# Identifying Latent Reduced Models to Precondition Lossy Compression

Huizhang Luo[1], Dan Huang[1], Qing Liu[1], Zhenbo Qiao[1], Hong Jiang[2],
Jing Bi[3], Haitao Yuan[4], Mengchu Zhou[1], Jinzhen Wang[1], and Zhenlu Qin[1]

[1] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, USA
[2] Department of Computer Science and Engineering, University of Texas at Arlington, USA
[3] Faculty of Information Technology, Beijing University of Technology, China
[4]School of Software Engineering, Beijing Jiaotong University, China

***Abstract*—With the high volume and velocity of scientific data produced on high-performance computing systems, it has become increasingly critical to improve the compression performance. Leveraging the general tolerance of reduced accuracy in applications, lossy compressors can achieve much higher compression ratios with a user-prescribed error bound. However, they are still far from satisfying the reduction requirements from applications. In this paper, we propose and evaluate the idea that data need to be preconditioned prior to compression, such that they can better match the design philosophies of a compressor. In particular, we aim to identify a reduced model that can be utilized to transform the original data to a more compressible form. We begin with a case study of *Heat3d* as a proof of concept, in which we demonstrate that a reduced model can indeed reside in the full model output, and can be utilized to improve compression ratios. We further explore more general dimension reduction techniques to extract the reduced model, including principal component analysis, singular value decomposition, and discrete wavelet transform. After preconditioning, the reduced model in conjunction with difference between the reduced model and full model is stored, which results in higher compression ratios. We evaluate the reduced models on nine scientific datasets, and the results show the effectiveness of our approaches.**

***Index Terms*—High-performance computing, data reduction, reduced model, data precondition.**

## I. Introduction

Simulation-based scientific discovery produces extreme volumes of data that capture new physics in high fidelity [1], [2]. For example, for fusion XGC [3], a high-fidelity first-principle simulation to model ITER-scale[1] fusion devices, it generates more than 1 TB of particle data per snapshot, with the total analysis output easily reaching PBs for one campaign. More recently, for climate modeling, it was estimated that running large ensembles of high-fidelity simulations on future exascale systems would generate 260 TB for every 16 seconds [4], [5], which translates to 1.4 EB per day. These science requirements have completely changed the status quo of scientific data management on large high-performance computing (HPC) systems. Data reduction, for a long time playing a performance improvement role that is often secondary and dispensable, has become a pivotal step in scientific processes to allow science campaigns to be done within the resource and time constraints. To date, researchers in HPC have proposed various approaches to reducing data, such as compression [6]–[10], data deduplication [11], delta snapshot [12], and more broadly, in-situ methods [13]. Among them, data compression is recognized as a fundamental technique that complements other layers in the software stack. In general, both lossless and lossy compressors are used to compress scientific data. Compared to lossless compressors that preserve the exact content of original data, lossy compressors yield much higher compression ratios by trading accuracy for performance. Despite the recent success in lossy compression, e.g., by ZFP [8] and SZ [9], the reduction ratios are still far from what are ultimately demanded by applications. A root cause is that floating-point compressors are typically designed around the presumptive local smoothness in data, which may not be true for all datasets.

In this work, complementary to the existing efforts, we explore the idea that data are transformed prior to compression such that they can better match the design philosophies (e.g., local smoothness) of a compressor. The intuition behind our strategies arises from the concept of preconditioning in solving a linear system, $Ax - b = 0$, to achieve an improved rate of convergence, e.g., using a preconditioned conjugate gradient method [14]. The goal of this work is to capture the latent characteristics or representations of data, termed as **reduced model**[2], and transform data in a way that compression ratios can be improved. A simple example that illustrates our methodology is when compressing a signal that consists of a sine wave and white noise, we can identify the parameters (amplitude, frequency, and phase) of the sine wave first, and only compress the white noise, which is intuitively smoother, thus being more compressible than the original data. In our prior work, *DuoModel* [15], a simple reduced model is constructed by running a light version of the application with enlarged grid spacing. However, this approach requires extra compute nodes and involves communications with simulations, a key concern for its applicability to large-scale runs.

In this work, we aim to address the aforementioned weak-

[1]https://www.iter.org/

[2]In the context of this paper, the terms of reduced model data and reduced representation are interchangeable.

ness by synthesizing a reduced model directly from the analysis data, as opposed to from the simulation model. To understand the effectiveness of this methodology, we examine a set of techniques to obtain reduced representation, including principal component analysis (PCA), singular value decomposition (SVD), and discrete wavelet transform (Wavelet). We investigate the latent reduced models in the context of spatial outputs, which are typical of scientific simulations. Our solution stores the reduced model in conjunction with difference between the reduced model and full model (delta), which results in higher compression ratios. The original data can be re-computed by reconstructing the full model data and applying the delta. The major contributions of this paper are as follows:

- We synthesize the reduced model directly from the analysis output. Our *Heat3d* case study suggests that this approach results in higher compression ratios than compressing data directly or using *DuoModel*.
- We develop the methodology to precondition, compress, and reconstruct data, and conduct comprehensive performance evaluations for three popular dimension reduction techniques to precondition compression, including PCA, SVD, and Wavelet.

