MDZ: An Efficient Error-bounded Lossy Compressor for Molecular Dynamics

Kai Zhao*, Sheng Di†, Danny Perez,‡, Xin Liang,§, Zizhong Chen*, and Franck Cappello†¶
* University of California, Riverside, CA, USA
† Argonne National Laboratory, Lemont, IL, USA
‡ Los Alamos National Laboratory, Los Alamos, NM, USA
§ Missouri University of Science and Technology, Rolla, MO, USA
¶ University of Illinois at Urbana-Champaign, Champaign, IL, USA
kzhao016@ucr.edu, sdi1@anl.gov, danny_perez@lanl.gov,
xliang@mst.edu, chen@cs.ucr.edu, cappello@mcs.anl.gov

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00007

***Abstract*—Molecular dynamics (MD) has been widely used in today's scientific research across multiple domains including materials science, biochemistry, biophysics, and structural biology. MD simulations can produce extremely large amounts of data in that each simulation could involve a large number of atoms (up to trillions) for a large number of timesteps (up to hundreds of millions). In this paper, we perform an in-depth analysis of a number of MD simulation datasets and then develop an efficient error-bounded lossy compressor that can significantly improve the compression ratios. The contributions are fourfold. (1) We characterize a number of MD datasets and summarize two commonly-used execution models. (2) We develop an adaptive error-bounded lossy compression framework (called MDZ), which can optimize the compression for both execution models adaptively by taking advantage of their specific characteristics. (3) We compare our solution with six other state-of-the-art related works by using three MD simulation packages each with multiple configurations. Experiments show that our solution has up to 233% higher compression ratios than the second-best lossy compressor in most cases. (4) We demonstrate that MDZ is fully capable of handing particle data beyond MD simulations.**

I. Introduction

Molecular dynamics simulations have become one of the most commonly-used methods to study the physical movements of atoms and molecules. For instance, MD simulations are often used to refine 3D structures of proteins and macro-molecules in terms of experimental constraints in X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy. In physics, MD simulations can be used to study the dynamics of atomic-level phenomena, such as thin-film growth and ion implantation (the atomic-scale details of which are very difficult to observe directly) or to investigate physical properties of nanoscale devices. In biophysics and structural biology, MD simulations are often applied to examine the motions of macromolecules (e.g., proteins and nucleic acids), for interpreting the results of some biophysical experiments and modeling interactions between molecules.

Generally speaking, scientific data can be categorized into three distinct types, including particle data (e.g., locations of atoms), structured mesh (regular multidimensional grid in space), and unstructured mesh (irregular mesh such as triangular grid). MD simulation is one of the most significant/typical particle-based research in the community. As the computational scales at which MD simulations are carried out rapidly increases [1], so does the volume of data generated during the simulations. For example, an atomistic model of the SGLT membrane protein may consist of 240 million frames each with 90k particles, producing a total of ∼260 TB of raw trajectory data over a 480 ns simulation[2]. The most recent MD simulations [1] are able to simulate 20 trillion particles in a long trajectory, generating 10 PB of data if there are hundreds of frames to store.

The explosive growth of data volume has brought major challenges to the storage systems designed for saving and managing scientific datasets [3], [4], [5]. For scientific applications, the vast amount of data are generally stored in the form of files [6], for the purpose of convenient post hoc analysis, management, and transfer. How to efficiently store and transfer the large amount of data becomes a serious concern. In fact, for today's supercomputers, a research project generally is allocated only dozens of terabytes of storage space (e.g., 50 TB by default on ORNL Summit [7]) or a few hundreds of terabytes upon requests. Obviously, efficiently reducing the volume of generated data can substantially lower the burden on storage, management and transfer.

Lossy compression has been considered by many researchers as a promising solution to the aforementioned data problems [8], [9], [10], [11], [12], [13]. In this paper, we aim at designing an efficient error-bounded lossy compressor for MD datasets, which presents a series of challenges. (1) In MD simulations, each snapshot may contain a large number of particles, so that only a limited number of snapshots can be held in memory and the compression should be done in batches. Therefore, compressors that rely on the time series patterns [14], [15], [16] will have sub-optimal performance, and a practical and effective compressor for MD data should involve both efficient time-based compression and efficient snapshot-based compression. (2) It is very challenging to develop an efficient snapshot-based compression method because

Corresponding author: Sheng Di, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 Cass Avenue, Lemont, IL 60439, USA

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00007

the adjacent data values in a snapshot may not be smooth (shown in Section V-B), while existing state-of-the-art lossy compressors substantially depend on the high smoothness of the data in space. (3) Unlike some existing compressors [16] optimized for cosmological N-body simulations, MD compressors could not exploit velocities to help compress position data in most cases, because MD particle often quickly vibrate around their equilibrium positions and velocities are only predictive of future positions for a few femtoseconds in the future (a fraction of a typical vibrational period).

With all the above challenges in mind, we propose a novel error-bounded lossy compressor that is particularly efficient for MD simulations. The key contributions are listed as follows:

- We carefully characterize a number of different MD simulation datasets and exploit some of the key patterns identified in the MD data to significantly improve compression ratios.
- We design an adaptive error-bounded lossy compressor for MD datasets which fully leverages the specific characteristics in both spatial and temporal dimensions.
- We evaluate our solution with six state-of-the-art related works. Experiments show that MDZ can always lead to the best compression quality in various execution patterns. In absolute terms, our solution obtains up to 233% higher compression ratios than does the second best error-bounded lossy compressor.
- We integrate our solution into the MD package LAMMPS. Evaluation shows our solution has negligible time overhead in real-world MD simulations under different scales and settings.
- We discuss the generalizability of our solution and demonstrate MDZ has the best compression quality on datasets beyond MD simulations.

The rest of the paper is organized as follows. In Section II, we discuss the related work. In Section III, we describe the research background. In Section IV, we formulate the research problem. In Section V, we present an in-depth characterization of several MD simulation datasets, which motivates our design and optimization. In Section VI, we describe in details our developed MD data compressor - MDZ. In Section VII, we present and discuss the evaluation results. We conclude the paper in Section VIII.

II. Related Work

The compression of MD datasets is critical to the cost-effective data processing of MD simulations.

In general, compression techniques can be divided into two categories - lossless compression and lossy compression. Lossless compressors have been deployed in many fields. For example, Google Brotli [17] and Facebook Zstandard [18] are widely used in industrial data management systems. Gorilla [19] and AMMMO [20] bring lossless methods to time series databases. However, lossless compressors suffer from very low compression ratios in the scientific domain, as demonstrated in Section VII-B. The reason is that scientific datasets are mainly composed of floating-point numbers each of which has very random ending mantissa bits so that it is very hard for lossless compressors to catch the repeated patterns during the encoding.

Lossy compression, unlike lossless compression, can reach a higher compression ratio with some information loss. Lossy compression has been adopted in some database systems. For example, ModelarDB [21], [22] is a time series management system with lossy compression built-in. It has three compression algorithms, including the PMC-mean [23], the linear Swing model [24], and the lossless method in Gorilla [19]. ModelarDB uses a window-based approach to find the best algorithm for each data segment. SummaryStore [25] is an approximate time-series store which merges the old data when the space limit is reached. Besides time series databases, there are also some lossy compression studies [26], [27], [28], [29] for GPS trajectory data systems.

Lossy compressors in database systems are not suitable for MD datasets for the following reasons. First, time series databases such as ModelarDB use simple data estimation methods and they do not have quantization or entropy coding process, thus they suffer from low compression ratios on MD datasets (demonstrated in Section VII-C). Second, GPS trajectory compressors are not suitable for MD datasets either because MD data is much more unconstrained than the GPS data (note that GPS devices follow direct lines while MD particles move rather randomly). Third, many database systems such as SummaryStore do not have an error-bounded design such that they cannot guarantee the quality of the decompressed data would satisfy the users' requirements.

Even general lossy compressors for scientific applications such as ZFP [30] and SZ-Interp [31] exhibit sub-optimal results on MD datasets [16], because they are designed and optimized for three-dimensional data. While MD datasets are two-dimensional and are split into batches for compression.

Due to the above limitations, researchers are investigating lossy compressors that are specifically designed for MD datasets. HRTC [2] adopts a piecewise linear representation of trajectories, followed by an error-controlled quantization and a variable length integer representation. Li et al. [16] improved the compression ratio by employing velocity fields to assist the prediction of spatial coordinates. Note that, as mentioned in Section I, this strategy may not be efficiently applied to MD datasets. PMC [32] utilizes information on atomic bonds in a molecule to predict atomic positions in each frame. This method, however, is not suitable for simulations with non-bonded interactions, where connectivity between neighboring atoms can dynamically change during the simulation.

III. Research Background

Two key sets of background concepts — MD simulation and error-bounded lossy compression — are important to the development of our novel error-bounded lossy compressor for MD simulation datasets.

A. MD Simulations

MD is a type of N-body simulations which is widely used to explore the behavior of materials at the nanoscale. As illus-

trated in Figure 1, a single MD simulation generally involves many time steps, in each of which the new position and velocity of each particle are predicted based on sophisticated calculations of interatomic forces. Force calculation typically consumes the overwhelming majority of the computing time. After adjusting atomic positions based on the calculated forces (shown as the highlighted arrow), boundary conditions are applied and coordinates or physical quantities of interest care calculated and written out.

