Compression in Molecular Simulation Datasets

Anand Kumar, Xingquan Zhu[1], Yi-Cheng Tu, Sagar Pandit[*]

Department of Computer Science and Engineering, Department of Physics, University of South Florida, Tampa, FL 33620, U.S.A.

Abstract

Storage of large scientific data, such as molecular dynamics (MD) simulation measurements, for analysis is a challenging task. The requirements on disk space, input/output (I/O) and data transfer bandwidth are excessively high due to the large volume of data generated. Storage of data in compressed form has been a popular approach to address such issues. In this paper, we present a lossy compression framework that yields significant performance by combining the strength of principal component analysis (PCA) and discrete cosine transform (DCT). Though it is a lossy compression technique, the effect on analytics performed on decompressed data is very minimal. We show the effectiveness of the technique by presenting results on real MD simulation data. The data storage requirement is reduced by a large magnitude while achieving a compression ratio up to 13. The errors comparable to other existing techniques are achieved.

Keywords: molecular simulations, encoding, data compression, compression ratio, principal component analysis, discrete cosine transform.

[*]Corresponding author. E-mail: `pandit@cas.usf.edu`
[1]Florida Atlantic University, Boca Raton, FL 33431, U.S.A.

1. Introduction

Large-scale scientific data management has become an important task, since scientific discoveries are more dependent on storage and analysis of data than ever before. The explosive growth of scientific data is mainly driven by two factors. First, the high-performance sensing devices today can measure and observe natural systems in very high resolution and efficiency. For example, a typical next-generation sequencer can generate 600GB to 6TB of genome sequence data over a period of only 2 to 3 days [1]. There are also non–biological applications where data are abundant – the Large Hadron Collider [2] generates raw data at a rate of 320MB/sec.

Many scientific disciplines, such as biochemistry [2], astronomy [3], and material sciences [4], are undergoing a radical change in research methodology from conducting "wet-bench" experiments to performing computer simulations. As a result, the particle simulations (PS) have seen tremendous efficiency improvements in the last decade. In PS, the system of interest (e.g., a protein and its environment) is studied as a collection of large number of basic components (e.g., atoms) whose behavior can be completely described by classical physics. Often such simulations generate spatio–temporal data that are in tera to peta bytes in size. With increasing size of data the problem of efficient storage and transfer persists. This paper addresses these issue in spatio-temporal data generated from PS applications by proposing effective data compression techniques.

[2]http://lhc.web.cern.ch/lhc

1.1. Motivation

The PS data is obtained from simulation results of a biological, physical or chemical phenomenon. Figure 1 shows part of a model bilayer system simulation. In this system, all individual atoms (we use atom and particle interchangeably) together represent large biological structures. Thus, providing nano-scopic description of biological process. The simulation consists of measuring properties such as, 3-D location of the atoms, velocity, charge, mass etc. at very small intervals of time (pico-seconds). Measurements of all atoms taken at a time instant, called snapshot (or *frame*), are stored on to computer disk. Considering the simulation for few microseconds, the data generated can easily reach terabytes. Since the simulation is generally done in large computer clusters (e.g., those in National Center for Supercomputing Applications), data needs to be transferred to the storage server (e.g., those in a scientist's own lab). On the server end, data analytics face bottleneck due to limited I/O bandwidth. The above facts necessitate the compression of data for better utilization of the storage devices and network transfer bandwidth. The MD and the data management communities have been primarily focused on the high-performance computing, visualization and simple data management, thus left the problem of MD data compression inadequately addressed.

Another property of the simulation data is that the changes of location for most of the atoms are small as a result of smaller time steps. The researchers are generally interested in analytical results on MD datasets. Therefore, some errors in the data due to compression and decompression would not affect the results of analytics significantly [5].

Traditionally, as the underlying simulation data contains the location and speed information of the particles (i.e., atoms), the textual data compression methods [6, 7], which use dictionary based approaches, are used to compress the data. The inherent disadvantage of such compression methods for particle simulation data is twofold:

- The resulting size of the compressed data can still be large, which adds high cost to I/O and network transfers;
- The compression method has to scan the whole data at once before it can begin compression.

In addition, the compressed data file can be completely corrupted (which makes decompression impossible), if one or few data bytes are corrupted or damaged.

1.2. Contributions

The disadvantage of the simple textual compression method can be overcome by a combination of different encoding-based compression technique for MD data as presented by Omletchenko *et al.* [5]. The technique uses space-filling curve (SFC) to sort the atom positions from the 3-dimensional space to a 1-dimensional space, followed by an adaptive and variable length encoding to achieve the compression. Another method by Meyer *et al.* [8] uses principal component analysis (PCA) [9] based method to achieve lossy compression. These methods do not consider the temporal locality of the atoms for compression, which leaves a significant amount of redundancy in the compressed data. We address these problems in this paper by presenting

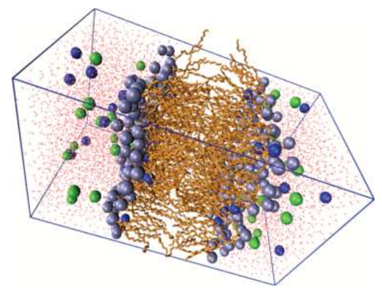

Figure 1: A simulated hydrated dipalmitoylphosphatidylcholine bilayer system.

a new compression method for MD data. Our technique is designed to meet the following four goals.

- **High compression ratio and dynamic error control:** As the MD data can easily reach tera to peta bytes in size, a compression method should provide high compression ratio (> 5), without noticeable errors. In addition, a compression method should allow users to fully control the errors, so that the compression quality can always meet the predefined requirements.

