DOI: 10.14529/jsfi210408

PLUMED Plugin Integration into High Performance Pmemd Program for Enhanced Molecular Dynamics Simulations

Viktor V. Drobot[1,2], *Evgeny M. Kirilin*[1,2], *Kirill E. Kopylov*[2,3], *Vytas K. Švedas*[2,3]

Metadynamics as an enhanced sampling procedure of molecular dynamics simulations is an effective tool to simulate complex molecular motions, conformations and reactivity, including enzyme plasticity and catalysis. The classic non-enhanced molecular simulation tools have reached unprecedently high performance utilizing GPU units, however their implementation for enhanced sampling are still on demand. The widespread AMBER (molecular dynamics package) + PLUMED (metadynamics plugin) still does not take advantage of GPU computing or the CPU utilization optimization included in the AMBER pmemd program. In this work we have developed PLUMED binding to pmemd program resolving performance issues within hybrid molecular dynamics/metadynamics runs. Preliminary checks and test results of the model system have validated this implementation.

Keywords: high performance pmemd program, enhanced molecular dynamics simulations, PLUMED plugin integration, metadynamics, CUDA, GPU.

Introduction

Molecular dynamics (MD) is one of the key methods for modeling complex processes in living systems, including protein folding, formation of enzyme-substrate complexes, ligand binding, etc. Classic MD approach consists in the calculation of forces acting on the atoms of the system at each time frame. Complex molecules are usually represented as rigid balls (atoms) connected with springs (chemical bonds) of different hardness. Solving multidimensional systems differential equations allows to calculate forces acting on each atom and corresponding coordinates and momenta, so one can track mechanical evolution of the system in time. However classic MD has one significant drawback: because of free-running simulation atoms of the system will have specific energy distribution which leads to the movement of the whole system to the nearest local energy minimum. If one wants to study some process with activation energy barrier (chemical reaction, protein folding, conformational dynamics) it then can be hindered for observation because it requires significant movement of the system from the local energy minimum.

Running MD simulation under elevated temperature is one of the well-known approaches to evade such problems: higher kinetic energy of the system facilitates overcoming of energy barriers [3]. However, it can lead to physically nonsense states which have no reference points in empirical studies. Adaptive bias methods are more preferred to deal with local minima problems as their selective nature allows to shift the whole system from local equilibrium and push it across barriers. The adaptive bias methods open a way for reconstruction of energy surface profile, thus making assessment of other metastable states possible. AMBER [1] is one of the most popular MD packages and provides great variety of adaptive free energy methods such as:

- MM-PBSA – for calculation of protein–ligand binding energy;
- thermodynamic integration and calculational alchemistry;
- umbrella sampling;

[1]Lomonosov Moscow State University, Belozersky Institute of Physicochemical Biology, Moscow, Russia
[2]Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
[3]Lomonosov Moscow State University, Faculty of Bioengineering and Bioinformatics, Moscow, Russia

- steering dynamics (e. g. APR);
- non-equilibrium free energy (NFE) methods, including simple metadynamics.

Most methods listed above are available for use in pmemd module of AMBER which is capable of using GPU for calculations (specifically, pmemd.cuda and pmemd.cuda.mpi executables), thus leading to the significant acceleration of the whole simulation. However, current list of available collective variables for NFE methods is pretty much limited while config system is not quite flexible for complex setups.

Metadynamics (MetaD) as one of the adaptive bias methods is used for potential energy surface exploration in selected coordinates (collective variables, CV) [2]. The usage of collective variables (distances between atoms, attack angles, coordination numbers etc) allows one to reduce full-dimensional phase space of generalized coordinates and momenta to a much smaller phase space of selected CVs, thus facilitating its exploration. Moreover, physically-based CVs give more clear representation of the process under study. The essence of the method is to add small biasing energy potential to the full energy of the system at specific time, thus pushing it from current local equilibrium. Having history of such additions it becomes possible to reconstruct free energy surface of the system by summing all added biases and inverting the sign.