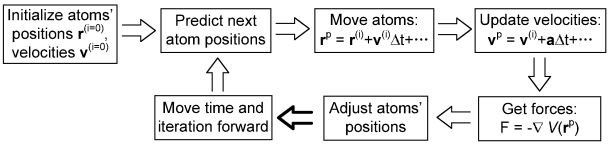


Fig. 1. Illustration of classic MD simulation

A typical MD output is dominated by the storage of coordinate information along the trajectory. Each particle's position is composed of three axes (x, y, z). This is why most of the existing lossy trajectory compressors [2], [32], [33], [34] focus on the positions rather than velocities. As such, in the following, the targets for compression are the particles' positions (x, y, z) alone.

B. SZ Error-bounded Lossy Compression Framework

Our proposed compression technique builds upon the SZ lossy error-bounded compression framework [35], [31], [36], [37]. The SZ framework supports customized prediction stage, allowing us to exploit MD application-specific characteristics and patterns in order to improve the compression quality.

SZ proceeds by four critical steps, as illustrated in Figure 2: (1) data prediction, (2) linear-scale quantization, (3) entropy coding (i.e., Huffman coding), and (4) lossless compression (e.g., Zstd [18]). In most applications, tuning the first step is crucial to achieve high compression quality on specific applications, as higher prediction accuracy would yield better distribution of quantization bins and thus higher compression ratio under the Huffman coding.

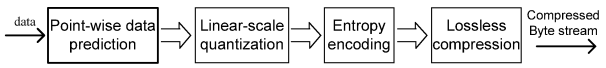


Fig. 2. Illustration of SZ compression framework

IV. Problem Formulation

In this section, we formulate the research problem by classifying the input and output of error-bounded lossy compressors in the context of MD simulation datasets.

The research problem can be formulated as follows. Suppose an MD simulation dataset (denoted by $\boldsymbol{D}$) is composed of M snapshots each containing N particles. Atomic positions (represented as three axes values $\{x, y, z\}$) need to be stored to disks during the simulation.

In general, compression time should be negligible compared with the time to execute hundreds to thousands of timesteps. In most MD simulations, the stiffness of the equations of motion entails very short timesteps on the order of femtoseconds, which is a small fraction of the vibration period of the fastest modes in the system. Hence, by construction, very little structural changes occur between neighboring timesteps. As transitions that change the topology are typically thermally-activated, simulation data need to be saved only occasionally (i.e., thousands to tens of thousands of timesteps). For applications to estimate systems with fast relaxation processes, e.g., to estimate the viscosity of liquids, or the vibrational properties of solids, a higher frequency might be required (e.g., hundreds of timesteps).

Accordingly, our research target can be summarized as maximizing the compression ratio while keeping the compression and decompression speed fast enough for the MD simulations, and processing the M snapshots in batches instead of compressing the entire dataset D at once.

Based on the above problem definition, traditional pure trajectory compression methods [16], [2], [34] are not suitable, since they need to collect a large number of snapshots for the compression, and decompressing any one snapshots needs to decompress all its preceding snapshots as well. Moreover, single-snapshot based compression [38], [8] is not an ideal solution either, in that it will suffer from low compression ratios because of the non-smooth nature of the spatial particle data. To address these issues, we propose MDZ which makes full use of the characteristics of MD datasets in both spatial and temporal dimensions to significantly improve compression ratios.

V. Investigation of MD Datasets

In this section, we identify key characteristics and patterns from a number of MD datasets. Specifically, we first analyze the spatial patterns present in MD datasets and then investigate their temporal features.

A. MD Simulations Used in Our Work

Table I summarizes the eight MD simulation datasets that are considered in the following. For Copper and Helium datasets, we include two broad execution modes, noted A and B. In the mode A simulations, each snapshot involves a relatively large number of atoms (generally more than 100K atoms). These are typical of conventional large-scale MD simulations. In mode B, each simulation involves a large number of timesteps and a relatively small number of atoms (such as 1k atoms). This mode is more typical of long-timescale simulations, e.g., using methods such as Parallel Trajectory Splicing [39].

The eight datasets can be described as follows.

- **Copper (Mode A&B)**: The data comes from the study of the influence of strong electric fields on copper in the context of particle accelerators. The mode A sample contains 1077290 atoms and the mode B sample has 3137 atoms. The time evolution was obtained by molecular dynamics method using the LAMMPS code [40] in the canonical ensemble at a temperature of 800K. The simulation was run on up to 30 nodes (1024 cores) of

TABLE I
MD SIMULATION DATASET IN OUR STUDY

Application	State	Code	Snapshots	Atoms
Copper-A	Solid	LAMMPS	83	1077290
Copper-B	Solid	LAMMPS	5423	3137
Helium-A	Plasma	LAMMPS	2338	106711
Helium-B	Plasma	EXAALT	7852	1037
ADK	Protein	CHARMM	4187	3341
IFABP	Protein	CHARMM	500	12445
Pt	Solid	LAMMPS	300	2371092
LJ	Liquid	LAMMPS	50	6912000

the Grizzly [41] supercomputer at Los Alamos National Laboratory (LANL).

- **Helium (Mode A)**: This dataset contains simulations of the growth of helium bubbles embedded in a body-centred cubic tungsten matrix. The simulation cell contains 106711 atoms. Helium atoms are gradually inserted in the bubble as the simulation proceed, mimicking the agglomeration of helium atoms incoming from the plasma into the first wall of a fusion reactor. The simulations were carried out with the Parallel Replica Dynamics method [42] using the LAMMPS code [40]. Simulations were carried out on up to 1000 nodes of the Trinity supercomputer [43] at LANL.
- **Helium (Mode B)**: This dataset contains simulations of small vacancy/helium clusters in a body-centred cubic tungsten matrix. The simulation cells contain 1037 atoms. Long-time simulations were carried out with the Parallel Trajectory Splicing methods [39] to investigate the mobility of these defects formed by helium atoms incoming through the plasma in contact with the first wall of fusion reactors [44]. These simulations were carried out on up to 2000 nodes of the Trinity supercomputer at LANL.
- **ADK:** This dataset is from the simulation of adenylate kinase (ADK) which is the critical enzyme controlling the energy balance in cells. According to Seyler[45], ADK was simulated with explicit water and ions in isothermal–isobaric ensemble settings with temperature being 300 K and pressure being 1 bar. The experiment was conducted on the biomolecular-optimized Anton supercomputer [46] at Pittsburgh Supercomputing Center. The snapshots contains 3341 atoms and were saved every 240 picoseconds for a total runtime of 1.004 µs.
- **IFABP:** The data comes from an MD simulation with 12445 atoms of intestinal fatty acid-binding protein in water. Fatty acid-binding proteins affect the transfer of fatty acids between cell membranes while their mechanism are largely unknown. The simulation data is valuable for studying protein dynamics, protein-ion, and protein-water interactions [47]. The experiment was running for 500 picoseconds using CHARMM[48]. The timestep is set to 2 femtoseconds and the snapshots are saved every 1 picosecond.
- **Pt**: The data corresponds to an MD simulation of surface diffusion and adatom clustering on a platinum surface. The model had 2371092 atoms and was run for 32M timesteps using the local hyperdynamics methodology. More details on the method and simulation analysis are given in [49]. The simulation was run on 64 KNL nodes (4096 cores) of the Theta supercomputer at ALCF [50].
- **LJ**: This simulation dataset was generated by the Lennard-Jones liquid benchmark [51], [52]. The Lennard-Jones potential estimates the potential between particles based on the particle distance. LAMMPS includes the Lennard-Jones potential as one of the simulation benchmarks. The simulation cell contains 6912000 atoms. The simulation was run on up to 500 cores of the Bebop supercomputer [53] at Argonne National Laboratory.

B. Characterization of Spatial Features

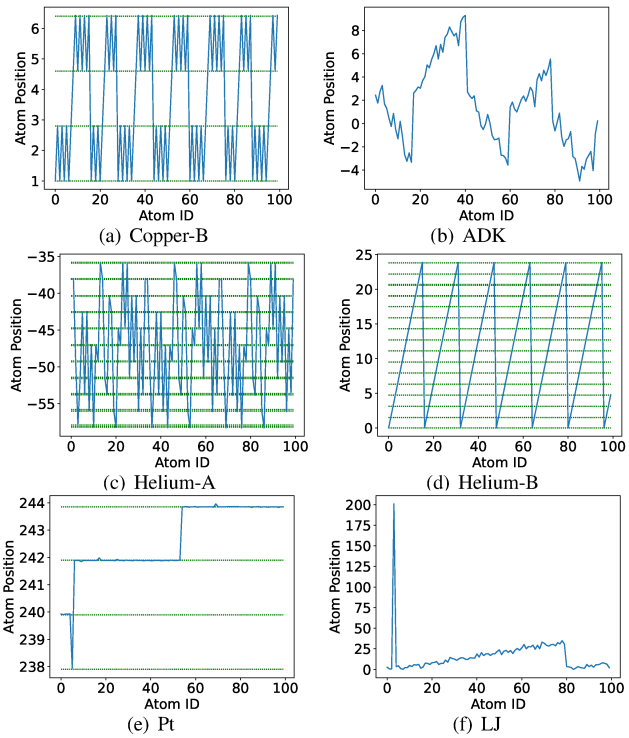


Fig. 3. Demonstration of spatial correlations in atom position data

Takeaway 1: Our first critical observation is that in many cases, the MD datasets exhibit various patterns in the spatial domain. Due to the space limit, we give six typical examples (including Copper-B, ADK, Helium-A, Helium-B, Pt and LJ) to demonstrate the diverse spatial patterns in Figure 3. As illustrated in the figure, the dataset may exhibit a stable zigzag pattern (Figure 3 (a) (d)), an erratic zigzag pattern (Figure 3 (c) (f)), a stair-wise pattern (Figure 3 (e)), or a random pattern (Figure 3 (b)).