- **Error tolerance and propagation:** Due to possible transmission errors or any other damage, the compressed data should be highly error tolerant and largely decompressed even if some data are corrupted. In addition, the compression method should ensure that the errors do not propagate across the data frames.

- **Random access to compressed data:** The compressed data should be randomly accessible so that the users can randomly access and decompress any particular frames without decompressing the whole data (which is computationally expensive and heavily time consuming).

- **Balanced compression across different dimensions:** Because MD data are recorded in 3-D for both location and speed, the compression method should ensure that the compression errors are balanced across all three dimensions, without diminishing the compression quality of each dimension significantly.

In this paper we focus on reduction of the data size from the perspective of disk storage and network transfer bandwidth, by achieving high compression ratio. In our framework, the MD data are first transformed, using PCA (section 3), from the generic 3-D coordinate space to another 3-D eigen space, with the dimensions sorted in decreasing importance levels in capturing the variance of the atoms' movements. In the eigen space, the DCT is applied (section 4) to achieve lossy compression across a window of consecutive frames. The lossy compression does not affect the results of the analytics that are often executed on molecular simulation data [5, 10, 11, 12]. The combination of the PCA and DCT ensures that our framework can (1) achieve balanced compression across 3-D coordinate space, (2) realize dynamic error control and avoid the propagation of the compression errors and data corruption; and (3) ensure random access to any portion of the data without fully decompressing the whole data file. Random access to a window of frames, compressed together, is possible. However, random access to single frame from each compression window results in decompression of the whole data.

2. Related Work

Popular compression tools like WinZip and Gzip use dictionary based methods (such as Ziv-Lempel [6]). Statistical methods like Huffman encoding [13] and arithmetic encoding [7] are used in compression of the multimedia data. These methods are either suitable for large text data or that exhibit certain statistical properties. The resulting size of the compressed data can still be large. Thus adding high cost to I/O and network transfers. There have been efforts to process approximate analytics using random sampling [10], histograms [11] and wavelets [12]. In these methods, the analysis is done on the synopses of the data. In wavelets based method, the wavelet decomposition is applied on the data to obtain compact synopses. The synopses comprise of a small collection of wavelet coefficients. The focus was mainly on the efficient data analysis while minimizing the I/O during execution time.

A number of compression techniques have been proposed for alphanumeric data. Some techniques compress each measurement separately [14]. A compression framework formed by composing primitive compression operators is presented by Chen *et al.* [15]. It considers compression of blocks of data from one or more measurements. The framework presented is used to compress the analysis results in a systematic way such that data transfer over network is efficient. This helps in choosing appropriate plans for data analysis also. However, these techniques could not achieve high compression ratios as the underlying encoding techniques are similar to the traditional techniques. A detailed survey of the traditional data compression techniques is provided by Salomon *et al.* [16].

A combination of different encoding-based compression techniques for molecular dynamics (MD) data is presented by Omletchenko *et al.* [5]. The technique presented uses an oct-tree index to sort the atom positions on a space-filling curve (SFC). Then, the sorted atom coordinates are encoded using adaptive and variable length encoding to handle exceptional values. It is purely an encoding technique that uses spatial locality of the atoms to achieve compression. Another approach to compression of trajectories and data management is provided by Essential Dynamics (ED) tool [8]. It is similar to compression of data using principal component analysis (PCA) [9]. The data are transformed into an orthogonal space defined by the eigenvectors obtained by diagonalization of the covariance matrix. The size of the compressed data (and compression ratio) is determined by the number of selected principal eigenvectors. The trajectory pattern and the number of eigenvectors determine the error produced in the uncompressed data. The ED tool does not consider the temporal locality of the atoms for compression. Hence, the achievable compression is limited by the number of eigenvectors in the data. In molecular dynamics simulations, most atoms move along a trajectory of their own that changes very little over time. There is a huge scope for achieving high compression if this temporal locality is considered. We attempt to achieve better compression in our approach by considering the temporal locality.

3. Principal Component Analysis (PCA)

The principal component analysis (PCA) is used to find different directions, that represent different variances available in the data. In most of the compression techniques, the variance is useful for encoding. The compression techniques find frequently appearing values (that can be obtained using variance) so that those can be encoded using less number of bits, as compared to the non-frequent values. In this section we provide a brief overview of the PCA and explain the intuition behind application of PCA for compression. Let $\mathbf{x}_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,d}]^T \in \mathbb{R}$ be the vector of d measurements of particles (atoms) obtained at time instance t during the simulation process. The measurements obtained at every time instant add a new row of information and grow to a $t \times d$ matrix $\mathbf{X}_t = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t]^T \in \mathbb{R}^{t \times n}$. In the MD data, $\mathbf{x}_t$ is the column vector of all measurements of single particle at time t. There are d number of measurements obtained for that particular particle. Such measurements are made for every particle in the system. In this discussion we explain by focusing on only one particle.

In the measurements obtained, there exists correlation between the d dimensions. Capturing these correlations helps to identify the directions in which maximum variance of the data is observed. The variance information is used to encode the data (section 5). The correlations can be captured by principal component analysis (see an example in figure 2). If we project every point onto a line passing through a direction v_1, the error of this projection is very small. In general, the principal component v_1 is defined as below.