The remainder of this paper is organized as follows. Section II provides the background. Section III provides the motivation. Sections IV and V discuss how to identify the reduced model from analysis output using project-based methods and dimension reductions. Section VI presents the most related works. Conclusions are presented in Section VII.

## II. Background

### A. Lossy Compression

Lossy compression for floating point data has received renewed interest recently due to its higher compression performance. The main idea is to trade accuracy for performance, and use approximations and partial data discarding to represent the content. In another word, if two data are within an given error tolerance, they are regarded as the same. We next focus on two leading lossy compressors, ZFP [8] and SZ [9], which were shown to be superior in our prior work [16].

ZFP employs fixed-point integer conversion, block transform, and binary representation analysis with bit-plane encoding. It partitions a $d$-dimensional array into blocks of $4^d$ elements, and each block is compressed independently. Overall ZFP compression involves the following three steps: 1) It aligns the floating-point data within a block to a common exponent, and the original data in the block are then converted to mantissas along with the common exponent; 2) It converts the mantissas to fixed-point signed integers. A de-correlating transform, e.g., discrete cosine transform (DCT), is applied to generate near-zero coefficients that can be compressed efficiently; 3) It encodes the coefficients using embedded encoding.

In contrast, SZ, based upon polynomial predictions, compresses the delta between the prediction and the original value. It comprises four steps: 1) It predicts the next data point using a polynomial combination of its neighboring points; 2) If there is a prediction hit, an $m$-bit quantization factor is used to encode the point; 3) If there is a miss prediction, it performs binary representation analysis; 4) The quantized data are then further compressed using conventional compression techniques, such as Huffman encoding and LZ77, to remove the redundancy of quantization factors.

### B. Model Reduction

It is well recognized that the complexity of running large-scale simulations grows exponentially with the desired resolution and fidelity [17]. The purpose of model reduction in general is to lower the computational complexity so that the problem is more tractable, while achieving similar results. By scaling down a computational model, such as the degrees of freedom, spatiotemporal resolution, and precision, the convergence to the solution can be made faster and more efficient resource consuming. Nevertheless, model reduction must meet the following requirements in order to be relevant: 1) Loss of accuracy is acceptable. The increasing error as a result of model reduction must be within the prescribed tolerance; 2) Features in analytical data are still preserved. For knowledge discovery, the key properties of a full model must be retained by the reduced model; 3) Complexity must be substantially reduced. The reduced model must be orders of magnitude cheaper than the full model to be meaningful.

In general, there are two approaches to reducing a full model [17]. The first approach is to simply run a light version of the full model in less expensive settings, e.g., using a lower spatiotemporal resolution or precision. A key advantage of this approach is that no source code modification is required, thus greatly simplifying the model reduction. In fact, *DuoModel* [15] adopted this approach to demonstrate the usefulness of using reduced models to compress data. However, a potential weakness is that it may discard important physics that would otherwise be captured in the full model. For example, a lower resolution may discard important features that can be only seen at the original grid spacing. Another approach is **projection-based model reduction**, which builds a reduced model by projecting the equations of a full model onto a reduced coordinate system. It essentially transforms the full model into a lower-dimensional reduced model by removing the non-critical terms. Clearly, compared to the first approach, it takes major engineering efforts to implement the reduced model. However, it may better retain the underlying structure of the full model, since only non-essential terms are discarded. For example, the reduced model of *Heat3d*, a code that models how heat propagates in a closed system, can be built by projecting the full *Heat3d*. It is shown that the data features of the two models are highly similar, as shown in Section IV-A. This reveals that a reduced model can indeed retain a similar system state, motivating us to use it to precondition data prior to compression.

TABLE I: Dataset description.

| Dataset | Description |
|---|---|
| *Heat3d* | Distribution of heat in a given region over time. |
| *Laplace* | Description of steady state situations of values distributions. |
| *Wave* | Hyperbolic PDE for the description of waves. |
| *Umbrella* | Molecular dynamics simulations for Umbrella sampling. |
| *Virtual_sites* | Molecular dynamics simulation for virtual sites. |
| *Astro* | Velocity magnitude in a supernova simulation. |
| *Fish* | Velocity magnitude in a CFD calculation of cooling air being injected into a mixing tank. |
| *Sedov_pres* | Pressure of strong shocks in a hydrodynamical simulation. |
| *Yf17_temp* | Temperature in a computational fluid dynamics calculation. |

## III. Motivation

In this section, we identify the opportunity of synthesizing a reduced model directly from analysis output and use it to precondition compression. We first illustrate through nine applications that similarity does exist between full model and reduced model output, and then we discuss the disadvantages of the prior work *DuoModel* which further motivates this work.