Takeaway 2: We also observe from Figure 3 (a) (c) (d) that in many cases, the data are clustering into several equal-distant discrete levels in the whole value range. In fact, for all the data points that are clustering at a specific level, their positions actually vibrate in a small range and are not strictly constant. These regular patterns emerge from the crystalline

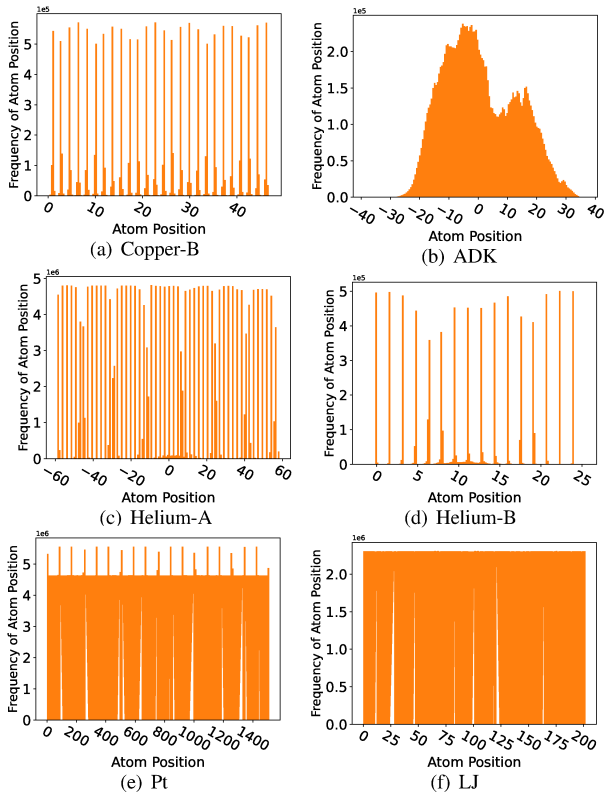


Fig. 4. Demonstration of frequencies of atom position data

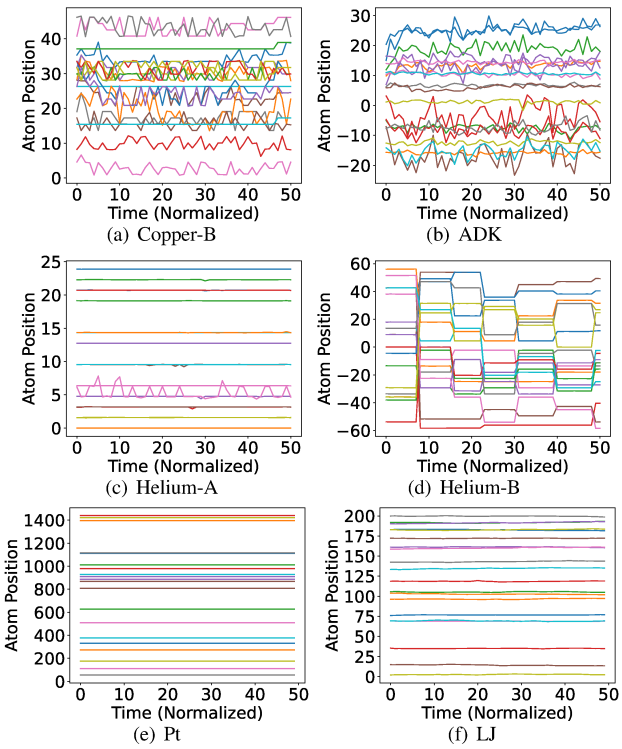


Fig. 5. Demonstration of temporal correlations in atom position data, time is normalized to 0-50

structure of the underlying materials. The observed zig-zag patterns are also typical of how crystalline samples are usually created. Such patterns can change with time, as the structure of the materials evolve.

Takeaway 3: Based on Figure 3, we also learn that the atom's coordinate may jump from one discrete level to another nearby level, point by point throughout the whole dataset. Since many prediction-based compressors such as SZ simply predict each data point based on its preceding data points without explicitly using the discrete levels, it would definitely suffer from relatively low prediction accuracy in this situation, leading to low compression ratios (as discussed in Section III).

As mentioned above, we observe that the data values often cluster and vibrate around a number of different discrete levels, which can be verified by the distribution of the data (as presented in Figure 4). As shown in the figure, the distribution of any MD dataset can be split into two categories - multiple-peak-dominated distribution (see Figure 4 (a) (c) (d)) and rather uniform distribution (see Figure 4 (b) (e) (f)). The former clearly indicates the strong clustering feature of the data in many cases, which is consistent with our analysis based on Figure 3.

C. Characterization of Temporal Features

Datasets which have no prominent spatial patterns often exhibit particular temporal correlations that can be used to achieve very high compression ratios. Figure 5 presents the position data in the time dimension (i.e., trajectories of atomic positions) for six datasets. It is clearly observed that the data value always exhibit more or less correlations in the time dimension. Basically, there are two correlation levels as summarized below. (1) The data may change relatively largely and frequently for some datasets (such as Copper-B, ADK, and Helium-B). (2) The data may change slightly in some situations (such as Helium-A, Pt, and LJ).

Takeaway 4: One very useful observation is that for the datasets which exhibit low spatial patterns, for example Pt and LJ, they often have extremely strong data smoothness in the time dimension and a large majority of the data are extremely close in the time dimension throughout the whole simulation.

Based on the four important takeaways explored in our characterization, we develop an adaptive error-bounded lossy compressor for the diverse MD datasets which can significantly improve the compression ratios over the existing state-of-the-art MD compressors.

VI. MDZ: An Adaptive Error-bounded Lossy Compressor for MD Datasets

The basic design idea is selecting the best method from among three compressors best suited to diverse data features in both the spatial and temporal dimension.

Figure 6 summarizes the design of MDZ. Basically, the datasets are generated by the data source such as the MD simulation, as illustrated in Figure 1. As mentioned in Section I and Section IV, the MD applications need to perform the compression operation periodically in order to avoid out-of-memory issue. The snapshots to be compressed are stored in

a buffer and the buffer size (BS) is defined as the maximum number of snapshots to be kept in the buffer. Finally, the compressed data will be stored into the parallel file systems (PFS).

As illustrated in Figure 6, the entire compression pipeline involves four critical steps: prediction, optimized quantization, encoding and Zstd, following the classic SZ compression framework [54], [35]. Our key contribution involves improvements to the prediction and quantization stages.

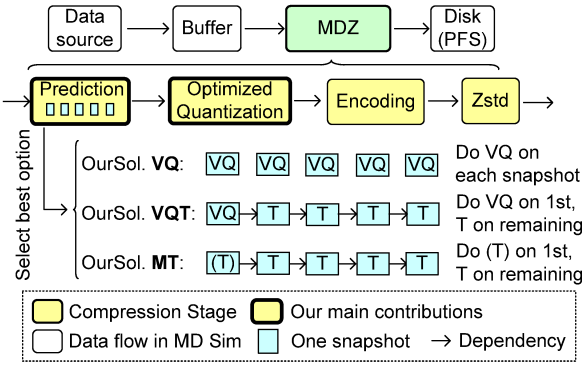


Fig. 6. Design Overview (VQ,VQT,and MT are described in Section VI-A and VI-B)

Specifically, we design three efficient MD data prediction strategies to adapt to diverse data patterns in the MD datasets:

- Vector-quantization-based compressor (abbreviated as **VQ**): The VQ compressor predicts the data values totally based on the spatial information, thus the data prediction for any one snapshot has no dependencies on any other snapshots, such that any snapshot data can be decompressed very quickly without a need in decompressing other snapshots. This is particularly effective on the datasets with very low smoothness in time dimension (see Figure 5 (a) (b)).
- Vector-quantization-time-based compressor (abbreviated as **VQT**): The VQT compressor adopts the VQ predictor on the first snapshot in the buffer, and adopts a time-based predictor (i.e., predict each data point using the corresponding data values in the previous snapshot) for all the remaining snapshots in the buffer. This method is designed particularly for datasets that have smooth time dimension and also have strong multi-peak-distribution patterns in space (see Figure 5 (c) and (d)).
- Multi-level-time-based compressor (abbreviated as **MT**): The MT compressor adopts a particular data prediction method - called *initial-time-based prediction* (shown as the notation (T) in Figure 6)), and applies the ordinary time-based predictor on all other remaining snapshots in the buffer. This method is particularly effective on the datasets with very high smoothness in the time dimension (see Figure 5 (e) (f)).

We describe in detail the compressors with optimized prediction and quantization methods in the following subsections.

A. Vector-Quantization-Based Compression (VQ and VQT)

The basic idea of the VQ algorithm is to leverage the spatial patterns characterized in Section V-B (i.e., takeaway 2 and takeaway 3). Takeaway 2 indicates that the data are clustering into different roughly equal-distant discrete levels (as shown in Figure 3 and Figure 4), which motivates us to use the centroid (a.k.a., center) of each cluster to predict all the data values within this cluster.

We present the pseudo-code of the VQ algorithm in Algorithm 1. The first step is computing the level distance λ and the initial level value μ (line 1), based on which every level value can be retrieved easily. Line 2∼8 is is the main compression procedure, including data prediction (line 4∼5), computation of level index (line 6) and quantization (line 7).