Definition 1. : *Given a collection of d-dimensional vectors* $\mathbf{x}_r = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d$, $r = 1, 2, \ldots, t$, *the first principal direction* $\mathbf{v}_1 \in \mathbb{R}^d$ *is the vector that minimizes the sum of the squared distances* $J_1(\mathbf{v}_1)$ *between* $\mathbf{v}_1$ *and the various* $\mathbf{x}_t$.

$$\mathbf{v}_1 = \underset{\|\mathbf{v}_1\|=1}{arg\ min}\, J_1(\mathbf{v}_1) \tag{1}$$

where the squared-error criterion function $J_1(\mathbf{v}_1)$ *is given by,*

$$J_1(\mathbf{v}_1) = \sum_{r=1}^{t} \| \mathbf{v}_1 - \mathbf{x}_r \|^2 \tag{2}$$

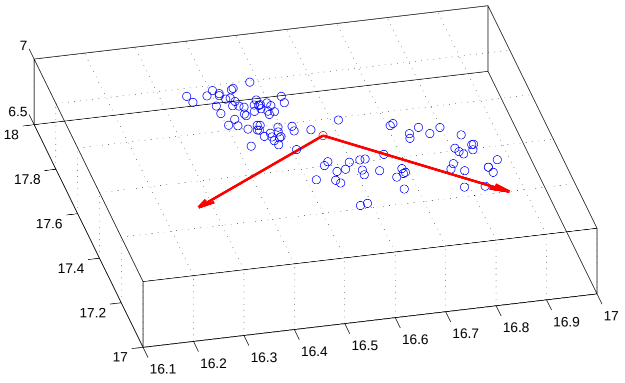

Figure 2: Principal Component Analysis: Two principal components (marked by thick lines) in the trajectory of an atom.

The projection of $\mathbf{x}_r$ *on to a line passing through the data mean in the principal direction is called the* **principal component***, given by* $y_{r,1} = \mathbf{v}_1^T \mathbf{x}_r, r = 1, 2, \ldots, t$.

The principal component can be projected back to get the original data, i.e., $\mathbf{x}_r$. Since $\parallel \mathbf{v}_1 = 1 \parallel$ the original d-dimensional data is obtained by the projection $\tilde{\mathbf{x}}_r = y_{r,1}\mathbf{v}_1$. However, the reconstruction $\tilde{\mathbf{x}}_r$ is not perfect. There is loss of some information due to the projection onto a single principal vector $\mathbf{v}_1$. The PCA can produce $k \leq d$ vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ such that we can represent the d-dimensional data in k dimensions by projecting onto the k principal components (vectors), $\mathbf{y}_t = [\mathbf{v}_1^T \mathbf{x}_r, \mathbf{v}_2^T \mathbf{x}_r, \ldots, \mathbf{v}_k^T \mathbf{x}_r]^T$, $r = 1, 2, \ldots, t$. The mean squared error between the original and reconstructed data (given by equation 3) can be minimized by selecting more vectors to project onto.

$$P_{err} = \sum_{r=1}^{t} \parallel \mathbf{x}_r - \tilde{\mathbf{x}}_r \parallel^2 \tag{3}$$

Some Observations:. Following are some observations that can be made about principal component analysis.

- When $k = d$, using d number of principal components, the reconstruction error $P_{err} = 0$.
- When the number of principal components $k \ll d$, we can represent the complete data using few dimensions. The storage space required for these dimensions is less than that required for the original data.

The space and time requirements of estimating PCA depend upon the time instant t. We partition the data into windows along the time t domain to compute the PCA faster. The measurements of every particle in the MD data is partitioned into windows of fixed size. Each window is processed separately to get the principal components. The principal components are actually the eigen vectors of $\mathbf{X}_t\mathbf{X}_t^T$. Different techniques are available to compute the eigen vectors. Singular value decomposition (also known as SVD) [9] is one of the efficient ways to compute.

PCA in Spatio-Temporal Data:. The PCA is used as a tool to obtain the direction of maximum variance in the data. The data is projected onto the principal components after the analysis. The eigen values $e_1, e_2, \ldots, e_d$ and eigen vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ are computed from $\mathbf{X}_t\mathbf{X}_t^T$. The eigen vectors are sorted in decreasing order to obtain the principal components in the corresponding order. The data shows maximum variance along the direction of component represented by highest eigen value and minimum along the component of least eigen value. The maximum variance direction gives large number of frequently occurring values, which can be encoded using small number of bits. Thus giving scope for better compression. A lossy compression with controllable error can be performed by applying discrete cosine transform and other encodings.

4. Discrete Cosine Transform and Particle localization properties

The discrete cosine transform (DCT) is often used in signal and image processing. Since it has strong *energy compaction* property, it is often used in data compression [17]. In this section we give a brief overview of DCT and intuition behind its use in our work.

The discrete cosine (DCT) transform is similar to the discrete Fourier transform (DFT). The DCT decomposes the input signal into a number of coefficients that can be encoded independently. Given an input signal with N values $\{x_0, x_1, \ldots, x_{N-1}\}$, a set of coefficients $\{X_0, X_1, \ldots, X_N\}$ is obtained by applying the cosine transform (given by equation 4). A few low frequency components of the DCT capture most of the signal information.