### A. Similarity Between Full Model and Reduced Model

In order to understand whether a full model and a reduced model can yield similar data products, we investigate a set of double-precision floating-point datasets that are generated from real HPC applications. The details of the datasets are shown in Table I. In particular, a full model dataset is directly generated by the application itself, and the associated reduced model output is generated by scaling down the full model. For the three classical partial differential equation (PDE) datasets, i.e., *Heat3d*, *Laplace*, and *Wave*, we obtain the reduced model by scaling down the problem size. For example, for *Heat3d*, we set the number of points on each dimension of full (reduced) model as $192 \times 192 \times 192$ ($48 \times 48 \times 48$). For the two Gromacs[3] simulations, *Umbrella* and *Virtual_sites*, we scale down the problem complexity by lowering the number of atoms simulated. In particular, there are 1,960 atoms in the full model, and 490 atoms in the reduced model. For other four applications, we set a smaller size of computational domain and examine physical quantities at shorter times. For example, for *Sedov_pres*, we set the sizes of computational volume of full model and reduced model as $(1, 1, 1)$ and $(0.5, 0.5, 0, 5)$, and the maximum number of time steps to compute before halting the simulation as 20,000 and 10,000, respectively, to observe Courant-Friedrichs-Lewy (CFL) condition [18].

Fig. 1 demonstrates the similarity between the full model and reduced model for the nine datasets. Herein, we use cumulative distribution function (CDF), along with three scalar quantities, $byte\ entropy$, $byte\ mean$, and $serial\ correlation$ [16], to describe data characteristics. In particular, $byte\ entropy$ denotes entropy that measures the randomness of data, which ranges in [0, 8]. The closer the entropy value is to 8, the higher the entropy is. $Byte\ mean$ captures the arithmetic mean of data in bytes. This is simply the result of summing all the bytes of a dataset and dividing by the file length. If the data are fairly random, this metric should

[3]http://www.gromacs.org/

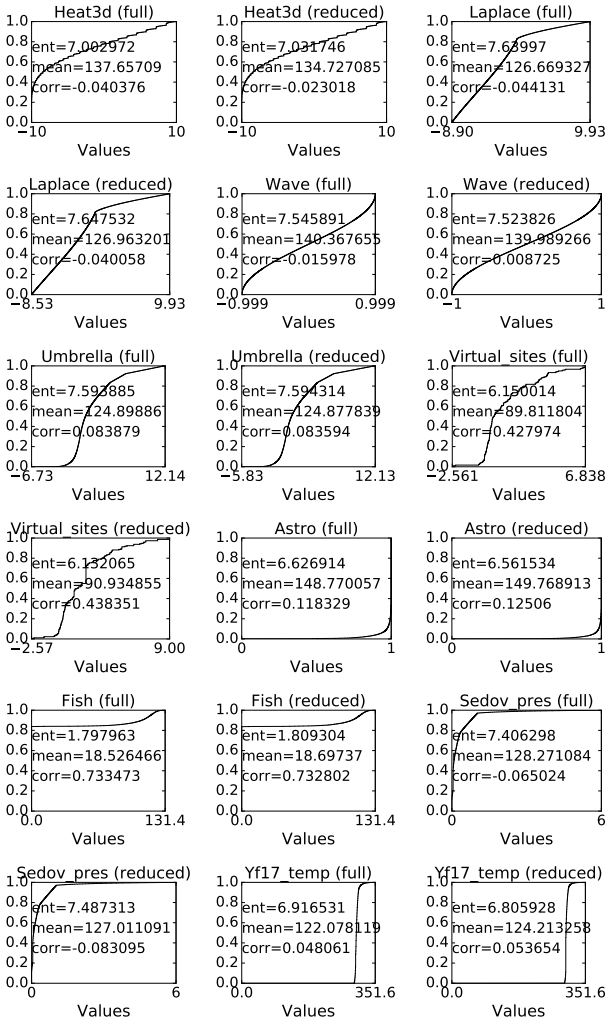



Fig. 1: Data characteristics of full model and reduced model. The curve shows the CDF of data values. $ent$, $mean$, and $corr$ denote the byte entropy, byte arithmetic mean, and serial correlation coefficient, respectively.

be close to 127.5. If it deviates from this value, the values are consistently high or low. $Serial\ correlation$ measures the extent to which each byte in the file depends upon the previous byte, and the metric ranges from -1 to 1. The closer the value is to 1 (or -1), the higher the data are positively (or negatively) correlated. For completely uncorrelated data, $serial\ correlation$ is close to 0.

It is evident that the full model and reduced model share nearly identical trends in their CDFs. All three scalar quantities also yield similar outcomes for the two models. Therefore, there exists high similarity between the full model and reduced model. The high correlations between them inspire us to use the reduced model output to precondition data and compress the difference.

### B. Weakness of DuoModel

*DuoModel* is based upon a simple idea that reducing the resolution of an application should result in similar results.

The main goal of *DuoModel* is to improve the compression ratios for a compressor $C$ on output $O$ produced by a simulation code $S$. The idea is to run a less expensive model $S'$, calculate the differences, i.e., $S(O) - S'(O)$, and then output $C(S(O) - S'(O))$, i.e., the compressed delta, instead of $C(S(O))$. Note that the minus sign here does not necessarily indicate a direct subtraction. For data analysis, the original data can be regenerated by re-running $S'(O)$ and applying the decompressed $S(O) - S'(O)$.