Algorithm 1 VECTOR-QUANTIZATION-BASED COMPRESSION (VQ)

Input: raw MD data D (single snapshot)
Output: compressed data (byte stream)

1: Compute the level distance λ and the initial level value μ by the sampling-based KMeans;
2: Store d_o as it is;
3: **for** $d_i \in D$, where i=1, 2, $\cdots$, N **do**
4: $L_i \leftarrow \text{Round}(\frac{d_i - \mu}{\lambda})$; /*Compute level*/
5: $V_i \leftarrow \mu + \lambda \cdot L_i$; /*Get the level's centroid value - **VQ based predictor***/
6: $j_i \leftarrow L_i - L_{i-1}$; /*Compute relative level index*/
7: $b_i \leftarrow (\frac{d_i - V_i}{e} + 1)/2$; /*linear-scale quantization, where e is error bound*/
8: **end for**
9: $\hat{B} \leftarrow \text{HUFFMAN}_{b_i \in B}(B)$; /*Huffman encoding on quantization codes*/
10: $\hat{J} \leftarrow \text{HUFFMAN}_{j_i \in J}(J)$; /*Huffman encoding on level index codes*/
11: $\text{ZSTD}(\hat{B} + \hat{J})$; /*Compress Huffman output by Zstd [18]*/

We illustrate the key steps (data prediction and quantization) of the VQ compression algorithm in Figure 7. The figure shows a snippet of the dataset (i=10 → 26). As we mentioned previously, the data values are clustered at different levels with a small vibration, so we use the corresponding level's centroid value to predict each data point. As such, the quantization bin (see the red number 3 in the figure) is calculated based on the prediction error (i.e., $d_i - V_i$). The vector B is used to hold the quantization bins, and J is used to hold the relative index numbers. Both of them will be compressed by Huffman encoding later on (line 9-10 in Algorithm 1).

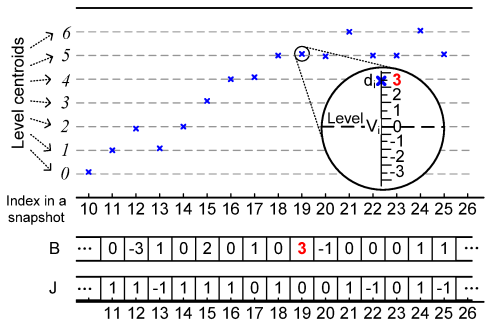


Fig. 7. Illustration of VQ-based Prediction + Quantization

As mentioned above, we develop an efficient sampling-based 1D K-means clustering algorithm to identify the level distance and initial level value. In what follows, we first describe the basic K-means algorithm and then discuss how we boost its performance in the context of compression.

Unlike the time-consuming 2D K-means problem, optimally partitioning N sorted 1-dimensional data points to K groups has polynomial time complexity solutions. Define the sorted data points as $d_1, d_2, .., d_N$, and the cost of clustering as the summation of the distance between the data points and their centroid points. In Formula (1), we define $Cost(l, r)$ as the optimal cost of clustering $d_l, .., d_r$ to one group, $F(n, k)$ as the optimal cost of clustering $d_1, .., d_n$ to k groups, and $H(n, k)$ as the argument that minimizes F.

$$\begin{aligned} &Cost(l,r) = \textstyle\sum_{i=l}^{r}(d_i - \frac{\sum_{j=l}^{r} d_j}{r-l+1}) \\ &F(n,k) = \min(F(i-1,k-1) + Cost(i,n), \forall 0 < i <= n) \\ &H(n,k) = \arg\min(F(i-1,k-1) + Cost(i,n), \forall 0 < i <= n) \end{aligned} \quad (1)$$

The boundaries of clusters can be restored from H iteratively. The naïve implementation to solve $F(N, K)$ has $O(KN^2)$ time complexity, and we adopt a solution [55] that optimizes the computational cost to $O(KN)$.

In our case, the number of clusters K is unknown and the data points are unsorted. To boost the performance, on the one hand, we compute F only once during the whole simulation, and we compute it on a sampled dataset that has 10% data points from the first single snapshot. We observe the snapshots have unchanged level patterns during the simulation thus the result on the first snapshot is applicable for the following snapshots. On the other hand, note that the value of $F(N, 1), F(N, 2), .., F(N, K)$ are computed in order when computing $F(N, K)$. Let $G(k) = \frac{F(N,k)}{F(N,k-1)}$; we stop the computation of F at κ if $G(\kappa)$ decreases significantly than $G(\kappa - 1)$. The maximum test value of K is set to 150 as a higher number of clusters will harm the compression ratio of the vector quantization indexes. The level distance λ and initial level value μ are computed using the boundaries obtained from H.

For the VQ compression method, we adopt the VQ algorithm on each snapshot, as illustrated in Figure 6. By comparison, the VQT compression method applies the VQ algorithm only on the first snapshot in each buffer, and all other snapshots in the buffer will be compressed by the classic time-based compression. Specifically, each subsequent data point will be predicted using the corresponding data value in the previous timestep. This may significantly improve the compression ratio especially in situations with relatively smooth data in the time dimension (see Figure 5 (c)-(f)).

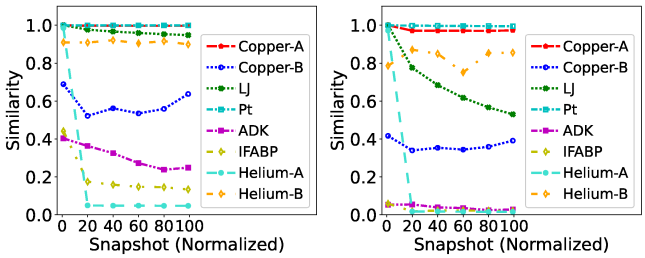
B. Multilevel Time-based Compression (MT)

We propose an additional error-bounded compression method - called multi-level-time-based compression (MT), which is particularly effective for the datasets with extremely high smoothness in the time dimension.

The MT compression algorithm also adopts the prediction-based compression model. The particular design of MT is that the first snapshot in the buffer will be predicted based on the initial snapshot of the whole dataset, which is motivated by the very strong correlation between all the simulation snapshots and the initial snapshot in some datasets. Figure 8 shows the similarity of all the snapshots compared with the initial snapshot (i.e., snapshot 0). The similarity is defined in Formula (2).

$$Similarity(\tau, i) = \frac{Count(|\frac{S_i[j]-S_0[j]}{S_i[j]}| < \tau, \forall j)}{Count(S_i)} \quad (2)$$

where τ refers to a threshold, S_i refers to snapshot i, $S_i[j]$ refers to the jth data point in the snapshot S_i. The similarity formula calculates the percentage of the "unchanged" data points (atoms) based on some threshold τ. The figure clearly demonstrates that succeeding snapshots in some datasets such as Copper-A and Pt are always extremely similar to the initial snapshot.



(a) Threshold = 10% (b) Threshold = 1%

Fig. 8. Snapshots Similarity with Snapshots-0 (Snapshots normalized to 0-100)

Using the particular snapshot-0-based prediction, the prediction error could be much lower than the traditional spatial prediction error such as Lorenzo-predictor, as presented in Table II.

TABLE II
PREDICTION ERRORS FOR THE FIRST SNAPSHOT IN BUFFER

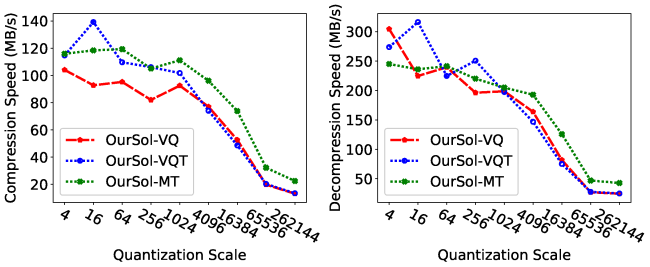
Method	Pt			LJ			Helium-B		
	x	y	z	x	y	z	x	y	z
Lorenzo+Regression	3.46	5.50	3.88	5.87	3.87	3.75	5.83	1.14	0.83
MT(Snapshot-0-based)	0.13	0.13	0.01	1.37	1.37	1.38	0.08	0.07	0.09

C. Linear-Scale Quantization Optimizations

In this section, we optimize the linear-scale quantization step by tuning two quantization settings to further improve the overall compression performance and quality.

1) Optimization of Quantization Scale: The quantization scale controls the value-range of the quantized integers. The data points that are out-of-scope will be marked with reserved integer value and stored separately. A smaller scale will increase the number of out-of-scope data points which impacts the compression ratio, while a larger quantization scale leads to a bigger Huffman tree such that the Huffman coding will be slower. In Figure 9, we illustrate the compression/decompression speed with different quantization scale settings. The compression speeds of VQ, VQT, and MT decrease from 95MB/s, 109MB/s, 119MB/s to 19MB/s, 20MB/s, 32MB/s respectively when the quantization scale changes from 64 to 65536. As such, in our solution, we set the optimal quantization scale to 1024, which can always

keep a high compression performance while preserving a high compression ratio.



(a) Compression (b) Decompression

Fig. 9. Compressor Performance Affected by Quantization Scale on Helium-B Dataset (value-range-based error bound (ϵ) = 1E-3, BS = 10)

2) Optimization of Quantization Sequence: The quantization sequence controls how the integers from multiple snapshots are stored together as 1D array for the Huffman coding and dictionary coding. We denote Seq-1 as storing one snapshot first, then storing the following snapshots. Seq-2 is denoted as storing one particle in all snapshots first, then storing the following particles. We observe that Seq-2 is better than Seq-1 in terms of compression ratio, especially when the data is stable in time (as shown in Figure 5 (c) (e) (f)). When many data points remain unchanged in the time dimension and if they are put together as required by Seq-2, the dictionary coder will have better compression results. Table III demonstrates compression ratios of the two sequences on Helium-B dataset. The second row of the table is the value-range-based error bound (ϵ), and the corresponding absolute error bound is $value\_range \times \epsilon$. The table shows Seq-2 improves the compression ratio by 37.8%, 37.6%, and 39.7% over Seq-1 on axis x, y, and z respectively. As a result, we adopt Seq-2 in our solution.