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], 0 \leq k \leq N-1 \tag{4}$$

The inverse DCT transform (iDCT) can exactly restore the original signal values from the coefficients (see equation 5). An interesting property of DCT is that, the signal can be reconstructed with very minimal error by rounding or loosing some precision of the DCT coefficients. Rounding off the coefficients (also called *quantization*) corresponding to certain frequencies gives many repeating values. The repeating values can be encoded very efficiently to get compressed data. A compression technique in which no coefficients

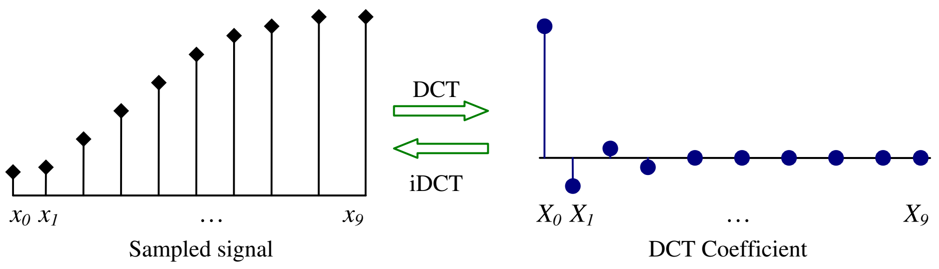

Figure 3: The discrete cosine transform (DCT) of a signal and its inverse.

loose precision gives lossless compression. However, to obtain high compression ratio, we drop (or quantize) insignificant DCT coefficients. As shown in Figure 3, the first three coefficients X_0, X_1, X_2 are more significant than the remaining coefficients. Dropping rest of the coefficients gives the lossy compression of the input signal. The use of DCT for MD data compression has two major advantages (1) the running time of compression is feasible (i.e., $O(N \log N)$); (2) the compression error can be adjusted by encoding desired number of DCT coefficients, allowing users to control the compression errors.

$$x_k = \sum_{n=0}^{N-1} \mathbf{X}_n cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right],\ 0 \leq k < N-1 \tag{5}$$

In the context of MD data we utilize the motion property of the particles to achieve compression. The movement of an atom is influenced by its interactions with other atoms in the vicinity. The movement can be decomposed as "slow" or "fast", each of which is a continuous function with respect to each atom. The speed and movement of atoms is highly localized in a sufficiently small time simulation. In addition, at any specific time, the atoms within a small range also have similar movement. The projection of data on to the principal components gives more number of particles that exhibit the locality property. The DCT can perform better with large number of data points along the principal component. As a combined result, the efficient encoding can be obtained to save space and achieve high compression ratio. We try to utilize this property of the atoms to compress the MD data.

5. Compression Framework

In general, the data compression methods can be categorized as either *lossy* or *lossless*. Lossless compression gives the exact copy of the original data after decompression, whereas the decompression of lossy compressed data does not. The choice of a particular type of compression is determined by the type of application that uses the data. In scientific datasets, most of the analytical results are acceptable within certain error bars. Hence, a lossy compression technique, with high compression ratio, and control over errors is a good choice.

The molecular dynamics simulation data is generated in large volumes, which is stored in single or multiple trajectory files containing time frames. Each frame is a sequential list of atoms with their positions, velocities, perhaps forces, masses and types. The data can be very large with up to millions of atoms and tens of thousands of frames. The disk storage required is minimized by applying compression technique explained in this section. In the following sections we discuss the the basic framework of compression along with the encoding techniques used. Figure 4 shows the framework of the proposed compression method. Our main theme is to employ PCA to transform data to another space from which lossy compression can be achieved using DCT.

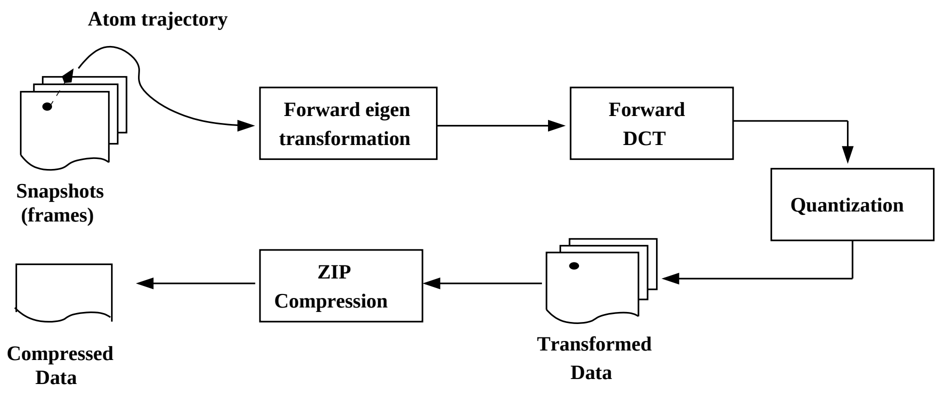

Figure 4: The framework for the compression of particle simulation data.

5.1. Transforming Data to Eigen Space

The measurement values captured during molecular simulation change over time as the data is stored after each snapshot. The measurements depict changes in certain properties of the atoms in the simulation. As the MD data is of large volume, we compress a group of snapshots (frames) called window. The number of frames in the window can be controlled depending on the availability of computational resources and the level of errors we are willing to tolerate in the final data. We apply orthogonal linear transformation to the data using principal component analysis (PCA) [9]. For each atom (as in Figure 5) we collect its locations across a number of adjacent frame, $f_1, f_2, \ldots, f_n$, each of which contributes a 3-D location (l_x, l_y, l_z). PCA transforms data to a new coordinate system (l_p, l_q, l_r) such that the greatest variance is observed on the first coordinate l_p after projection of the data. The coordinates are ordered in increasing order of the variance of the projection, which is obtained by ordering the eigen values and corresponding eigenvectors of the data. The transformation $\mathbf{F}$ of given data $\mathbf{D}$, with eigen-