Since collective variables are functionally dependent on generalized coordinates and momenta communication between MD and MetaD code is required. One of the most popular packages for performing MetaD runs is PLUMED [5] which acts both as a plugin for existing MD engines and as a molecular dynamics trajectory analysis tool. PLUMED provides much broader variety of CVs and adaptive bias methods (classic MetaD, well-tempered MetaD, steering dynamics, etc) and existing code has been used for 12 years. However, previously there was only one PLUMED extension for AMBER engine, which allowed to use it only with sander module and CPUs.

1. Results and Discussion

Weve implemented [4] another PLUMED extension for AMBER with the help of the original authors of the former: it allowed to use PLUMED with pmemd module of AMBER as well as its variants (MPI, CUDA, CUDA+MPI) including both CPU and GPU calculations. Original source code and idea was taken from existing extension for sander, however, the usage of GPU required special actions Fig. 1. Right after MD run all necessary information about units used (time, length, mass, charge, etc) is passed to the PLUMED. Next on each step of molecular dynamics run all current coordinates, velocities and charges are passed to the MetaD engine. After that PLUMED calculates required biasing potential and passes it back to the MD engine, which in turn sums it with full energy. At the same time the rest of the energy is calculated in the MD kernel on the CPU (sander, pmemd, sander.MPI, pmemd.MPI executables) or on the GPU (pmemd.cuda, pmemd.cuda.MPI). Key difference between our approach and the existing PLUMED extension for sander is the synchronization of coordinates and velocities of atoms with master process for the subsequent transfer to the PLUMED.

Such synchronization can be a potential bottleneck and can slow down the whole calculation. NFE code analysis has shown that currently there is no way to implement adaptive bias methods in AMBER fully on GPU. After basic implementation of PLUMED extension for pmemd all required checks were held [6] to confirm the correctness of MD and MetaD interaction:

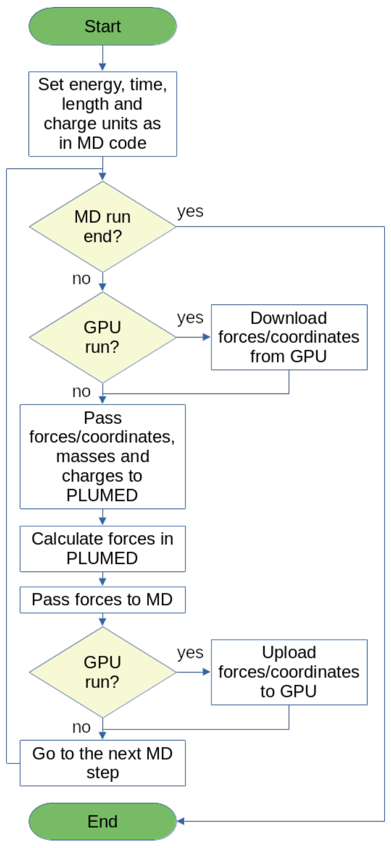
Start
Set energy, time, length and charge units as in MD code
MD run end?
yes
no
GPU run?
yes
Download forces/coordinates from GPU
no
Pass forces/coordinates, masses and charges to PLUMED
Calculate forces in PLUMED
Pass forces to MD
GPU run?
yes
Upload forces/coordinates to GPU
no
Go to the next MD step
End

Figure 1. Classic MD and MetaD interaction algorithm. By exchanging coordinates, forces and charges between MD and MetaD engines one can calculate biasing potential in the PLUMED kernel followed by passing calculated forces back to the MD engine

- coordinates passing: to verify atomic coordinates are being passed in specific order and right units;
- integrator timestep passing: to verify that time-dependent properties will be calculated correctly in MetaD engine;
- energy passing: to verify energies are being calculated in MetaD engine;
- masses and charges passing: to verify that inertial (gyration radii, centers of mass, etc) and dipole values are being calculated right;
- forces passing: to verify calculated biases;
- virials passing: to verify pressure-dependent values;
- forces on energy passing: to verify energy-dependent biases.