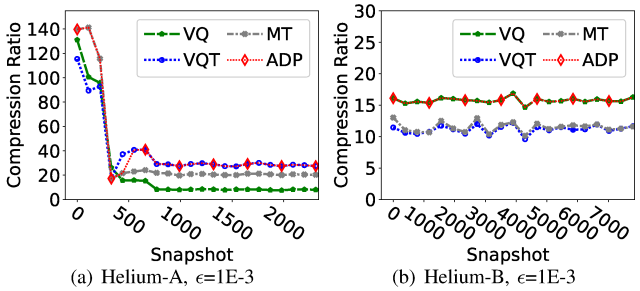
TABLE III
COMPRESSION RATIO (CR) OF HELIUM-B DATASET WITH DIFFERENCE SEQUENCE SETTINGS, BUFFER SIZE (BS) = 10 (METHOD=MT)

Axis	X			Y			Z		
ϵ	1E-1	5E-2	1E-2	1E-1	5E-2	1E-2	1E-1	5E-2	1E-2
Seq-1	156	97	46	176	101	47	146	97	46
Seq-2	**215**	**132**	**53**	**236**	**139**	**54**	**204**	**133**	**53**

D. Adaptive Selection of Best Compressor (ADP)

In this section, we propose our adaptive solution (ADP) that can select the best compressor (VQ, VQT or MT) dynamically at runtime. MDZ uses ADP by default to simplify the compression configuration, while manually choosing VQ, VQT, or MT as the compressor is also supported in MDZ.

We notice that during the simulation, the data patterns stay the same in a short term and the patterns (either spatial or temporal) may change prominently in the long term. Furthermore, the best compressor keeps its advantage across some snapshots, but it may not be the best one on all the snapshots. As illustrated in Figure 10 (a), MT has the highest compression ratio before snapshot 400 and VQT becomes the best compressor after that snapshot. As a result, we propose to evaluate the three compressors (VQ, VQT, and MT) periodically by using them to compress the same data batch independently and selecting the one with the best compression ratio for the following snapshots. The evaluation will be invoked every 50 compression operations. This time interval ensures that the best compressor is updated in time, while keeping the updating overhead low (less than 6% of the total compression time). Figure 10 confirms the effectiveness of our adaptive solution (ADP). All other datasets exhibit similar results (i.e., our ADP algorithm can always select the best solution accurately).

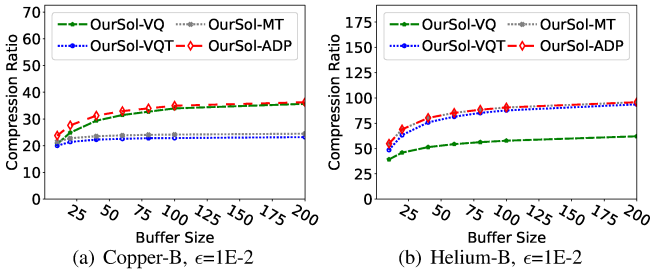


(a) Helium-A, ϵ=1E-3 (b) Helium-B, ϵ=1E-3

Fig. 10. Illustration of smooth CR in short term and diverse CR in long term (BS=10). ADP can pick up the best compressor throughout all the snapshots.

VII. EXPERIMENTAL EVALUATION

In this section, we present the experimental settings and the evaluation results of our solution on eight MD simulation datasets.



(a) Copper-B, ϵ=1E-2 (b) Helium-B, ϵ=1E-2

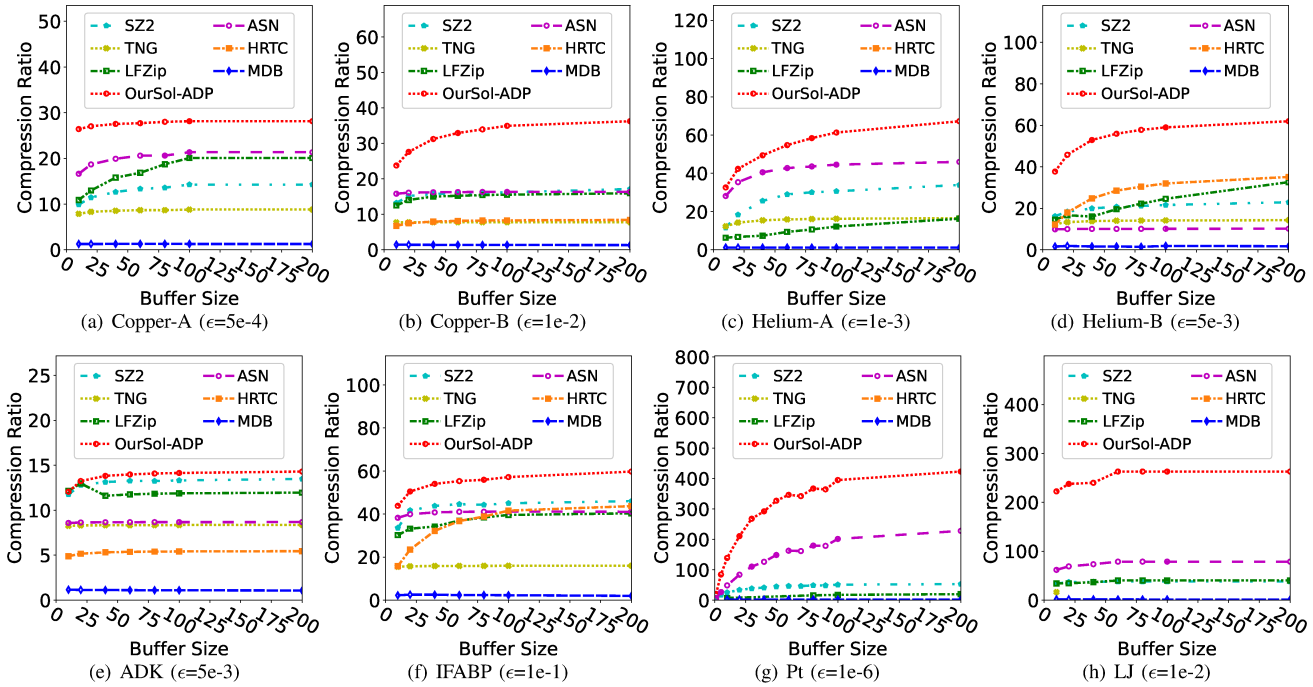
Fig. 11. Our adaptive solution (ADP) has the highest compression ratio over VQ, VQT, and MT under different datasets and buffer size (BS) settings, because ADP can always select the best compression method accurately.

A. Experimental Setting

1) Execution Environment: The experiments are executed on the Bebop supercomputer [53] at Argonne National Laboratory with up to 216 cores. Each node in Bebop is equipped with two Intel Xeon E5-2695 v4 processors and 128GB memory.

2) Datasets: The experiments are evaluated on eight real-world MD simulation datasets. The detailed information about the datasets is presented in Section V-A and Table I.

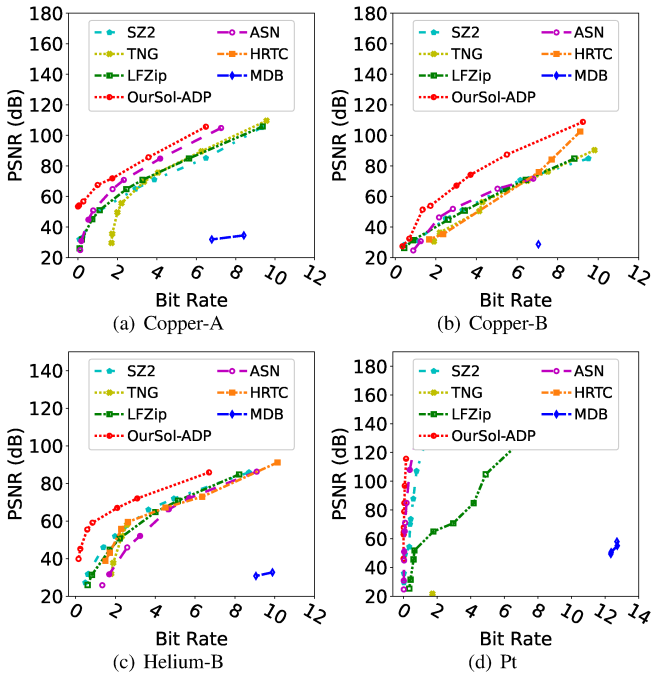
3) State-of-the-Art Lossless Compressors in Our Evaluation: We evaluate six lossless compressors as a comparison with lossy compressors. We include Zstd, Brotli, and Zlib which are widely used in databases and file systems. We also include ZFP, Fpzip, and FPC which specifically target the floating-point data format and are the state-of-the-art lossless compressors for scientific datasets.



(a) Copper-A (ϵ=5e-4) (b) Copper-B (ϵ=1e-2) (c) Helium-A (ϵ=1e-3) (d) Helium-B (ϵ=5e-3)

(e) ADK (ϵ=5e-3) (f) IFABP (ϵ=1e-1) (g) Pt (ϵ=1e-6) (h) LJ (ϵ=1e-2)

Fig. 12. Our solution has the highest compression ratio on all datasets and under different buffer size settings, HRTC and TNG fail to run on some datasets.