*DuoModel* builds upon two key observations: 1) Compared to storage, the cost of compute is getting cheaper. We can recompute data rather than store them with a huge I/O overhead. A trade-off between compute and storage can be exploited to reduce the I/O cost. 2) For modern lossy compressors, e.g., ZFP, SZ, the compression ratios are data dependent. They rely on the local smoothness within a dataset to compress data, and this can be captured by a reduced model. While achieving substantial compression ratio improvement, *DuoModel* has the following shortcomings: 1) The model reduction for simulations is in general complex, and requires substantial domain knowledge. 2) *DuoModel* involves extra communications between full model and reduced model. This may not be efficient at scale on large HPC systems. 3) For decompression, the overhead of running the reduced model is unacceptable for some situations, e.g., using 25% of compute resources that are originally allocated to the simulation [15].

## IV. Projection-based Reduced Model

In this section, we explore the mechanisms to build a reduced model using projection-based model reduction. We conduct a case study using *Heat3d*, and develop an ad-hoc solution to precondition *Heat3d* data.

### A. Case Study of Heat3d

The projection-based model reduction targets a broad class of modeling and simulation-based problems for which there are underlying governing equations determining the system behaviors [17]. A central idea is to remove non-essential terms, thus reducing the computational complexity. To understand the effectiveness of this methodology, we conduct a case study using a classical PDE application, *Heat3d* [19], which studies how heat propagates in a closed system. The *Heat3d* is parallelized using the message passing interface (MPI) for inter-processor communication. Mathematically, the 3D heat equation can be described as follows.

$$\frac{\partial u}{\partial t} = \kappa \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

where $\kappa$ is the thermal conductivity coefficient, and $u$ is the temperature at coordinate $(x, y, z)$ at time $t$. After discretization, $u$ at coordinate $(i, j, k)$ of step $n + 1$ can be calculated using the central difference, as shown below.

$$\begin{aligned} u_{n+1}[i][j][k] = u_n[i][j][k] + \\ \kappa \Delta t ( \frac{u_n[i+1][j][k] - 2u_n[i][j][k] + u_n[i-1][j][k]}{h_x^2} \\ + \frac{u_n[i][j+1][k] - 2u_n[i][j][k] + u_n[i][j-1][k]}{h_y^2} \\ + \frac{u_n[i][j][k+1] - 2u_n[i][j][k] + u_n[i][j][k-1]}{h_z^2} ) \end{aligned} \quad (1)$$

Subsequently, we project the 3D solution space into 2D by collapsing the Z dimension. This essentially disregards the heat conduction in Z direction. Therefore, equation (1) is reduced to the following.

$$\begin{aligned} u_{n+1}[i][j] = & u_n[i][j] + \kappa \Delta t ( \frac{u_n[i+1][j] - 2u_n[i][j] + u_n[i-1][j]}{h_x^2} \\ & + \frac{u_n[i][j+1] - 2u_n[i][j] + u_n[i][j-1]}{h_y^2} ) \end{aligned}$$

TABLE II: Heat3d full model and reduced model.

| | **Full model** | **Reduced model** |
|---|---|---|
| **Problem size** | $192 \times 192 \times 192$ | $192 \times 192$ |
| **Processors** | $8 \times 8 \times 8$ | $8 \times 8$ |
| **# of steps** | 50,000 | 260 |
| **Time step** | $1.712 \times 10^{-8}$ | $3.391 \times 10^{-6}$ |
| **Byte entropy** | 7.002972 | 7.031746 |
| **Byte mean** | 137.657090 | 134.727085 |
| **Serial correlation** | -0.040376 | -0.023018 |

Table II shows the setup and some key quantities of both models. In particular, the number of processors is reduced from 512 to 64. Besides, to guarantee that equation (1) can result in a mathematically stable solution, the length of a timestep should be less than the stability condition [18], $\Delta t = \frac{1}{8\kappa} (\min(h_x, h_y, h_z))^3$. Since the Z dimension is collapsed, we set a larger time step for the reduced model, i.e., increasing from $1.712 \times 10^{-8}$ to $3.391 \times 10^{-6}$. We also calculate the three scalar quantities of data characteristics (detailed in Section III). It is shown that they are nearly the same. This indicates that the reduced model generated by projection-based reduction method can be highly similar to the full model.

We further calculate the difference between the full model and reduced model output. To that end, we subtract each horizontal plane in the original 3D space by the reduced 2D plane to calculate the residuals. We notice that for the *Heat3d* solution space, its middle plane is almost the same as 2D solution of the reduced model, and it is the symmetry plane in the solution space. Therefore we intuitively choose the middle plane as the reduced model. Overall, this observation inspires us that a reduced model can reside in the full model output itself, without constructing another application instance which can be cumbersome.