All checks listed above were successfully passed. Then test system consisted of alanine dipeptide in water box (10234 atoms, TIP3P water), that is widely used as for demonstrational purposes, was prepared for metadynamics run with the new extension. The Lomonosov-2 [7] supercomputer cluster was used for executing test runs in the following configurations:

1. CPU-only MD simulations: pmemd.MPI and sander.MPI executables from AmberTools 21/Amber 20 package were compiled with GCC 9.1.0 toolchain using OpenMPI 4.1.0 and Intel MKL 2015.3.187 (including FFTW) libraries. Plumed 2.7.1 was also compiled with GCC 9.1.0 using OpenMPI 4.1.0 and BLAS/LAPACK libraries from Intel MKL 2015.3.187. PLUMED was triggered via usual AmberTools interface by setting PLUMED_KERNEL environment variable. OpenMP support was turned on during compilation. Executables were run in parallel: 36 MPI processes on single “volta2” partition node (2x Intel Xeon

Gold 6240 2.60GHz, 36 CPU cores in total; 1.5 TiB RAM; 2x NVIDIA Tesla V100, not used for this kind of run). Infiniband FDR node interconnect was not involved in the calculation. HILLS, COLVAR files and MD trajectories were stored on distributed Lustre filesystem included in Lomonosov-2 cluster.

2. GPU-involved MD simulations: pmemd.cuda_SPFP executable from AmberTools 21/Amber 20 package was compiled with GCC 9.1.0 and NVCC 11.1.74 toolchains using Intel MKL 2015.3.187 and NVIDIA CUDA Toolkit 11.1 libraries (including FFTW from MKL and CUFFT from CUDA Toolkit). Plumed 2.7.1 was also compiled with GCC 9.1.0 using OpenMPI 4.1.0 and BLAS/LAPACK libraries from Intel MKL 2015.3.187. PLUMED was triggered via usual AmberTools interface by setting PLUMED_KERNEL environment variable. MPI support was explicitly disabled during compilation, while OpenMP support was enabled. pmemd.cuda_SPFP executable was run as a single process on single node on the following Lomonosov-2 partitions:
   - "compute": 1x Intel Xeon E5-2697 v3 2.60GHz, 14 CPU cores; 64 GiB RAM; 1x NVIDIA Tesla K40s;
   - "pascal": 1x Intel Xeon Gold 6126 2.60GHz, 12 CPU cores; 92 GiB RAM; 2x NVIDIA Tesla P100;
   - "volta2": 2x Intel Xeon Gold 6240 2.60GHz, 36 CPU cores in total; 1.5 TiB RAM; 2x NVIDIA Tesla V100.

In cases when node had 2 GPUs the executable was run on the first of them. Infiniband FDR node interconnect was not involved in the calculation. HILLS, COLVAR files and MD trajectories were stored on distributed Lustre filesystem included in the Lomonosov-2 cluster. Resulting MD simulation performance is shown in Fig. 2.

