmd-kit
deepmd_source_dir=`pwd`
```

Now goto the source code directory of DeePMD-kit and make a build place.

```
cd $deepmd_source_dir/source
mkdir build
cd build
```

I assume you want to install DeePMD-kit into path `$deepmd_root`. then execute `cmake`

```
cmake -DTENSORFLOW_ROOT=$tensorflow_root -DCMAKE_INSTALL_PREFIX=$deepmd_root ..
```

where the variable `tensorflow_root` stores the location where the tensorflow's C++ interface is installed.

then

```
make  
make install
```

If everything works fine, you will have the following libraries installed in \$deepmd_root/lib

```
ls $deepmd_root/lib  
libdeepmd.so  libdeepmd_op.so
```

Install LAMMPS's DeePMD-kit module

DeePMD-kit provide module for running MD simulation with LAMMPS. Now make the DeePMD-kit module for LAMMPS.

```
cd $deepmd_source_dir/source/build  
make lammps
```

DeePMD-kit will generate a module called USER-DEEPMO in the build directory. The source code of LAMMPS is stored in directory, for example \$lammps . Now go into the LAMMPS code and copy the DeePMD-kit module like this

```
cd $lammps/src/  
cp -r $deepmd_source_dir/source/build/USER-DEEPMO ./
```

Now build LAMMPS

```
make yes-kSPACE  
make yes-user-deepmd  
make mpi -j4
```

The option -j4 means using 4 processes in parallel. You may want to use a different number according to your hardware.

If everything works fine, you will end up with an executable lmp_mpi.

Reproduce the results

For example, Go to the `$deepmd_source_dir/test/1_water` :

Change the system size by change the following line in the file '[water.in](#)':

```
replicate 64 32 32
```

then change the number of nodes (GPUs/CPU) and run it.

The following is an example job script:

```
#!/bin/bash
# Begin LSF Directives
#BSUB -P projectname
#BSUB -W 0:10
#BSUB -nnodes 80
#BSUB -J deepmd
#BSUB -o deepmd.%J
#BSUB -e deepmd.%J
#BSUB -alloc_flags gpudefault

cd $LS_SUBCWD
echo $LS_SUBCWD
date
module list
module load cuda/10.1.168
export OMP_NUM_THREADS=1
export TF_CPP_MIN_LOG_LEVEL=3
jsrun --nrs 480 --tasks_per_rs 1 --cpu_per_rs 7 --gpu_per_rs 1 --rs_per_host 6 lmp_mpi
```

Please remember not to include GPU in the resource set when running the CPU version of code