Fig. 2 illustrates various reduced models for *Heat3d*. We develop two ad-hoc projection-based schemes, termed as, *one-base* and *multi-base*, to understand how well the proposed idea can perform. For *one-base*, as detailed in Algorithm 1, we choose a horizontal mid-plane from the full model and use

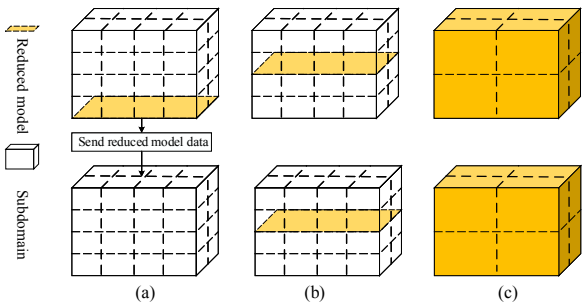



Fig. 2: Reduced models of *Heat3d*. a) *One-base*: the middle horizontal plane within the solution space is used as the reduced model; b) *Multi-base*: the middle horizontal plane in each sub-domain is used as the reduced model; c) *DuoModel*: a lower resolution of the original model is used.

it as reference to calculate the delta. Implementation wise, the processors that contain the mid-plane will send the plane to other processors to calculate the delta. For *multi-base*, to avoid the communication overhead of sending the mid-plane, we select the mid-plane on a per sub-domain basis. Accordingly, the delta calculation within a sub-domain is done by only subtracting its local mid-plane. For comparison, we also illustrate *DuoModel* that uses a lower resolution as a reduced model. The delta is calculated as the difference between the full model data and the linear constructed data (detailed in prior work [15]).

**Algorithm 1** Calculating the delta (using *one-base* as an example).

**Require:** $m_z$ planes of full model data matrix $u$, the $z_{high}$ and $z_{low}$ are the upper and lower indices in the Z dimension of the subdomain.

**Ensure:** The delta between the full model and reduced models.

1: **if** $u(m_z/2)$ is within the MPI rank **then**
2: Broadcast the plane to all other ranks.
3: **else**
4: Receive the plane $u(m_z/2)$.
5: **end if**
6: **for all** $i \in [z_{low}, z_{high}])$ **do**
7: $\Delta(i) = u(i) - u(m_z/2)$.
8: **end for**
9: Gather the delta.

### B. Performance Evaluation

We evaluate the projection-based reduced models on Titan at Oak Ridge National Laboratory. We use 32 compute nodes (512 MPI processors) to run *Heat3d* and *Laplace*, respectively. Since the *Wave* application is only one dimensional, it is not relevant here. We run both lossless and lossy compressors to understand more broadly how much projection-based reduced models can improve compression ratios. In particular, we use SZ (1.4.11) with the default mode and point-wise relative error bound of $10^{-5}$. For ZFP (0.5.0), since it does not support point-wise relative error, we choose the fixed-precision mode with 16 bits of precision. For FPC (1.1), we use the level of 20 with the internal table size of $2^{24}$ bytes.

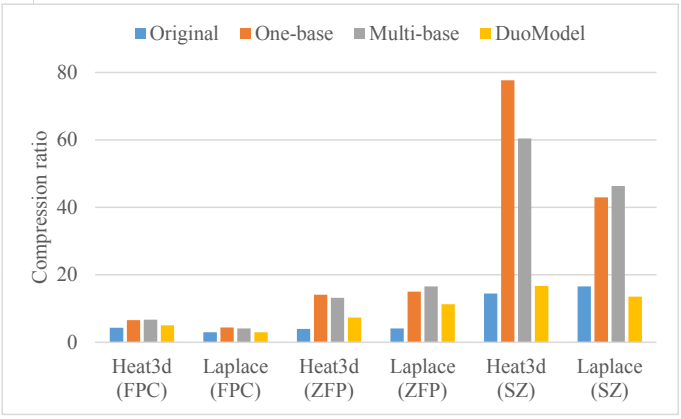


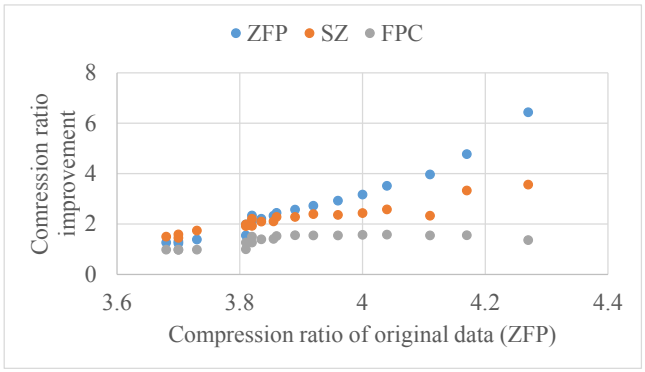

Fig. 3: Compression ratios using projection-based methods. The blue bars represent the baseline where datasets are directly compressed, and the yellow bars represent our previous work *DuoModel*. The x-axis is the test setup (i.e., dataset name and compressor used. In this paper, original means to use the compressors directly). Each data point shows the average compression ratio of 20 outputs of each application.