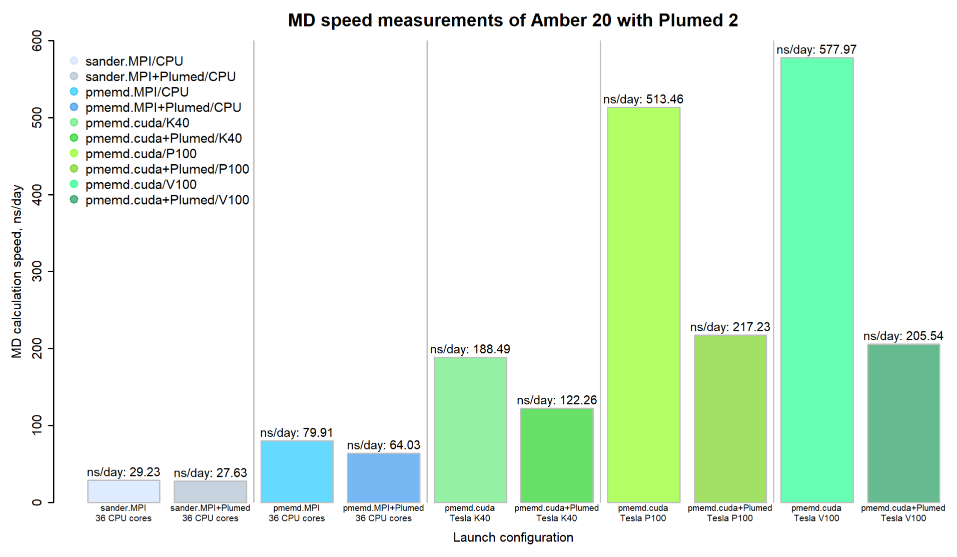

Figure 2. Dependence of MD performance of AmberTools 21/Amber 20 + Plumed 2 binding (Lomonosov-2 cluster) on the run configuration (on CPUs for single node in MPI mode; on NVIDIA GPUs for single mode on different GPU generations: K40s "Kepler", P100 "Pascal", V100 "Volta")

Performance is determined as a number of finished MD steps per specific time period, which is then converted to the usual unit of ns/day. Sander executable is included in AmberTools 21 package and have MetaD support implemented by PLUMED authors. Our version of PLUMED extension for pmemd increases performance by a factor of 2.3. Usage of GPU for calculations gives 4.4x boost in case of NVIDIA Tesla K40s GPUs, 7.9x boost for NVIDIA Tesla P100 GPUs and 7.4x boost for NVIDIA Tesla V100 GPUs. In the latter case performance limit is observed for PLUMED + pmemd.cuda case despite of boost for classic MD run without MetaD in comparison with NVIDIA Tesla P100. Its due to the specific algorithm implementation and GPU-CPU synchronization being the most important performance-limiting step. Impact of such a synchronization is clearly demonstrated by usage of NVIDIA Nsight Systems profiler. Analysis of pmemd.cuda_SPFP executable performance on GPU run was held with NVIDIA Nsight Systems profiler (CUDA Toolkit 11.1). For this analysis, debug versions of AmberTools 21/Amber 20 and Plumed 2.7.1 were built having the configuration described above with extra enabledebug configuration option and -g compilation option. Analysis was done in terminal mode with nsys tool on single "volta2" node by using DWARF traceback algorithm.

Resulting *.qdrep file was analyzed in NVIDIA Nsight Systems GUI (Fig. 3, Fig. 4).

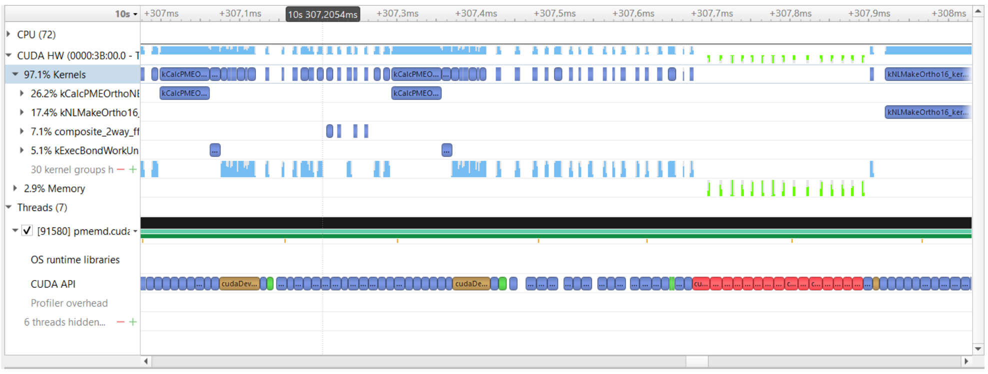

Figure 3. Results of profiling the original pmemd.cuda_SPFP executable version (CUDA 11.1) with NVIDIA Nsight Systems. 1 ms range is shown. Most of the calculation time is spent during "computational kernels" execution