(a) Copper-A (b) Copper-B

(c) Helium-B (d) Pt

Fig. 13. Rate-distortion graphs show our solution has the best compression quality. Lower bit rate and higher PSNR indicate better compression quality.

4) State-of-the-Art Lossy Compressors In our Evaluation: We compare our solution with two MD data compressors, two widely used scientific data compressors, as well as two state-of-the-art time series compressors.

- TNG [15]: a MD compressor that uses quantization, delta coding, and a set of integer compressors to compress the trajectory data. TNG is supported by the MD simulation package GROMACS[56].
- HRTC [2]: a lossy compressor targets on MD trajectory compression. HRTC relies on piecewise linear function to approximate data points.
- ASN [16]: a scientific compressor designed for N-body simulation that utilizes the time dimension for prediction.
- SZ2 [57]: a prediction based error-bounded lossy compressor. SZ is widely used in many scientific domains. The framework of SZ2 is described in Section III-B.
- MDB: a full C++ implementation of ModelarDB's compression solution. ModelarDB is described in Section III. ModelarDB tightly couples its compressor with many database features which are useless for scientific data and introduce extra overhead. As such, we eliminate the overhead caused by those features for a fair comparison.
- LFZip [58]: a lossy compressor designed for multivariate floating-point time series data. LFZip is a prediction-based lossy compressor. We evaluate LFZip with its normalized least mean square (NLMS) predictor and skip its neural network (NN) predictor because the NN predictor requires training and is 2000X slower than the NLMS predictor according to the authors [58].

SZ supports both 1D mode and 2D mode. Table IV presents the compression ratios of SZ in the two modes. We can observe from the table that the 2D mode has up to 200% higher compression ratios than 1D, because 2D mode can utilize the data continuity in the space and time dimension at the same time. In our experiments, we use 2D mode for SZ.

5) Excluded Cases: HRTC has runtime exceptions on Copper-A, Helium-A, Pt, and LJ datasets. TNG has runtime

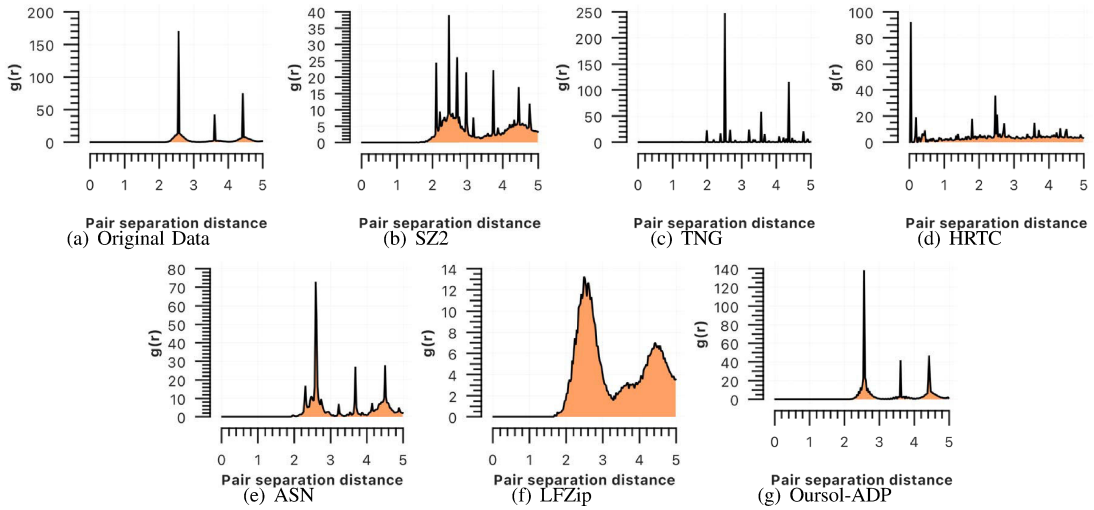


Fig. 14. Only our solution yields the correct radial distribution function (RDF) on decompressed data (Copper-B, CR=10, BS=10)

TABLE IV
COMPRESSION RATIOS OF SZ IN 1D AND 2D MODES (BS=10, ϵ=1E-3)

Method	Mode	Pt			LJ			Helium-A		
		x	y	z	x	y	z	x	y	z
SZ2	1D	150.5	139.6	38.9	6.35	6.45	6.53	7.18	7.28	6.36
	2D	356.5	371.6	32.1	12.26	12.40	12.44	11.11	12.03	11.58

exceptions on Pt and LJ datasets. A possible reason is the number of atoms is larger than their upper limit. As a result, no HRTC or TNG results are shown on those datasets.

B. Evaluation Results and Analysis of Lossless Compressors

We first evaluate the six state-of-the-art lossless compressors. Table V shows the compression ratios of the lossless compressors on four of the MD datasets (results are similar on other datasets). It is clear that all the lossless compressors have extremely low compression ratios (around 1∼2). The results confirms our statement in Section II that lossless compressors are not suitable for scientific applications.

TABLE V
COMPRESSION RATIO COMPARISON OF LOSSLESS COMPRESSORS

Dataset	Zstd	Zlib	Brotli	Fpzip	FPC	ZFP
Copper-A	1.13	1.15	1.14	1.41	1.18	1.47
Helium-B	1.38	1.33	1.37	1.29	1.22	1.30
ADK	1.08	1.07	1.08	1.26	1.09	1.21
LJ	1.23	1.31	1.24	1.44	1.16	1.39

C. Evaluation Results and Analysis of Lossy Compressors

The evaluation involves two aspects - the compression quality and performance. On the one hand, the evaluations of compression error, compression ratio, and rate-distortion demonstrate that our solution has superior compression quality over other state-of-the-art lossy compressors. On the other hand, the performance evaluation reveals that our solution has near the top compression and decompression throughput.

1) Compression Ratio: Figure 11 demonstrates the compression ratio of our solutions. We can observe that ADP has the highest compression ratio among our solutions under different datasets and buffer size settings. It further confirms our claim in Section VI-D that ADP can always select the best compressor from VQ, VQT and MT accurately during runtime. As such, we focus on ADP in the following evaluation section.

Figure 12 compares the compression ratio of the lossy compressors in different buffer size settings. It clearly shows that our solution always has the highest compression ratio on all the eight datasets with any buffer settings. In particular, when buffer size is 100, our solution has 31%, 114%, 38%, 84%, 6%, 27%, 96%, 233% compression ratio improvements over the second-best on Copper-A, Copper-B, Helium-A, Helium-B, ADK, IFABP, Pt, and LJ datasets respectively. MDB has extremely low compression ratios (1∼6) on all the datasets, as shown in Figure 12. The result confirms our statement in Section III that simple data estimation methods and the lack of quantization and entropy coding make ModelarDB suffer from low compression ratios on MD datasets. As a comparison, LFZip, which is also a time series compressor, has comparable results with other lossy compressors, because LFZip has the adaptive linear predictor as well as quantization and entropy coding steps. However, LFZip is still not as good as our solution. The key reason why our solution has such a high compression ratio is that we investigate and utilize the MD data features in both spatial and temporal dimensions (as shown in Figure 3, Figure 4, Figure 5, and Table II).

2) Rate-Distortion: Rate-distortion graph is one of the main assessment metrics of lossy compression quality. Rate-distortion involves bit rate and PSNR. The bit rate is defined as the average bits per data point of the compressed data. PSNR is the peak-signal-to-noise ratio and it is inversely proportional to mean squared error. Lower bit rate or higher PSNR indicts better compression quality. Figure 13 presents the rate-distortion results of all the lossy compressors. It is clear that our solution has the best PSNR given the same bit rate (about 20dB improvement in most cases), and also has the lowest bit rate given the same PSNR (about 50% reduction in size in most cases).

3) Compression Error: In the domain of lossy compression, compression error is defined as the differences between the decompressed data and the original data. The maximum of the compression error (MaxError) and the normalized root-mean-square error (NRMSE) are two key metrics to evaluate the compression quality of lossy compressors. As an example, we present in Table VI the two error metrics for all the lossy compressors (excluding MDB) based on the Copper-B dataset. Other datasets exhibit the similar results. MDB is excluded from this section because it could not achieve a compression ratio of 10. In this example, the MaxError of ADP always matches the lowest one from VO, VQT, and MT because VQ is always the best on x/y-axis, and MT is always better than the others on z-axis. Thus ADP chooses VQ for x/y-axis and MT for z-axis all the time. In other cases when no compressor is always better than the others, ADP will have even lower MaxError and NRMSE than any of the three compressors. We can observe from Table VI that our solution has the lowest MaxError and NRMSE on all axes. Specifically, the MaxError of our solution is 87%, 87%, 60% lower than the second-best compressor on x, y, and z axis, respectively.

To further demonstrate that our solution upholds the physical characteristics of the data after compression, we present the radial distribution functions (RDFs) of the original data and decompressed data in Figure 14. RDF, denoted as $g(r)$, is a critical analysis metric, which represents the possibility of finding a particle from the base particle at distance r. RDF is proportional to the local density of the particle systems. Figure 14 reveals that only our solution could yield the correct RDF on Cooper-B dataset under the same compression ratio. Therefore, only our solution delivers the decompressed data with unaltered local density to downstream applications. In order to get the same RDF as ours, other compressors need to significantly reduce their compression ratios. In summary, the RDF result proves that, with suitable compression ratio, our solution maintains the physical characteristics of the data accurately.

