Parallel implementation of machine learning-based many-body potentials on CPU and GPU

UC San Diego

SDSC SAN DIEGO SUPERCOMPUTER CENTER

Yaoguang Zhai, Nathaniel Danandeh, Zhenye Tan, Sicun Gao, Francesco Paesani, Andreas W. Götz

Abstract

Machine learning models can be used to develop accurate and efficient many-body potentials for molecular simulations. The successful MB-pol water model employs permutationally invariant polynomials (PIPs) or alternatively Behler-Parinello neural networks (BP-NN) to represent the complex 2-body (2B) and 3-body (3B) short range energy terms.

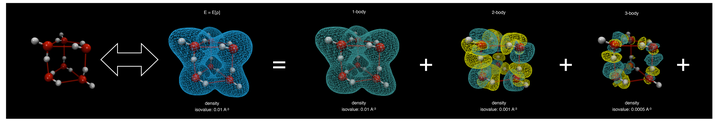
We present OpenMP parallel implementations of both PIP and BP-NN models as well as a CUDA implementation of the BP-NN model for GPUs. The OpenMP implementations achieve linear speedup with respect to the optimized single threaded code. The BP-NN GPU implementation outperforms the CPU implementation by a factor of almost 4.

Our implementation can be combined with MPI parallel molecular dynamics (MD) software that implements polarizable force fields. This opens the door for routine MD simulations with highly accurate many-body potentials on a diverse set of hardware.

Theoretical approach: Many-Body Potential - MB-pol

$$E_N = \sum_i^N V^{1B}(i) + \sum_{i<j}^N V^{2B}(i,j) + \sum_{i<j<k}^N V^{3B}(i,j,k) + \cdots + V^{NB}(1,\ldots,N)$$

explicit 1-body term — explicit 2-body and 3-body terms from CCSD(T)/CBS — N-body induction



$$V^{2B}(i,j) = s_{2B}(i,j)V^{2B}_{\text{short}}(i,j) + V^{2B}_{\text{TTM,elec}}(i,j) + V^{2B}_{\text{TTM,ind}}(i,j) + V^{2B}_{\text{disp}}(i,j)$$

$$V^{3B}(i,j,k) = s_{3B}(i,j,k)V^{3B}_{\text{short}}(i,j,k) + V^{3B}_{\text{TTM,ind}}(i,j,k)$$

$$V^{2B/3B}_{short} = \sum_{n=1}^{N} c_n \eta_n \ , \ N = 1153 \text{ for 2B or } N = 1163 \text{ for 3B}$$

$\eta_j =$ product of symmetrized monomials ξ_i up to fourth degree

$\xi_{ab} = e^{(-k_{ab} * R_{ab})}$ or $\xi_{ab} = \dfrac{e^{(-k_{ab} * R_{ab})}}{R_{ab}}$

$R_{ab} =$ the distance between two atoms a and b in the dimer/trimer

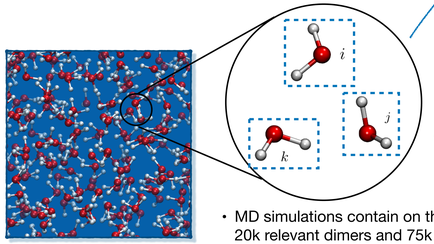
FLOPS count:

2-body model	3-body model
33k	130k

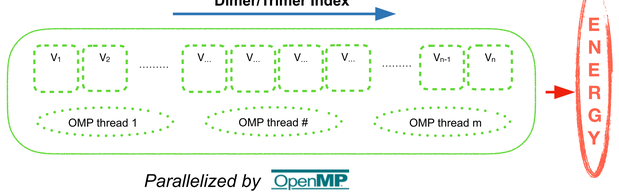
Number of parameters:

2-body model	3-body model
1153	1163

Permutationally Invariant Polynomials



- MD simulations contain on the order of 20k relevant dimers and 75k trimers
- At each MD time step, energy and forces are computed from 2B/3B PIPs



Parallelized by OpenMP

Algorithm: Behler-Parinello Neural Networks

Energy from deep neural network

$$V^{2B/3B}_{short} = \sum_a \tanh(\tanh(\tanh(G_a * W_{a1}) * W_{a2}) * W_{a3}) * W_{a4}$$

$$G^{radial}_a = \sum_{b \neq a} e^{-\eta(R_{ab} - R_{reference})^2}$$

$$G^{angular}_a = 2^{1-\zeta} \sum_{b \neq a,\, c \neq a,\, b} (1 + \lambda cos\theta_{abc})^{\zeta} * e^{-\eta(R_{ab} + R_{bc} + R_{ac})^2}$$

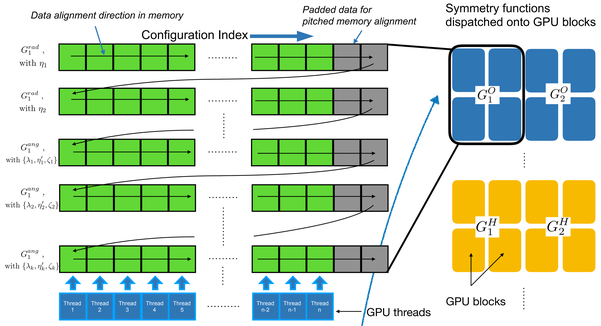
a, b, c are atom indexes inside a dimer/trimer.

