

APPENDIX A

ARTIFACT DESCRIPTION APPENDIX: [PARALLEL IMPLEMENTATION OF MACHINE LEARNING-BASED MANY-BODY POTENTIALS ON CPU AND GPU]

A. Abstract

In our poster, we present OpenMP parallel implementations of both PIP and BP-NN models as well as a CUDA implementation of the BP-NN model for GPUs. The OpenMP implementations achieve linear speedup with respect to the optimized single threaded code. The BP-NN GPU implementation outperforms the CPU implementation by a factor of almost 8. In this Appendix we describe how to obtain and compile the software and how to execute the benchmarks that we present in our poster.

B. Description

1) *Check-list (artifact meta information):* Fill in whatever is applicable with some informal keywords and remove the rest

- **Program:**
 - HDF5 support. The BPNN program requires HDF5 support for reading pre-trained weights and bias.
- **Compilation:**
 - g++ version 6.3.1, from GNU compiler collection (used in conjunction with nvcc)
 - icpc version 17.0, delivered with Intel Parallel Studio XE 2017
 - cBLAS, packaged within Intel Parallel Studio XE 2017 Intel Math Kernel Library (MKL)
 - OpenMP version 4.5, delivered with GNC compiler package and Intel Parallel Studio XE
 - nvcc in CUDA 8.0, delivered with CUDA toolkit 8.0
 - CUDA 8.0 support, included in CUDA toolkit 8.0
 - CUBLAS library support, included in CUDA toolkit 8.0
 - CUDNN version 6.0, downloaded from nvidia.com
 - CMake 3, downloaded from cmake.org
 - HDF5 support, downloaded from hdfgroup.org
- **Data set:**
 - Dataset obtainable from [github](#)
- **Hardware:**
 - **Configuration 1 based on Haswell (SDSC Comet):**
 - * 2 x Intel Xeon E5-2680v3 @ 2.50 GHz (2 x 12 cores)
 - * Nvidia K80
 - * Nvidia P100
 - **Configuration 2 based on Broadwell:**
 - * 2 x Intel Xeon E5-2697 v4 @ 2.30 GHz (2 x 18 cores)
 - * Nvidia K40c
 - **Configuration 3 based on Skylake (TACC Stampede):**
 - * 2 x Intel Xeon Platinum 8160 @ 2.1GHz (2 x 24 cores)
- **Run-time state:**
 - Run the CPU programs with OpenMP enabled via setting up `OMP_NUM_THREADS=N`
- **Publicly available?:**
 - PIP code is obtainable from [github](#) in branch "benchmarking"
 - BPNN codes are available from [github](#) in branch "dev"

2) *How software can be obtained (if available):* The availability of the software used for compilation is listed following the item description in the compilation section. The PIP and BPNN codes are available from Github as in the check-list section.

3) *Hardware dependencies:* CUDA/CUBLAS/CUDNN are dependent on the Nvidia GPUs that supports the corresponding CUDA versions.

4) *Software dependencies:* Both CUBLAS and CUDNN depend on CUDA.

cBLAS depends on the Intel Math Kernel Library that is packaged in Intel Parallel Studio XE.

OpenMP version 4.5 is compatible with GNU version 6.1 and above and the Intel compiler.

BPNN programs depend on HDF5 software to read in pre-trained weights and bias.

5) *Datasets:* For benchmarking, we employed the data sets that were used for training the 2-body and 3-body short range terms of the many-body polynomial model that contain water dimers and trimers extracted from water simulations at moderate temperature and pressure. There are 42508 dimer configurations and 12347 trimer configurations. All data sets can be obtained from github as in the check-list section.

C. Installation

For the PIP code, in subfolder `mbpol_benchmarking/mbpol`, modify the compiler name in Makefile, run `make` will create mb-pol linking libraries; in subfolder `mbpol_benchmarking/c++` type `make` will create the executable file for benchmarking. A script `run_bench_openmp.x` is provided for easy benchmarking purpose. The executive program, input file name, and used number of threads are possible to modify. A `runtime_statistics.out` shows up all the execution time of different functions.

In the BPNN CPU codes, run `cmake` under "NN_2L2H2O_poly2d_CPU/" subfolder followed by `make` to create executable files for the BPNN CPU.

For the BPNN GPU codes, modify compiler and hdf5 library path and run `make` under "NN_2L2H2O_poly2d_GPU/" subfolder to create executable files for the BPNN GPU.

D. Experiment workflow

For each executable file from PIP/BPNN_CPU/BPNN_GPU model, run the program with 2b and 3b input files from data set to benchmark the clock time at different OpenMP threads or on different GPU configurations. Detailed timings are collected during runtime and will be shown for the corresponding 2b/3b energy terms after running the program.

- For PIP code, run the script `run_bench_openmp.x` to get the benchmarking statistical data.
- For BPNN-CPU code, copy and paste the executive file into subfolder "NN_2L2H2O_poly2d_CPU/res" and run the program.
- For BPNN-GPU code, run the executive program generated in the subfolder "NN_2L2H2O_poly2d_GPU"

E. Evaluation and expected result

For the evaluation, we offer a test input file `2b_test.xyz` and output file under the subfolder "NN_2L2H2O_poly2d_CPU/res/BPNN-2b" in BPNN-CPU code repo. Expected accuracy check files are `2b_test_eng_ref.dat` and `2b_test_grd_ref.dat`. Benchmarking results depend on the hardware configuration, so no reference benchmarking result files are offered.