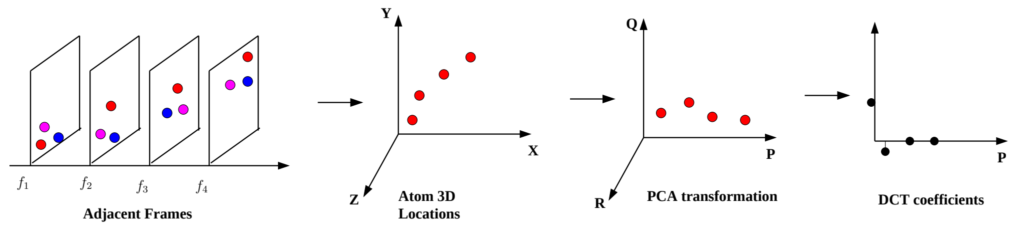

Figure 5: Intuition behind compression: Performing principal component analysis (PCA) followed by discrete cosine transform (DCT) on data.

vectors **E** in increase order of eigen values, using PCA is given by equation (6).

$$\mathbf{F}^T = \mathbf{D}^T \mathbf{E} \tag{6}$$

The eigen values and eigenvectors can be found from the covariance matrix of the data in the window. The original data **D** can be obtained back from the transformation using equation (7).

$$\mathbf{D}^T = \mathbf{E}^{-1} \mathbf{F} \tag{7}$$

The eigen space transformation aligns the data along eigenvectors based on the variance observed in the data. By transforming the data to the new coordinate system (P, Q, R), we can pick few dimensions that represent the complete data (compressed) while maintaining low errors (in decompressed data). This is lossy compression. In our approach we transform the data into new coordinates to obtain the different directions of variance. These directions are then compressed independently using DCT. Each dimension uses different DCT parameters based on the degree of variance for compression. This gives data with minimal loss to achieve better performance of the DCT compression (section 5.2).

5.2. Computing DCT of transformed data

The eigen analysis step transforms data into new dimensions [9]. We encode every dimension (attribute/measure) of the atoms separately. However, the eigen space transformation is applied to 3D trajectories of each atom separately. Finally, we apply the DCT [18] on each dimension P, Q, R of the transformed data. By transforming the data to the new coordinate system, we are trying to reduce the errors in all dimensions that may occur in performing the DCT step. This makes recovering DCT coefficients more accurate during decompression, as the coefficients in the beginning contain more accurate information. The DCT has been used in compression of multimedia content. It is being used as part of the JPEG image compression and video compression standard also [19].

The atom localization property is utilized by applying the DCT to a constant number of adjacent frames, say N. Given a window of size N (frames), we compress single measurement of every atom individually. The trajectory of length N is transformed using the DCT as given in equation (4). The trajectories of all the atoms in the window are transformed similarly. Now, the DCT data $\mathbf{C}$ is quantized to reduce the number of bits required to store the coefficients. The coefficients can be eliminated by assigning very low weights to the high frequency components. As a result, the coefficients are close to 0 value, the quantization step helps in achieving further compression of the data. Therefore, after all the data is processed, we apply the ZIP compression on the DCT data. As a lossless compression method, the ZIP compression removes the redundant information present in the data. It uses dictionary based methods to achieve this (e.g. LZ method [6]). This gives the final compressed data. We discuss the compression ratio and the error in uncompressed data in experiments section.

The combination of the PCA and DCT ensures that our framework can (1) achieve balanced compression across 3-D coordination space, (2) realize dynamic error control and avoid the propagation of the compression errors and data corruption; and (3) ensure random access to any portion of the data without fully decompressing the whole data file.

5.3. Decompressing The Data

The original data is obtained back from the compressed data by performing the operations of Figure 4 in reverse order. First, we unzip the data to obtained quantized data. Then the quantized data is again de-quantized by using the same window size W as used during the compression step. Then inverse DCT is applied (Equation (5)) on the de-quantized data to obtain eigen space transformed data. Finally, the original trajectories **T** are obtained by back projection of the principal components. The experimental results of the proposed compression framework are discussed in section 6.

5.4. Wavelets for Compression

Multi-resolution decomposition is another technique to capture different frequencies in the signals. The signal processing literature provides the wavelet transform tool for multi-resolution analysis. The wavelet transform is applied to capture different frequencies present in the signal. The transform is applied repeatedly, on the transformed signal, to get all frequencies present at different resolutions.

Table 1: The molecular simulation datasets used for the compression experiments.

Data set	Description	Objects	Data size
Protein	MD data of protein	286,849	3.8 GB
Fiber	MD data of collagen fiber	891,272	6.4 GB

The wavelet transform is reversible, like inverse DCT, to reconstruct the original signal. The reconstruction can be done to any required resolution level. The transform is represented as a set of coefficients. At each resolution the signal is represented in different coefficients. The low energy coefficients of the transform can be ignored or quantized to achieve compression. This is a lossy compression technique similar to the DCT based compression discussed before. Therefore, there is scope for using the discrete wavelet transform or DWT (wavelet transform for discrete signals) as a replacement to DCT in our compression technique.

6. Experimental Results and Discussion

The proposed compression technique is tested on real molecular dynamics datasets. We used two different datasets, as shown in table 1, for the experiments. The protein data set was obtained from a protein MD system of $286,849$ particles. The collagen fiber data was obtained from another MD system of $891,272$ particles. These data sets were obtained from snapshots of real molecular simulations. The complete data set had around 600 frames of snapshots. The measurements stored in the data are: x, y and z coordinates of the atoms, charge and mass during the simulation.

The effect of compression on the quality of the data is measured as the

Table 2: Compression using PCA and DCT. The compression ratio on different data sets using window size W = 128.