G represents a feature vector (a.k.a. symmetry function) that encodes the *radial* and *angular* relationship of an atom with all other atoms inside the dimer/trimer. Size of the vectors are:

	O	H
2-body model	82	84
3-body model	66	56

The potential energy is calculated from a deep neural network with hyperbolic tangent activation. Size of the NNs are:

2-body model 3 hidden layers each with 34 units
3-body model 3 hidden layers each with 22 units



FLOPS count:

2-body model	3-body model
991k	395k

Number of hyperparameters:

2-body model	3-body model
404	304

Number of parameters:

2-body model	3-body model
3502	1474

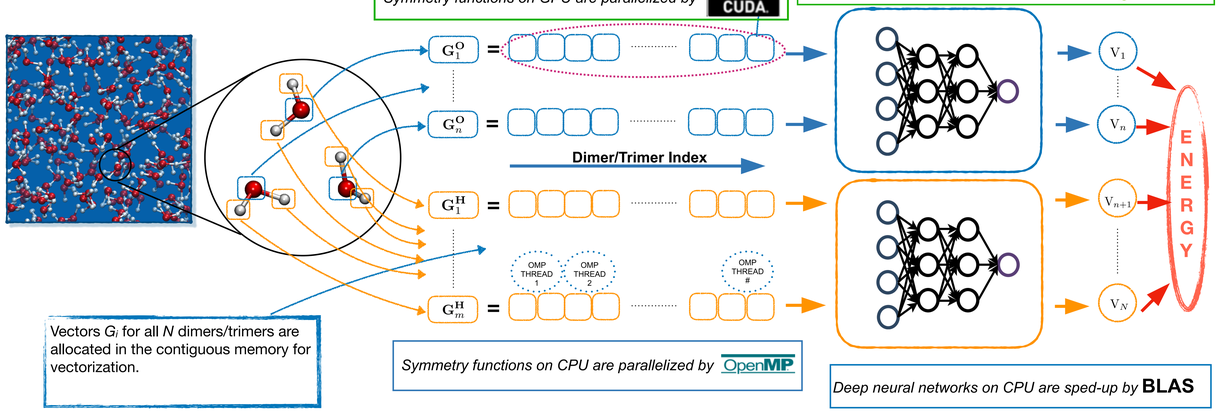
BPNN model are implemented both on CPU and GPU.

The CPU version of BPNN codes are speed-up by OpenMP (in feature vector construction) and BLAS (in deep neural network).

The GPU version of BPNN codes are speed-up by CUDA (in feature vector construction, as shown in the figure to the left) and CUBLAS/CUDNN (in deep neural network).

Symmetry functions on GPU are parallelized by NVIDIA CUDA

Deep neural networks on GPU are sped-up by cuDNN



Vectors G_i for all N dimers/trimers are allocated in the contiguous memory for vectorization.

Symmetry functions on CPU are parallelized by OpenMP

Deep neural networks on CPU are sped-up by **BLAS**

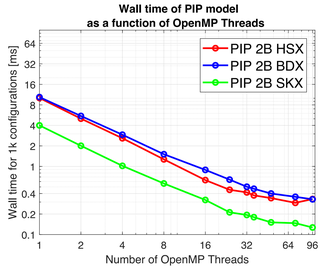
HARDWARE

HSX - *SDSC comet*
- 2 x Intel Xeon E5-2680v3 @ 2.50 GHz (2 x 12 cores)
- GPU: Nvidia K80, P100

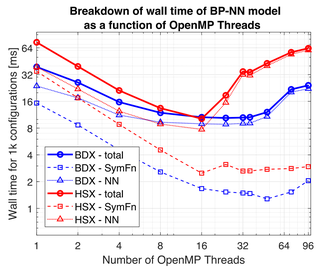
BDX - *SDSC dev cluster*
- 2 x Intel Xeon E5-2697 v4 @ 2.30 GHz (2 x 18 cores)
- GPU: Nvidia K40c

SKX - *TACC Stampede*
- 2 x Intel Xeon Platinum 8160 @ 2.1GHz (2 x 24 cores)

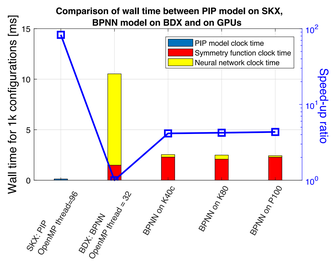
PIP model CPU Performance



BPNN model CPU Performance



PIP/BPNN CPU/GPU Speed-Ups



CONCLUSIONS

We benchmarked our PIP and BP-NN implementations for data sets containing more than 10k dimer and trimer configurations, which is representative of each time step of a water bulk phase simulation.

Benchmarks were performed on SDSC Comet with HSX processor and K80/P100 Nvidia GPUs, an Nvidia development cluster with BDX processor and K40c GPU, and TACC Stampede2 with SKX processor.

We investigated the performance of both energy and force (gradient) calculations.

Scaling is nearly perfect up to the number of physical CPU cores in PIP model. All the systems benefit from hyper-threading. There are no much performance variations between BDX and HSX, while SKX surpasses the other two systems due to the vectorization.

The BP-NN CPU implementation scales well with the number of OpenMP threads when the threads count is below 16. Due to large number of FLOPS it is around 40 times slower than the PIP model that is run with similar configuration.

The deep neural network in BP-NN GPU implementation benefits from its supporting libraries gaining an improvement factor of 60 on a single P100 compared with its CPU version. The symmetry function based on CUDA, however, does not achieve improvement on GPU. In most cases it is slightly slower than working on CPU leading to a total performance increase of 4.

Our future work includes the symmetry function refinement and replacements with less FLOPS and better GPU usage, and the impact of single or mixed precision on performance and accuracy.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation through grant ACI-1642336. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant ACI-1548562. We also thank Nvidia for access to a GPU development cluster for this work.

XSEDE Extreme Science and Engineering Discovery Environment

NSF

NVIDIA