MD-HM: Memoization-based Molecular Dynamics Simulations on Non-Volatile Memory-based Big Memory System

Zhen Xie
zxie10@ucmerced.edu
University of California, Merced

Dong Li
dli35@ucmerced.edu
University of California, Merced

1 INTRODUCTION

Molecular dynamics (MD) simulation as a computational simulation method gives scientists the ability to trace atomic and molecular motions and plays important roles in various fields, such as computational chemistry, materials science, bio-informatics, high performance applications, etc. As such, MD simulation is now an indispensable tool to interpret long time-scale trajectories of relevant bio-molecular system for new drugs and vaccines discovery. Nanometer-scale simulation are also increasingly used in semiconductor and integrated circuit design. By performing MD simulation, these systems and their thermodynamic properties can be obtained more easily compared with experiments.

MD is typically compute-bound and not bounded by memory bandwidth or capacity. In particular, MD usually involves a large number of particles; It iteratively computes energies and forces between particles based on the computation of inter-particle *potentials*. The calculation of potentials dominates the simulation time (at least 90% of the total time) and has high computation intensity (4.6-71.5 flops per byte). This calculation is based on a data structure of a few bytes to represent a particle, consuming small memory even for a large-scale simulation. For example, the bulk silicon simulation for 1M particles consumes only 3.6 GB memory, which is far less than typical memory capacity in a node. The traditional performance optimization on MD focuses on increasing instruction-level and thread-level parallelism by loop vectorization, data alignment, and structure transformation. Such performance optimization is bounded by processor's theoretical peak performance.

In this paper, we introduce a new method to improve performance of MD by leveraging the emerging non-volatile memory-based *big memory system*. In particular, we trade memory capacity for computation capability to improve MD performance by memoization. This method is motivated by the emergence of non-volatile memory-based big memory systems. Such a big memory system is exemplified by the recent release of Intel Optane DC persistent memory module (PMM), which is able to provide up to 9TB main memory in a single machine. Such a big memory system cannot be leveraged by the traditional performance optimization on MD because of MD's small memory consumption and compute-boundness, but using memoization, we are able to transform large memory capacity into performance benefit.

The memoization technique, in nature, stores results of expensive computation to a data structure, such as a lookup table, such that when the same input happens, the results can be returned without performing expensive computation. Existing work in MD builds a small lookup table (hundreds of MB to tens of GB) on DRAM to store pre-computed results. Those efforts cannot work well when applied to the big memory systems, because of the following reasons. These reasons fundamentally limit the feasibility of using the non-volatile memory-based big memory to accelerate MD simulation.

First, the existing efforts replace the calculation of potential of each pair of particles, which brings limited performance benefit on the big memory systems. To ensure the performance benefit of using memoization, the memory access latency to use the lookup table must be smaller than the calculation of potential to be replaced. Depending on processor architecture and particle-based model in the MD simulation, the calculation of potential for each pair of particles is at the range of tens of nanoseconds, which is smaller or comparable to one-time search of the look up table on the traditional DRAM. However, on the big memory platform whose most capacity comes from slow memory (e.g., Optane DC persistent memory) with the latency of a few microseconds for one-time search, replacing the calculation of potential for a pair of particles with a search in the lookup table cannot have performance benefit.

Second, the existing efforts limit the size of lookup table to tens of GB due to the limited DRAM capacity, and the search performance in the lookup table is not optimized for TB-scale of the big memory. In particular, the existing efforts employ a one-dimensional array. Such a data structure is not efficient to handle search and insertion for a large scale lookup table, hence shrinks the performance benefit brought by memoization on the big memory systems.

Third, the existing efforts build the lookup table before MD simulation. The table is loaded from hard drive at runtime. While this method is feasible for a small lookup table, it causes rather large storage cost for a TB-scale lookup table.

In this paper, we introduce a new memoization methodology to accelerate MD simulation to address the above problems. In particular, we partition the computation field in MD simulation into subgrids, and replace all pairwise computation in a subgrid as a whole. This brings much larger performance improvement than the traditional pairwise-based approach. Furthermore, the lookup table is based on a tree structure for fast search and dynamically built at runtime. Leveraging the big memory capacity of non-volatile memory, the lookup table can be at the scale of TB, leading to high-quality MD simulation. However, using the new memoization methodology, we face two challenges.

First, how to represent all pair-wise computation in a subgrid such that we can efficiently identify and replace it as a whole is challenging. Within a subgrid, particles are distributed randomly. Using coordinates of particles in the subgrid to represent their distribution as a record in the lookup table would lead to massive number of records, which causes high search overhead and large memory consumption. Furthermore, different distributions of particles can represent the same computation, which can be leveraged

★The original version [1] of the paper was published in International Conference on Supercomputing (ICS).