Data set	Data size	Compressed size	Compression ratio	RMSE (Å)
Protein	3.8 GB	293.45 MB	13.26	0.14
Fiber	6.4 GB	535.86 MB	12.23	0.09

error in the final decompressed data. The error is measured as the root mean square error (RMSE; equation (8)).

$$Error = \sqrt{\frac{1}{N_a \times N_f} \sum_{i=1}^{N_f} \sum_{j=1}^{N_a} \{F_c - F_u\}} \tag{8}$$

The distance between the original data frame F_u and the compressed data frame F_c (we call the data obtained after *decompression* as compressed) is used to compute the error. N_a is the number of atoms in every frame, and N_f is the total number of frames in the data set. We computed the RMSE on the locations of the objects present in the system.

The results shown in table 2 explain the performance of the proposed compression technique. In this method we are able to achieve high compression ratio. The results shown in table 2 are obtained with a window size of 128. The effect of window size on the compression ratio of the data is shown in table 3.

The compression of protein data using only DCT gives compression ratio of about 7.5. A decrease in compression ratio is observed by increasing the window size. Table 4 shows the effect of window size on the compression ratio. A balance between the errors and compression ratio is achieved by choosing window sizes between 32 and 128. The errors obtained in this method are

Table 3: Effect of window size on compression using PCA and DCT on the protein MD data.

Window size	Compressed data size	Compression ratio	RMSE (Å)
008	987.61 MB	3.94	0.25
016	603.29 MB	6.45	0.23
032	432.36 MB	9.00	0.21
064	353.75 MB	11.00	0.18
128	293.45 MB	13.26	0.14
256	386.03 MB	10.08	0.09
512	407.46 MB	9.55	0.05

similar to the proposed technique. The error introduced by the proposed technique in the compressed data are shown in table 3. It can be seen that the amount of error introduced in the compressed data is small. Choosing a right window size, therefore, depends on the required compression ratio and the acceptable error rates.

The compression ratio directly depends on the amount of error that can be allowed to appear in the final data. The errors can be controlled by adjusting coefficients that are retained (as explained in section 5) after the DCT step. The error should increase with the compression ratio. This effect is observed in our dataset also. The plot shown in Figure 6(a) presents the relation between the compression ratio and the error. The RMSE increases as the compression ratio is increased.

The PCA and DCT based compression technique gives better compression ratio while maintaining low errors in the compressed data. This method is

Table 4: Compression ratio of the DCT technique using different window sizes.

Window size	Compressed data size	Compression ratio
008	508.65 MB	7.65
016	518.82 MB	7.50
032	533.04 MB	7.30
064	555.89 MB	7.00
128	595.00 MB	6.54
256	649.61 MB	5.99
512	720.59 MB	5.40

best suited for compression of very large volumes of data, which are common in the scientific data bases.

6.1. Data Analysis Results

The effect of compression on the data set can also be measured using the results of analytical queries. We applied some queries, interesting to researchers, on the decompressed data to measure this effect.

Mean square displacement (MSD). Consider a system with N particles. Let $\mathbf{r}_i(t)$ be the location of particle i, of mass m_i, at time instant t. The mean square displacement (MSD) information biophysicists ask frequently is given by (equation (9)).

$$MSD(t) = \frac{1}{N}\sum_{i=1}^{N} |\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2 \tag{9}$$

The proposed compression technique introduces very small error in the MSD. The plot of figure 6(b) shows the root mean square displacement of

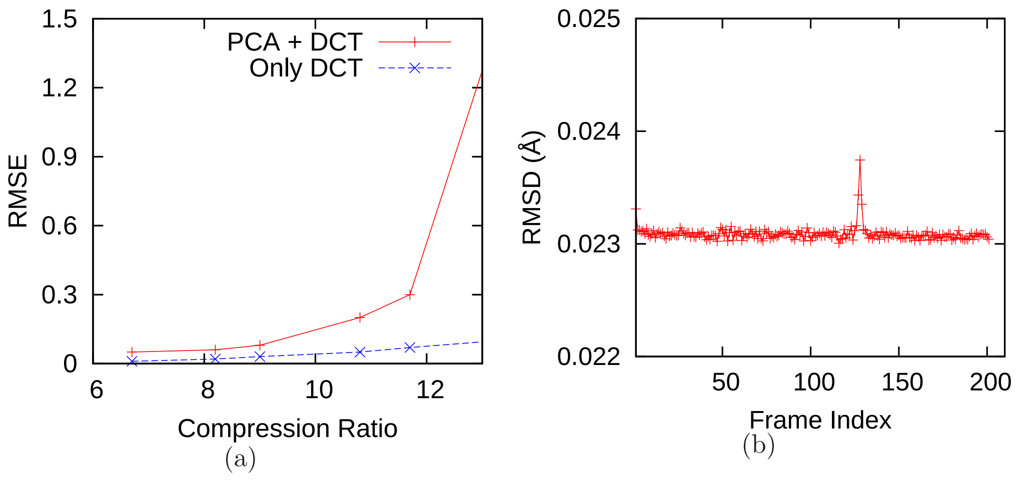

Figure 6: (a) Relation between the compression ratio and the error in compressed data. Average RMSE (in Å) is plotted against the compression ratio. Window size of 128 is chosen for compression (b) Effect of compression on the root mean square displacement (RMSD in Å) between original and decompressed frames.

the compressed frame from the original frame. A very small peak in the plot indicates the start of a new window used for the compression. Window size of 128 frames was used for the particular experiment.