TABLE VI
MAXERROR AND NRMSE OF DECOMPRESSED COPPER-B DATASET, CR=10, BS=10

Type	Axis	SZ2	ASN	TNG	HRTC	LFZip	OurSol			
							VQ	VQT	MT	**ADP**
Max Error	X	0.37	0.23	0.44	2.06	0.35	0.03	0.10	0.10	**0.03**
	Y	0.32	0.23	0.44	1.94	0.35	0.03	0.10	0.09	**0.03**
	Z	0.16	0.10	0.44	1.13	0.17	0.11	0.05	0.04	**0.04**
NRMSE (×1E-4)	X	45.5	27.4	61.8	133	43.2	3.10	9.00	11.9	**3.10**
	Y	40.0	28.5	61.9	128	43.3	3.10	8.92	10.0	**3.10**
	Z	20.6	9.41	45.2	74.1	21.4	15.3	8.18	5.32	**5.32**

4) Compression/Decompression Throughput: We present the throughput comparison among all lossy compressors in Figure 15. It is clear that our solution is one of the fastest among all the lossy compressors on all datasets. As a comparison, ASN is slower than some compressors on Pt and Helium-B datasets. There are no results for TNG and HRTC on datasets such as Pt due to runtime exceptions, as explained in Section VII-A5. The results of LFZip are barely visible in this figure because LFZip has intermediate disk operations which bring significant runtime overhead. The excellent performance of our solution is attributed to both effective prediction methods and our optimized quantization settings (see Section VI-C1 for details).

D. Integration with LAMMPS

We integrate MDZ into the MD simulation software LAMMPS. To enable MDZ, LAMMPS users only need to adjust the data dumping option in the configuration file.

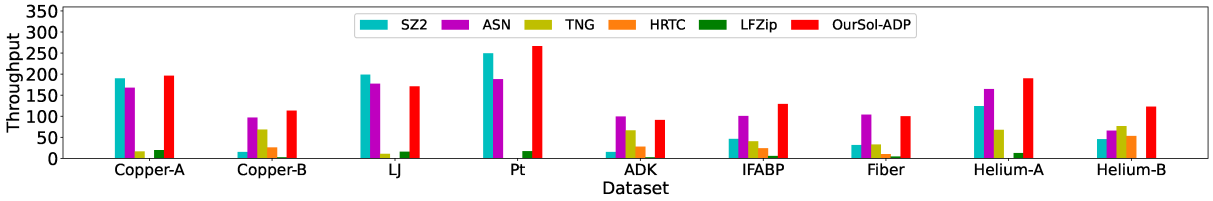
We executed the Lennard-Jones benchmark in LAMMPS with different settings to evaluate the overhead of our solution in real-world MD systems. The simulation lasts 1 million timesteps and is executed in three different scales, with the number of atoms ranging from 64K to 4096K. We choose data saving frequencies of 1 per 100 timesteps and 1 per 5000 timesteps, which is the range of the typical data saving frequency of MD simulations, as discussed in Section IV. The runtime breakdown is shown in Table VII. It is clear that enabling MDZ does not affect the output portion of the runtime or the total runtime. MDZ even improves the output performance when the data saving frequency is 100 because the I/O time is significantly reduced due to the reduced file size by MDZ. In general, MDZ has negligible overhead to the MD simulation, and it can improve the simulation performance if a large mount of data needs to be saved and the I/O speed is the bottleneck.

TABLE VII
RUNTIME BREAKDOWN OF LJ SIMULATION (F: DATA SAVING FREQUENCY, COMP: COMPUTATION TIME, COMM: COMMUNICATION TIME, OUTPUT: DATA SAVING TIME INCLUDING COMPRESSION)

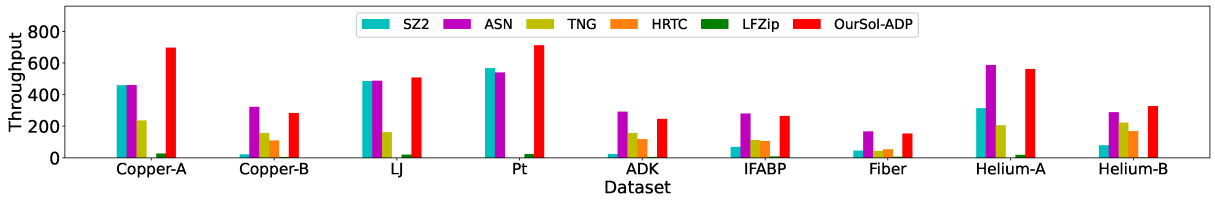
F	# Atoms	Option	Duration (minutes)	Runtime Breakdown		
				Comp	Comm	Output
100	64K	w/o MDZ	329	96.4%	1.4%	**2.2%**
		w MDZ	322	98.3%	1.4%	**0.3%**
	512K	w/o MDZ	428	93.9%	3.5%	**2.6%**
		w MDZ	418	95.5%	3.6%	**0.9%**
	4096K	w/o MDZ	516	82.7%	12.8%	**4.5%**
		w MDZ	513	83.0%	12.7%	**4.3%**
5000	64K	w/o MDZ	322	97.0%	2.9%	**0.06%**
		w MDZ	312	98.5%	1.5%	**0.01%**
	512K	w/o MDZ	415	96.2%	3.7%	**0.07%**
		w MDZ	425	93.7%	6.2%	**0.03%**
	4096K	w/o MDZ	474	90.0%	9.8%	**0.14%**
		w MDZ	480	88.9%	10.9%	**0.16%**

E. Generalizability of Our Solution Beyond MD Simulations

In this section, we discuss the generalizability of our solution beyond MD simulation datasets. As we mentioned in Section I, scientific data can be categorized into particle data, structured mesh, and unstructured mesh. In order to reach as high compression quality as possible, developers need to design specific solutions for each of the three categories of data. For example, our previous work [31] proposes a customized interpolation-based compressor for structured mesh data (the second category). In comparison, the solution proposed in this paper leverages both spatial and temporal data characteristics that exist in many domains. It can be applied to the first category of datasets (all kinds of particle data instead of only MD simulation data).



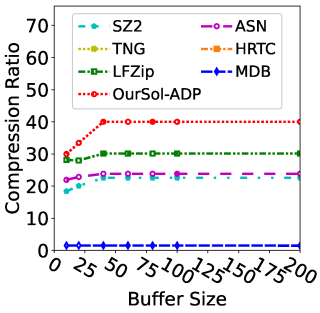
(a) Compression throughput



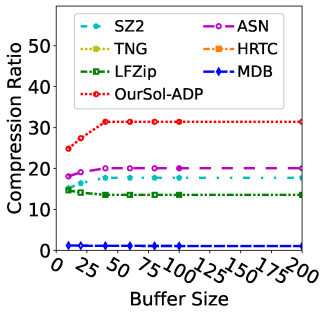
(b) Decompression throughput

Fig. 15. Our solution is the only one that always has high compression/decompression throughput (MB/s) on all datasets. As a comparison, ASN is slow on Pt and Helium-B. TNG and HRTC fail to run on some datasets and LFZip is very slow due its intermediate disk operations.

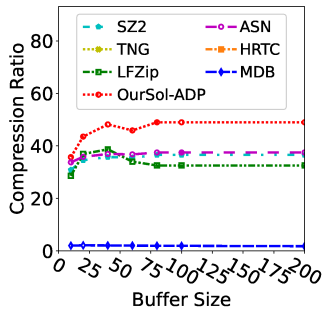
We present the compression ratio evaluation on the Hardware/Hybrid Accelerated Cosmology Code (HACC) datasets in Figure 16 to demonstrate the effectiveness of our solution in domains other than MD simulation. HACC is an extreme-scale cosmological simulation code that studies the structure formation in the Universe. HACC saves the positions and velocities of the particles periodically. We include two HACC datasets in this evaluation (HACC-1: 30 snapshots $\times$ 15767098 atoms, HACC-2: 80 snapshots $\times$ 13131491 atoms). It is clear that our solution is the best among all the compressors on both of the datasets, and it has 30%$\sim$56% higher compression ratios than the second-best compressor.



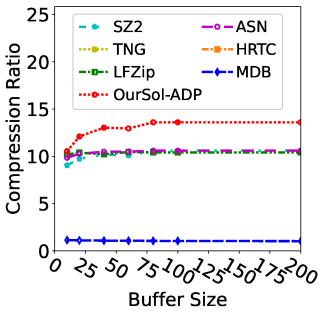
(a) HACC-1 position (ϵ=1e-4)



(b) HACC-1 velocity (ϵ=1e-3)



(c) HACC-2 position (ϵ=5e-4)



(d) HACC-2 velocity (ϵ=5e-4)

Fig. 16. Our solution has the best compression ratios on HACC datasets

VIII. Conclusions and Future Work

In this paper, we develop an efficient error-bounded lossy compressor (called MDZ) for MD simulation datasets. The key idea is significantly improving the prediction accuracy based on the regularities and correlations of the data in both spatial and temporal dimensions. We propose vector-quantization-based compressor VQ and VQT, and multilevel-time-based compressor MT. Our adaptive solution ADP can select the best compressor (VQ, VQT, or MT) dynamically at runtime. The key findings based on our experiments with eight real-world MD simulation datasets are summarized as follows.

- MDZ can improve the compression ratio by up to 233% compared with the second-best compressor on eight MD real-world MD simulation datasets.
- MDZ has the best data distortion among all the compressors. The RDFs confirm MDZ can maintain the physical characteristics of the data accurately.
- We integrate MDZ to the MD software LAMMPS. MDZ shows negligible overhead in real-world MD simulations under different scales and settings.
- The generalizability experiments show MDZ is capable of handing particle datasets beyond MD simulations.