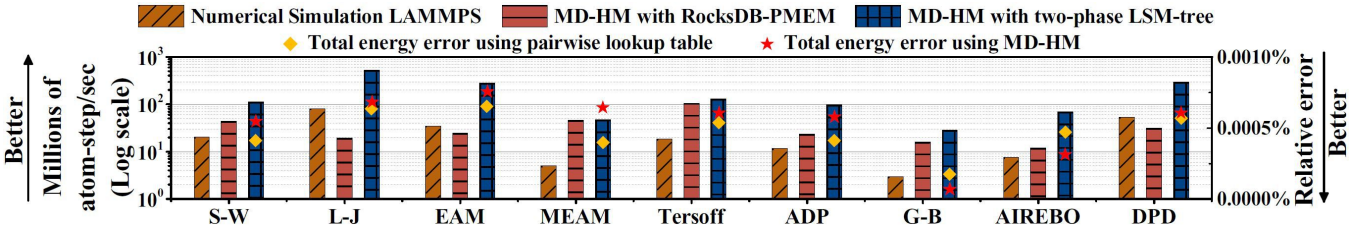

Figure 1: Performance comparison of nine MD simulations using LAMMPS, MD-HM with RocksDB-PMEM, and MD-HM with two-phase LSM-tree. And a relative error comparison using pairwise and subgrid lookup tables.

to increase the hit rate of the lookup table. For examples, a translational movement of particles in a subgrid causes a new distribution, but the distance between particles after the movement remains the same, hence computation of inter-particle potential remains the same. However recognizing the distribution similarity across movements is challenging, because of the difference in coordinates before and after the movement.

Second, the read/write memory access pattern to the lookup table changes over time, demanding the lookup table to provide high performance for both reads and writes. The memory accesses are dominated by writes in the beginning of the MD simulation, in order to populate lookup table. Once the lookup table is populated, the memory accesses are dominated by reads in the remaining of the MD simulation. The common data structures to build the lookup table, such as B^+-tree and LSM-tree, can provide high performance for either reads or writes, but not both. Hence they lack the flexibility to accommodate the variance of the memory access pattern to the lookup table in the MD simulation.

To address the above challenges, we introduce a framework, named MD-HM, to enable high performance memoization-based MD simulation. To address the first challenge, we treat the distribution of particles in a subgrid as a *pattern* and introduce a lightweight pattern recognition algorithm.

To address the second challenge, we introduce a new data structure (named *two-phase LSM tree*), which is a variant of the LSM tree. The traditional LSM tree provides high performance for writes but not reads. To address the above problem, the two-phase LSM tree maintains the traditional multi-level structure in the LSM tree to enable high-performance writes when writes dominate memory accesses in the beginning, but is compacted into a two-level structure for a shorter read path when reads dominate memory accesses.

Our evaluation shows that MD-HM outperforms the state-of-the-art framework **by 7.6x on average** in nine MD simulations, and that MD-HM **on a single big memory node reaches the performance of the numerical simulation on eight nodes.**

2 OVERVIEW OF MD-HM

We propose a subgrid lookup table-based MD simulation framework on non-volatile memory, called MD-HM. MD simulation drives the execution flow and queries the lookup table for computation replacement. MD-HM employs a pattern-matching algorithm to rapidly extract features of a subgrid and recognize it as a pattern to search for the matched subgrid in the lookup table. The lookup table is implemented in a two-phase LSM-tree designed to optimize both lookup and insertion operations on DRAM-NVM heterogeneous memory. Throughout a simulation, the replacement strategy adapts parameters for simulation stability and lookup efficiency.

The workflow of a MD-HM simulation starts from initializing the two-phase LSM-tree as the data storage for constructing the subgrid lookup table. Based on the input problem, MD-HM then builds a subgrid lookup table-based simulation using suitable subgrid size. At each time step, to compute subgrid potentials, MD-HM tries to identify a subgrid from the lookup table. If a match is found (i.e., a hit), the stored potentials will be used. Otherwise (i.e., a miss), direct numerical computation needs to be performed, and a new subgrid-potentials pair will be inserted into the lookup table. After the potential computation, all particles will be updated to new positions, and their velocities will be updated for the next step.

3 EVALUATION

Platform. We evaluate MD-HM on two Intel Xeon Gold 6252N 24-core processors running Linux 5.4.0. Each socket has 12 DIMM slots, six for 16-GB DDR4 DRAM modules, and six for 128-GB Optane DC modules. **Input problems.** We use nine molecular dynamics problems, such as, S-W, L-J, EAM, MEAM, Tersoff, ADP, G-B, AIREBO, and DPD. They cover a wide range of applications and come from LAMMPS benchmarks. Figure 1 shows the results.

Overall, MD-HM outperforms the baseline numerical framework (LAMMPS) on all input problems by up to 9.71× and 7.62× on average. Without the two-phase LSM-tree data storage (i.e., MD-HM with RocksDB-PMEM), the subgrid lookup based framework outperforms the baseline on six problems (i.e., S-W, MEAM, Tersoff, ADP, G-B, and AIREBO), achieving an average 4.41× speedup. However, due to the high read latency of RocksDB-PMEM, MD-HM with RocksDB-PMEM underperforms the baseline in three problems (L-J, EAM, and DPD). Our analysis shows that these problems have low FLOPs (typically lower than 90) for the computation of potential, which increases the impact of read latency due to frequent queries to lookup table. This challenge is addressed by the shallow search hierarchy in our proposed two-phase LSM-tree data storage. Thus, MD-HM with the two-phase LSM-tree outperforms the RocksDB-PMEM based implementation by an average speedup of 2.76×. We highlight that for the computation part of MD, MD-HM with two-phase LSM-tree can achieve higher performance, and the average speedup is 27.93× (up to 48.07× for AIREBO problem).

REFERENCES

[1] Zhen Xie, Wenqian Dong, Jie Liu, Ivy Peng, Yanbao Ma, and Dong Li. 2021. MD-HM: memoization-based molecular dynamics simulations on big memory system. In *Proceedings of the ACM International Conference on Supercomputing*. 215–226.