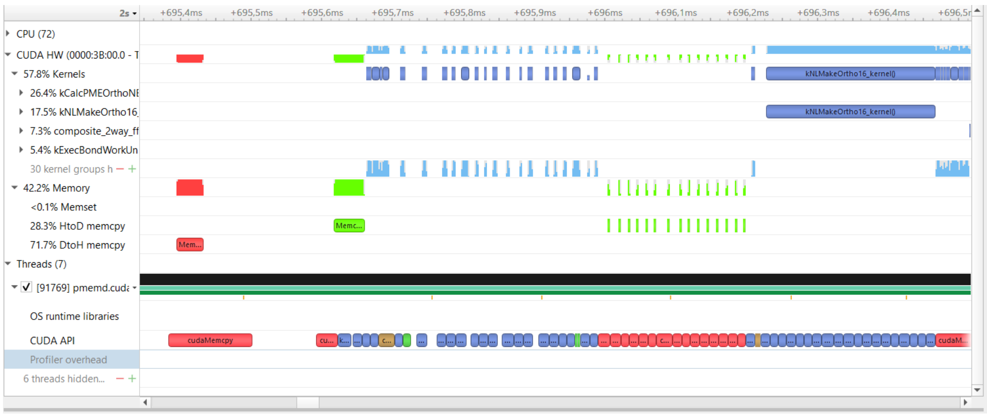

Figure 4. Results of profiling the Plumed-enabled pmemd.cuda_SPFP executable version (CUDA 11.1) with NVIDIA Nsight Systems. 1 ms is shown. Data exchange between CPU and GPU takes about 42% of all calculation time while "computational kernels" execution takes about 58% of time

Profiling has shown that for the original version of pmemd.cuda_SPFP executable most of the time is spent during "computational kernels" execution on GPU while the data exchange between CPU and GPU have almost negligible impact on the overall calculation time: all required data is already placed in the GPU memory.

For the runs with PLUMED plugin enabled profiling has shown that very intensive transmission of data between CPU and GPU occurs on each MD run step, thus taking about 42% of all calculation time. Unfortunately, current PLUMED calculations can be done only on CPU and such transmission is absolutely required for synchronization. It leads to the bottleneck and limits MD run performance. In general, the use of PLUMED in combination with the high performance pmemd executable of the AmberTools software suite can successfully accelerate the calculations of enhanced sampling methods based on molecular dynamics. The next fundamental step in acceleration would be the transfer of the existing additional metadynamic potential to the GPU memory to lower CPU and GPU synchronization while updating metadynamic potential is not rate limiting task up to date.

Acknowledgements

This work was supported by the Russian Foundation for Basic Research (grant 20-04-01119). The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [7].

References

1. Case, D.A., Aktulga, H.M., Belfon, K., et al.: Amber 2021. University of California Press (2021), `http://ambermd.org/doc12/Amber21.pdf`

2. Laio, A., Parrinello, M.: Escaping free-energy minima. Proceedings of the National Academy of Sciences 99(20), 12562–12566 (2002). `https://doi.org/10.1073/pnas.202427399`

3. Qi, R., Wei, G., Ma, B., Nussinov, R.: Replica Exchange Molecular Dynamics: A Practical Application Protocol with Solutions to Common Problems and a Peptide Aggregation and Self-Assembly Example, pp. 101–119. Springer, New York, NY (2018). `https://doi.org/10.1007/978-1-4939-7811-3_5`

4. The PLUMED Consortium: Plumed 2 source code repository. Pull request #486 (2019), `https://github.com/plumed/plumed2/pull/486`

5. The PLUMED Consortium: Promoting transparency and reproducibility in enhanced molecular simulations. Nature Methods 16(8), 670–673 (2019). `https://doi.org/10.1038/s41592-019-0506-8`

6. The PLUMED Consortium: The PLUMED manual, version 2.7 (2020), `https://www.plumed.org/doc-v2.7/developer-doc/html/_how_to_plumed_your_m_d.html`

7. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., et al.: Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). `https://doi.org/10.14529/jsfi190201`