In the future, we plan to integrate MDZ in more state-of-the-art MD packages such as CHARMM and EXAALT, so that the MD researchers can use it very conveniently.

IX. Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, and by DOE's Advanced Scientific Research Computing Office (ASCR) under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant No. 1617488, No. 2003709, and No. 2104023/2104024. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

[1] N. Tchipev *et al.*, "Twetris: Twenty trillion-atom simulation," *The International Journal of High Performance Computing Applications*, vol. 33, no. 5, pp. 838–854, 2019.

[2] J. Huwald, S. Richter, B. Ibrahim, and P. Dittrich, "Compressing molecular dynamics trajectories: Breaking the one-bit-per-sample barrier," *Journal of Computational Chemistry*, vol. 37, no. 20, pp. 1897–1906, 2016.

[3] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *SIGMOD Rec.*, vol. 34, no. 4, p. 34–41, Dec. 2005.

[4] Y. Cheng and F. Rusu, "Parallel in-situ data processing with speculative loading," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, p. 1287–1298.

[5] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou, "The researcher's guide to the data deluge: Querying a scientific database in just a few seconds," *Proc. VLDB Endow.*, vol. 4, no. 12, p. 1474–1477, Aug. 2011.

[6] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki, "Nodb: Efficient query execution on raw data files," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12, 2012, p. 241–252.

[7] ORNL Summit supercomputer, https://www.olcf.ornl.gov/summit/, 2021, online.

[8] A. Omeltchenko, T. J. Campbell, R. K. Kalia, X. Liu, A. Nakano, and P. Vashishta, "Scalable i/o of large-scale molecular dynamics simulations: A data-compression algorithm," *Computer Physics Communications*, vol. 131, no. 1, pp. 78–85, 2000.

[9] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. R. Pugmire, M. Wolf, N. Podhorszki, and S. Klasky, "Mgard+: Optimizing multilevel methods for error-bounded scientific data reduction," *IEEE Transactions on Computers*, no. 01, pp. 1–1, jul 5555.

[10] J. Tian *et al.*, "CuSZ: An efficient gpu-based error-bounded lossy compression framework for scientific data," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '20, 2020, p. 3–15.

[11] S. Li, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Resilient error-bounded lossy compressor for data transfer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21, 2021.

[12] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen, and F. Cappello, "Exploring autoencoder-based error-bounded compression for scientific data," https://arxiv.org/abs/2105.11730, 2021, online.

[13] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "Sdrbench: Scientific data reduction benchmark for lossy compressors," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2716–2724.

[14] D. Y. Yang, A. Grama, and V. Sarin, "Bounded-error compression of particle data from hierarchical approximate methods," in *Proceedings of the 1999 ACM/IEEE Conference on Supercomputing*, ser. SC '99, New York, NY, USA, 1999, p. 32–es.

[15] M. Lundborg, R. Apostolov, D. Spangberg, A. Gardenas, D. van der Spoel, and E. Lindahl, "An efficient and extensible format, library, and api for binary trajectory data from molecular simulations," *Journal of computational chemistry*, vol. 35, 01 2014.

[16] S. Li, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression with adjacent snapshots for n-body simulation data," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 428–437.

[17] J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka, and L. Vandevenne, "Brotli: A general-purpose data compressor," *ACM Trans. Inf. Syst.*, vol. 37, no. 1, Dec. 2018.

[18] Zstandard, https://github.com/facebook/zstd/releases, 2021, online.

[19] T. Pelkonen *et al.*, "Gorilla: A fast, scalable, in-memory time series database," *Proc. VLDB Endow.*, vol. 8, no. 12, p. 1816–1827, Aug. 2015.

[20] X. Yu *et al.*, "Two-level data compression using machine learning in time series database," in *36th IEEE International Conference on Data Engineering*, 2020, pp. 1333–1344.

[21] S. K. Jensen, T. B. Pedersen, and C. Thomsen, "Modelardb: Modular model-based time series management with spark and cassandra," *Proc. VLDB Endow.*, vol. 11, no. 11, p. 1688–1701, Jul. 2018.

[22] S. K. Jensen, T. B. Pedersen, and C. Thomsen, "Scalable model-based management of correlated dimensional time series in modelardb+," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 1380–1391.

[23] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, 2003, pp. 429–440.

[24] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, and W. Zwaenepoel, "Online piece-wise linear approximation of numerical streams with precision guarantees," *Proc. VLDB Endow.*, vol. 2, no. 1, p. 145–156, Aug. 2009.

[25] N. Agrawal and A. Vulimiri, "Low-latency analytics on colossal data streams with summarystore," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17, 2017, p. 647–664.

[26] H. Cao, O. Wolfson, and G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds," *The VLDB Journal*, vol. 15, pp. 211–228, 01 2006.

[27] C. Long, R. Wong, and H. Jagadish, "Trajectory simplification: On minimizing the directionbased error," *Proceedings of the VLDB Endowment*, vol. 8, pp. 49–60, 01 2014.

[28] Z. Fang, Y. Du, L. Chen, Y. Hu, Y. Gao, and G. Chen, "E2dtc: An end to end deep trajectory clustering framework via self-training," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 696–707.

[29] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018, pp. 617–628.

[30] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[31] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 1643–1654.

[32] J. Dvořák, M. Maňák, and L. Váša, "Predictive compression of molecular dynamics trajectories," *Journal of Molecular Graphics and Modelling*, vol. 96, p. 107531, 2020.

[33] P. Marais, J. Kenwood, K. C. Smith, M. M. Kuttel, and J. Gain, "Efficient compression of molecular dynamics trajectory files," *Journal of Computational Chemistry*, vol. 33, no. 27, pp. 2131–2141, 2012.

[34] A. Shkurti, R. Goni, P. Andrio, E. Breitmoser, I. Bethune, M. Orozco, and C. A. Laughton, "pypcazip: A pca-based toolkit for compression and analysis of molecular simulation data," *SoftwareX*, vol. 5, pp. 44–50, 2016.

[35] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium*, 2017, pp. 1129–1139.

[36] X. Liang *et al.*, "SZ3: A modular framework for composing prediction-based error-bounded lossy compressors," https://arxiv.org/abs/2111.02925, 2021, online.

[37] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, and F. Cappello, "Significantly improving lossy compression for HPC datasets with second-order prediction and parameter optimization," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '20, 2020, pp. 89–100.

[38] D. Tao, S. Di, Z. Chen, and F. Cappello, "In-depth exploration of single-snapshot lossy compression techniques for N-body simulations," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 486–493.

[39] D. Perez, E. D. Cubuk, A. Waterland, E. Kaxiras, and A. F. Voter, "Long-time dynamics through parallel trajectory splicing," *Journal of Chemical Theory and Computation*, vol. 12, no. 1, pp. 18–28, 2016.

[40] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[41] LANL Grizzly, https://www.lanl.gov/org/ddste/aldsc/hpc/index.php, 2021, online.

[42] A. F. Voter, "Parallel replica method for dynamics of infrequent events," *Physical Review B*, vol. 57, 1998.

[43] LANL Trinity, https://www.lanl.gov/projects/trinity/, 2021, online.

[44] D. Perez, L. Sandoval, S. Blondel, B. Wirth, B. Uberuaga, and A. Voter, "The mobility of small vacancy/helium complexes in tungsten and its impact on retention in fusion-relevant conditions," *Scientific Reports*, vol. 7, 05 2017.

[45] S. Seyler and O. Beckstein, "Molecular dynamics trajectory for benchmarking MDAnalysis," 6 2017.
[46] Anton supercomputer, Available at https://www.psc.edu/resources/anton/ , 2021, online.
[47] O. Beckstein, "Molecular dynamics trajectory of I-FABP for testing and benchmarking solvent dynamics analysis," 9 2018.
[48] B. Brooks *et al.*, "Charmm: The biomolecular simulation program," *Journal of Computational Chemistry*, vol. 30, 2009.
[49] S. J. Plimpton, D. Perez, and A. F. Voter, "Parallel algorithms for hyperdynamics and local hyperdynamics," *The Journal of Chemical Physics*, vol. 153, no. 5, p. 054116, 2020.
[50] ANL Theta, https://www.alcf.anl.gov/theta, 2021, online.
[51] S. N. Laboratories, "Lennard-jones liquid benchmark on lammps," https://lammps.sandia.gov/bench.html#lj, 2021, online.
[52] J. E. Jones and S. Chapman, "On the determination of molecular fields ii. from the equation of state of a gas," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 106, no. 738, pp. 463–477, 1924.
[53] Bebop supercomputer, Available at https://www.lcrc.anl.gov/systems/resources/bebop , 2021, online.
[54] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 730–739.
[55] A. Grønlund, K. G. Larsen, A. Mathiasen, and J. S. Nielsen, "Fast exact k-means, k-medians and bregman divergence clustering in 1d," https://arxiv.org/abs/1701.07204, 2017, online.
[56] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. Berendsen, "Gromacs: fast, flexible, and free," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1701–1718, 2005.
[57] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 179–189.
[58] S. Chandak, K. Tatwawadi, C. Wen, L. Wang, J. Aparicio Ojea, and T. Weissman, "LFZip: Lossy compression of multivariate floating-point time series data via improved prediction," in *2020 Data Compression Conference (DCC)*, 2020, pp. 342–351.