Fig. 3 shows the compression ratios achieved by the three reduced models. It is evident that the projection-based reduced models result in significant improvement in compression ratios for both *Heat3d* and *Laplace*, particularly for lossy compression. On average, the compression ratios of ZFP increase from 4X to greater than 15X for all three methods. Using SZ to compress the preconditioned data, the compression ratios increase from 17X to greater than 40X for both *one-base* and *multi-base*. Intuitively speaking, *multi-base* should outperform *one-base* due to its better capability of capturing data characteristics. However, *multi-base* requires more storage space to save the reduced models, thus offsetting its compression ratios. In addition, *one-base* and *multi-base* outperform *DuoModel*. The reason is that the delta generated by the *one-base* and *multi-base* are with more smoothness than those generated by *DuoModel*. Even for those planes that are physically far away from the middle plane, the absolute values of delta are large, but the variations are smaller than those of *DuoModel*, thus being more compressible. To further show this, we evaluate the techniques with FPC [6]. It is also found that *one-base* and *multi-base* can significantly improve the compression ratio of FPC, but *DuoModel* cannot. This suggests that the delta generated by *one-base* and *multi-base* are more compressible.

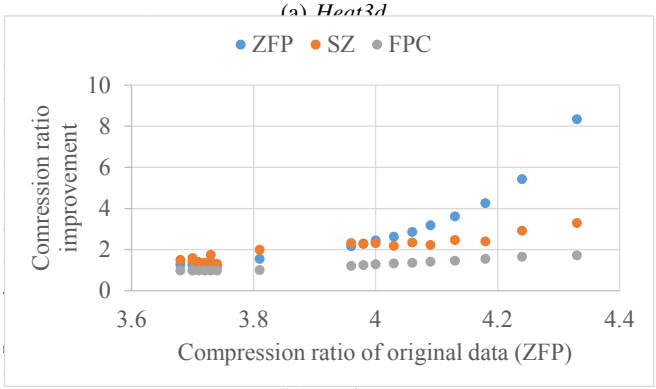
Fig. 4 further reveals that the effectiveness of projection-based reduced models depends on the compressibility of datasets. The more compressible the data are, the more compression ratio improvement we can achieve by using reduced models.

## V. Exploring General Reduced Models

For datasets that model sophisticated problems with complex geometry, identifying the reduced model may not be




(a) *Heat3d*




(b) *Laplace*.

Fig. 4: Compression ratio improvement vs. the compressibility of original data (indirectly captured by the compression ratios achieved by ZFP). The reduced model used is *one-base*. For *Heat3d* and *Laplace*, both include 20 outputs of the simulations, uniformly chosen from the beginnings to the ends of their lifetimes.

as straightforward as *one-base* and *multi-base* in *Heat3d*. Fortunately, scientific data are naturally multi-dimensional, capturing physical quantities in both space and time [20]. A straightforward idea is to further explore common dimension reduction techniques to extract reduced model. Unlike *one-base* or *multi-base* reduced models, whose data outputs are a subset of the full model data, the dimension reduction can be viewed as a transformation from the full model data, e.g., the reduced model data of PCA are linear combinations of the original data in columns.

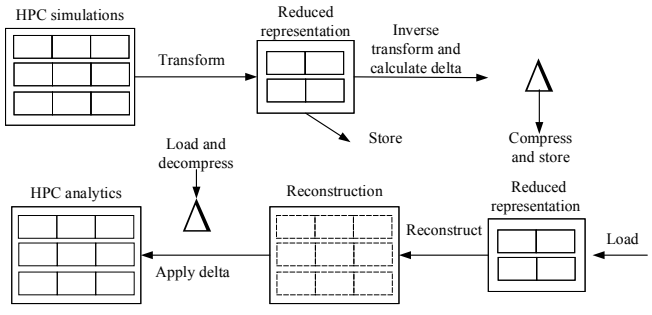



Fig. 5: Data reduction and reconstruction using dimension reduction techniques.

Fig. 5 illustrates the workflow of using dimension reduction techniques to find the reduced model, including PCA, SVD, and Wavelet. In particular, the upper part shows the reduction phase, and the lower part shows the reconstruction (i.e., decompression) phase. For the reduction phase, we transform analysis data to their reduced representations, which are then used to precondition the compression. In particular, we run the inverse transformation on the reduced representation and then calculate the delta between the original data and the reconstructed data. In the end, the reduced representation along with the delta is compressed and stored. The key idea, similar to projection-based methods, is that after dimension reduction, the reduced representation is much smaller but can still capture the characteristics of the original data. This also results in highly compressible delta. For the reconstruction phase, an inverse transformation is executed upon the reduced representation, and the original data is regenerated by further applying the decompressed delta.

### A. Adopting Dimension Reduction Techniques

*1) PCA:* PCA uses an orthogonal transformation to convert correlated variables into a set of uncorrelated variables, called principal components [21]. In particular, the first primary component captures the original variance as much as possible, and the second primary component captures the remaining variance, and so on. To use PCA to find the reduced model, the eigenvectors and eigenvalues of the covariance matrix of the original data are first calculated. Then, $k$ eigenvectors with the largest eigenvalues are selected and multiplied by the original data to reorient the dimension-reduced data. The dimension-reduced data and the eigenvectors are retained to be the reduced representation.