Mass center (MC). Another information of interest to biophysicists is the mass center of the system at any given time instant. The mass center of the system at any time instant is given as in equation (10). Figure 7(a) shows the effect on the mass center $R(t)$ of the particle space at a given time instant t. It can be seen that the displacement in the mass center is very small. The small fluctuations in the plot are the effects of DCT coefficient quantization. This shows that the proposed method performs well in the compression and decompression of the simulation data.

$$R(t) = \frac{\sum_{i=1}^{N} m_i \mathbf{r}_i(t)}{\sum_{i=1}^{N} m_i} \tag{10}$$

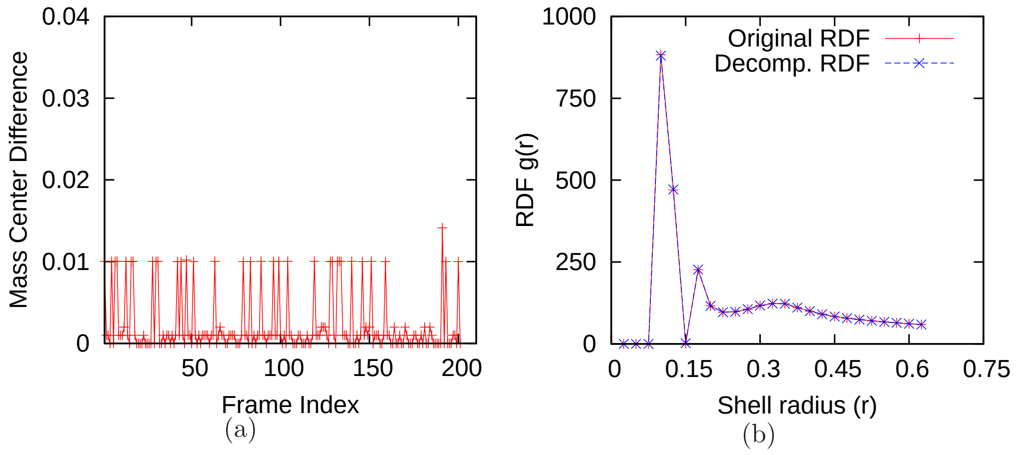

Figure 7: Effect of compression: (a) On center of mass (in Å). The difference between actual and decompressed mass centers is plotted. (b) On radial distribution function (RDF). Shell resolution $\delta r = 0.025$Å and window size of 128 frames was used.

Radial distribution function (RDF). The main motivation behind the proposed compression approach is to demonstrate the minimal effect on the analytical queries. We observed this behavior in the experimental results on radial distribution function (RDF [20]). The RDF is defined as in equation (11).

$$g(r) = \frac{N(r)}{4\pi r^2 \delta r \rho} \tag{11}$$

where $N(r)$ is the number of atoms in the shell between r and $r+\delta r$ around any particle, ρ is the average density of particles in the whole system, and $4\pi r^2 \delta r$ is the volume of the shell. The RDF can be viewed as a normalized spatial distance histogram (SDH [21]). The SDH is a fundamental tool in the validation and analysis of particle simulation data.

Figure 7(b) shows that the RDF is not much affected by the compression. The figure shows the beginning part of the RDF plot. The RDF values in the compressed and decompressed data are almost same for shell resolution $r > 0.025$Å. This shows that the error introduced by the compression in the RDF of the frames is minimal. Hence, the proposed technique is suited for compression of the molecular simulation data.

6.2. Discussion

The temporal locality of the atoms found in the trajectories is better utilized in the proposed compression technique, achieving high compression ratio. The PCA transforms the data into eigen space such that the encoding of the DCT coefficients along the eigenvectors gives better performance. Therefore, the proposed compression technique gives high compression ratio, with acceptable errors, as compared to the DCT-only compression technique (table 3). Moreover, the error rate in our method is acceptable, as specific values of atoms are of no interest to the researchers. Instead, the results of analytical queries on the complete dataset are of interest, which are not affected.

As mentioned in the earlier sections, considering the temporal locality of the atoms gives high compression ratios, which is not possible to achieve by using combination of different encoding methods [5] or only the PCA [8, 9]. The compression ratio achieved using the space-filling curve (by Omeltchenko *et al.* [5]) is limited by the encoding method applied and the volume of the boxes that enclose the atoms to map them onto the curve. As no errors due to volume of the SFC boxes are reported by Omeltchenko *et al.* [5], a direct comparison of our method cannot be made.

Compression with PCA only. For comparison purposes we used the PCA–only technique to measure the compression ratio on the protein dataset (table 1). Only top two eigen vectors were used to compress the data. A compression ratio of about 4.5 was achieved while producing maximum average RMSE in a dimension of about 1.2 and minimum of 0.07. The huge difference in error on different dimensions is an inherent feature of PCA, only the more important dimensions are kept correct. To compare our method with the PCA-only method, we combined the errors in all three dimensions together by calculating the displacement of the original particle from the decompressed coordinates. We measured a displacement of 0.9 in 3D space in the PCA-only experiments. The compression ratio obtained with this approach was about 5. However, our compression technique gives a displacement of about 0.05. It can also be observed that the proposed technique achieves good compression ratio while producing similar (or smaller) errors in the data.

One coarse way to compress data is by down sampling. Some of the frames, chosen randomly, from the data set are dropped to get the compressed data. The number of such random frames chosen is proportional to the compression ratio required. The major disadvantage of this approach is the significant loss of dynamics of the particles of the simulation system. Therefore, down sampling of the MD data is not feasible.

Compression of other measurements. The simulation data may also contain measurements such as charge, mass and velocity of the particles at a given time instant. The charge and mass values do not change or change very rarely in a simulation. Hence, any encoding technique (dictionary based) can be applied to achieve high compression. The velocity values also exhibit the change patterns as that of position values. Velocity values differ by small fractions in short time intervals (i.e., close frames). Therefore, the proposed compression technique can also be applied to store the velocity measurements.