*2) SVD:* In contrast to PCA that only focuses on the column space, SVD focuses on both the column and row space [22]. The singular-value decomposition of a matrix $A$ can be represented as $U \cdot \Sigma \cdot V$, and the diagonal entries $\sigma_i$ of $\Sigma$ are known as the singular values of $A$. Similar to PCA, a larger singular value tends to capture more important information than a smaller singular value. To obtain the reduced representation, we only retain $k$ largest singular values, and the corresponding $k$ rows of $U$ and $k$ columns of $V$.

*3) Wavelet:* Discrete wavelet transform [23] decomposes a signal into mutually orthogonal sets of wavelet basis functions. In this work, we apply Haar Wavelet [24] to find the reduced model detailed in the following steps.

**Step 1:** For each row, group the entries into pairs, store their differences, and pass their sums to prove a new row with smaller scale. This process is repeated recursively, which ends when only one entry is a sum and all the other entries are differences.

**Step 2:** Repeat the same process of Step 1 to each column.

**Step 3:** The resultant matrix contains many near-zero entries. We subsequently pick a threshold $\theta > 0$ and set those entries with absolute value smaller than $\theta$ to zeros. The new matrix in principle is sparser and can be stored more efficiently. The new spare matrix is regarded as the reduced representation.

TABLE III: Comparison of PCA, SVD, and Wavelet for a multi-dimensional dataset of $m \times n$. For the storage cost, all methods need to additionally store deltas, which are not listed here.

| | Method | Complexity | Storage |
|---|---|---|---|
| **PCA** | Column correlation | $O(mn^2 + n^3)$ | Dimension-reduced data, associated eigenvectors |
| **SVD** | Column/row correlation | $O(m^2n + mn^2 + n^3)$ | Three refactored matrices |
| **Wavelet** | Haar wavelet | $O(4mn^2 \log n)$ | Sparse matrix |

*4) Comparisons of Dimension Reduction Techniques:* Table III shows the comparisons of dimension reduction methods, in terms of methodology, complexity, and storage. The key idea of PCA is to find correlation between columns, while SVD considers both columns and rows. Wavelet is to find an inverse transformation that results in a sparse matrix that is typically more efficient to store. For a dataset that is a $m \times n$ matrix, the time complexity of PCA and SVD is $O(mn^2+n^3)$ and $O(m^2n + mn^2 + n^3)$, respectively. Haar Wavelet, similar to 2D Fast Fourier Transform [25], has the complexity of $O(4mn^2 logn)$. In regards to storage cost, PCA needs to store eigenvectors associated with $k$ primary components, while SVD stores three refactored matrices. For Wavelet, the sparse matrix is stored. All three methods need to additionally store deltas, which are needed to reconstruct the original dataset.

### B. Performance Evaluation

Herein we evaluate the compression ratio, information loss, and compression/decompression overhead. The experiments were conducted on a Linux server with Intel $Core^{TM}$ i5-7500 that has 4 cores with a frequency of 3.4GHz and 16 GB of memory. The operating system is Ubuntu 16.04.5 LTS. Note that the size of compressed data is contributed by both the reduced representation and the compressed delta. With regard to the information loss, we use the root mean square error (RMSE) to assess the compression quality of dimension reduction techniques. The configurations of ZFP and SZ are the same as those in Section IV-B. Note that different relative error bounds are applied to the original data and delta. The reason is that the effect of relative error bounds is data dependent. Since the delta is often much smaller in its magnitude than the original data, as a result of the similarity between full model and reduced model, we need to apply a looser relative error bound to achieve the prescribed accuracy. For example, assuming the value of a data point is denoted as $V$, the corresponding delta is $10^{-2} \cdot V$. A relative error bound of $10^{-5}$ for the original data means that the decompressed data error is in the range of $[-10^{-5} \cdot V, +10^{-5} \cdot V]$. For the delta, maintaining a relative error bound of $10^{-5}$ would result in an error in the range of $[-10^{-7} \cdot V, +10^{-7} \cdot V]$, a tighter error bound that is unnecessary. Therefore, in this work, for ZFP, the error bounds for original data and delta are set to 16 and 8 bits of precision, respectively. For the SZ, the relative error bounds for original and delta are set to $10^{-5}$ and $10^{-3}$, respectively. We select the number of components, $k$, such that $\sum_{i=1}^{k} \sigma_i / \sum_{i=1}^{n} \sigma_i \geq 95\%$ [26], in which $\sigma_i$ is the variance of the $i$-th largest primary components (or singular values) for PCA (or SVD). Finally, for Wavelet, we set the threshold $\theta$ to 5% of the maximum value in the transformed matrix.