Algorithmic complexity. The algorithmic time complexity to compress a data of N atoms and F frames using space-filling curves [5] is $O(NF\log(NF))$. The complexity to compress, with window size W, using proposed method is $O(NFW^2 + NF\log W)$. The PCA is computed for trajectory of every atom, which requires $O(FW^2)$ operations. For N data points, the fast DCT transformation can be done in $O(N\log N)$ operations [22]. In our approach, DCT has to be performed for each particle over W frames. Therefore, it can be computed in $O(NF\log W)$ time. The window size W is always a small constant. As a result, the computational overhead is not significant.

7. Conclusions and Future Work

A lossy compression technique for storage of large volumes of molecular simulation data is presented in this paper. The proposed technique uses eigen space transformation (using PCA) and DCT to compress the data. Compression ratio of about 13 is achieved using the proposed technique. A comparison of the DCT and PCA based compression techniques is made to analyze the feasibility and select appropriate steps for the compression.

The proposed technique exploits the temporal locality of the atoms to achieve good compression and hence is able to achieve very high degree of space utilization. The achieved compression can be used to satisfy the requirements of disk I/O and network bandwidth for the high volume of molecular simulation data.

A future direction of research would be to analyze the effects of compression ratio and the error bounds on the quality of the analytical results on the decompressed data. Some of the problems that need to be addressed in future are:

- The compression and decompression are computationally intensive tasks. Hence, measurements on the compressed data directly (without decompression) would be one of the interesting problems.
- Accessing every frame randomly without decompressing all frames from a compressed window of data.
- Need for an efficient encoding technique, in place of ZIP used here, to utilize the correlation between different trajectories.
- Data mining to find patterns of interest in the trajectories.

As an initial work in the topic of simulation data compression, the findings that we reported in this paper will definitely lead to abundant research efforts.

Acknowledgments. The authors Yi-Cheng Tu, Sagar Pandit and Anand Kumar are supported on NIH project R01GM0 86707 from the National Institute of General Medical Sciences (NIGMS).

References

[1] W. V. Etten, Managing data from next-gen sequencing, Genetic Engineering and Biotechnology News 28 (8).

[2] D. Frenkel, B. Smit, Understanding molecular simulation: From algorithm to applications, Computational Science Series 1.

[3] J. L. Stark, F. Murtagh, Astronomical image and data analysis, The Astronomy and Astrophysics Library Series.

[4] S. Klasky, B. Ludscher, M. Parashar, The center for plasma edge simulation workflow requirements., in: International Conference on Data Engineering Workshops (ICDE) Workshops, 2006, pp. 73–73.

[5] A. Omeltchenko, T. J. Campbell, R. K. Kalia, X. Liu, A. Nakano, P. Vashishta, Scalable i/o of large-scale molecular dynamics simulations: A data-compression algorithm, Computer physics communications 131 (1-2) (2000) 78–85.

[6] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Transactions on Information Theory 23 (3) (1977) 337–343.

[7] I. H. Witten, R. M. Neal, J. G. Cleary, Arithmetic coding for data compression, Commun. ACM 30 (6) (1987) 520–540.

[8] T. Meyer, C. Ferrer-Costa, A. Prez, M. Rueda, A. Bidon-Chanal, F. J. Luque, C. A. Laughton, M. Orozco, Essential dynamics: A tool for efficient trajectory compression and management, Journal of chemical theory computation (JCTC) 2 (2) (2006) 251–258.

[9] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, Wiley-Interscience Publication, 2000.

[10] W. G. Cochran, Sampling Techniques, 3rd Edition, John Wiley and sons, 1977.

[11] Y. E. Ioannidis, V. Poosala, Histogram-based approximation of set-valued query-answers, in: International Conference on Very Large Data Bases (VLDB), 1999, pp. 174–185.

[12] K. Chakrabarti, M. Garofalakis, R. Rastogi, K. Shim, Approximate query processing using wavelets, The VLDB Journal 10 (2-3) (2001) 199–223.

[13] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the Institute of Radio Engineers 40 (9) (1952) 1098–1101.

[14] G. Ray, J. R. Haritsa, S. Seshadri, Database compression: A performance enhancement tool, in: International Conference on Management of Data (COMAD), 1995, p. 0.

[15] Z. Chen, P. Seshadri, An algebraic compression framework for query results, Proceedings of the international conference on data engineering (ICDE) (2000) 177–188.

[16] D. Salomon, Data Compression: The Complete Reference, 2nd Edition, Springer-Verlag, 2004.

[17] K. Rao, P. Yip, Discrete cosine transform: algorithms, advantages, applications, Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[18] N. Ahmed, T. Natarajan, K. Rao, Discrete cosine transfom, IEEE Transactions on Computers C-23 (1) (1974) 90–93.

[19] Digital compression and coding of continuous-tone still images.
URL `http://www.w3.org/Graphics/JPEG`

[20] M. Bamdada, S. Alavib, B. Najafic, E. Keshavarzi, A new expression for radial distribution function and infinite shear modulus of lennard-jones fluids, Chemical Physics 325 (2006) 554–562.

[21] Y.-C. Tu, S. Chen, S. Pandit, Computing distance histograms ef?ciently in scientific databases, in: International Conference on Data Engineering (ICDE), 2009, pp. 796–807.

[22] Y. Arai, T. Agui, M. Nakajima, A fast dct-sq scheme for images, IEICE Transaction 71 (11) (1988) 1095–1097.