*1) Compression Ratio:* Fig. 6 shows the compression ratios of using PCA, SVD, and Wavelet to precondition data. The resultant data are further compressed using either ZFP or SZ. It is shown that the PCA and SVD can significantly improve the compression ratios of *Heat3d*, *Laplace*, *Wave*, *Astro*, and *Sedov_pres*. However, the improvement for other datasets is insignificant. In particular, for *Fish* data, PCA, SVD, and Wavelet all result in lower compression ratios than compressing it directly. The reason is that *Fish* is a peculiar dataset that contains many zeros. Using these three methods, the delta will, however, contain near-zero values, which are less compressible than the original *Fish*. To further understand the different outcomes across datasets, Figs. 7 and 8 show the proportion of variance of the primary components in PCA, and that of singular values of SVD, respectively. It is found that, the more dominant the first primary component (or singular value) is, the higher compression ratio improvement we can achieve.

Among the three reduced representations, PCA and SVD are overall better than Wavelet for most datasets. The improvement from Wavelet is insignificant as compared to compressing data directly. The reason is that the sparse matrix produced by Wavelet can still result in high storage overhead. Fig. 9 shows the size of reduced representations produced by PCA, SVD, and Wavelet. The size of Wavelet is much higher than the other two methods. One could set a larger threshold $\theta$ to further reduce the reduced representation size of Wavelet. However, the associated delta will become less compressible, which offsets the overall improvement.

*2) RMSE Evaluation:* Fig. 10 shows the RMSE results of PCA, SVD, and Wavelet, respectively. Overall Wavelet yields a much higher RMSE than compressing data directly for most datasets. Given the insignificant compression ratios and high RMSE, Wavelet is not deemed to be a good strategy to precondition compression. For PCA and SVD, despite the significant compression ratio improvement, they also yield a higher RMSE than compressing data directly. The reason is that there is information loss for the reduced representation with ZFP and SZ, and the information loss may be amplified in the inverse transformation during reconstruction phase.

This further raises a new question - can PCA and SVD improve the compression ratio while maintaining the same information loss? We evaluate different error bounds on all datasets using ZFP. In particular, the number of precision bits for the compression is varied from 8 to 32. Fig. 11 shows the results of compression ratios under different RMSEs. It is found that under the same information loss, PCA and SVD can achieve higher compression ratios than compressing data directly using ZFP for some of the datasets. Therefore, the finding here is that using dimension reductions is possible to outperform conventional compressors but this depends on the characteristics of data.

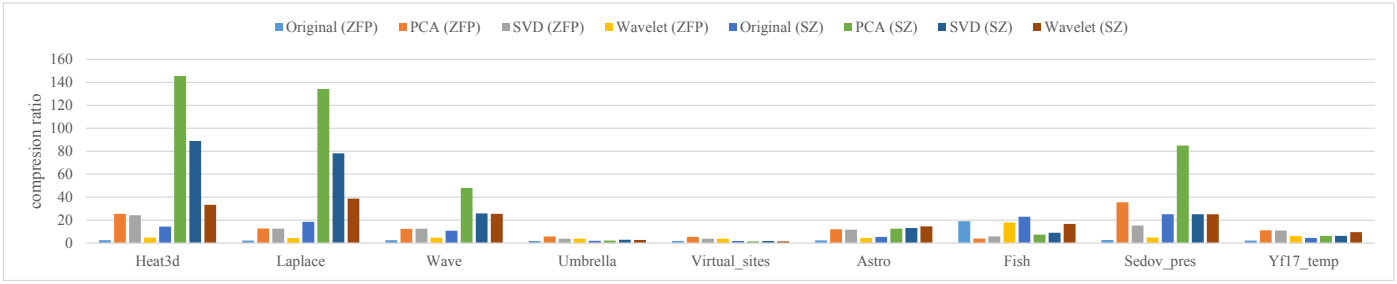



Fig. 6: Comparison of compression ratios. The legends show the conjunctions of the three dimension reduction techniques with ZFP and SZ, respectively.

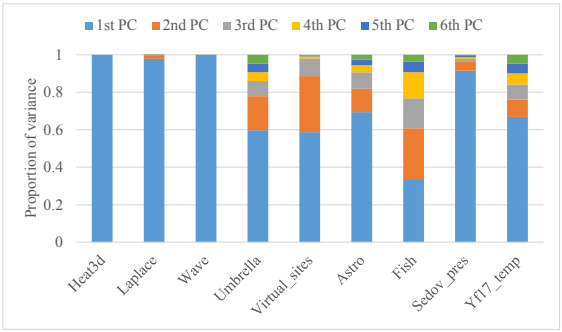



Fig. 7: PCA proportion of variance of the primary components. Note that PC is the abbreviation for primary component.

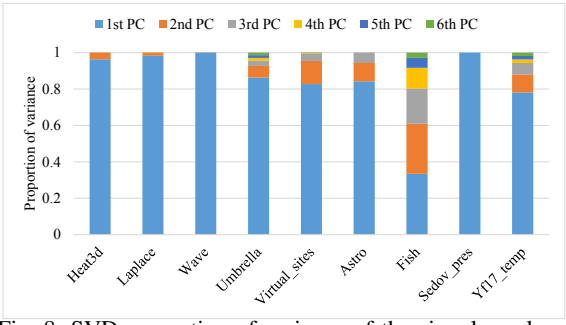



Fig. 8: SVD proportion of variance of the singular values.

*3) Overhead Analysis:* Fig. 12 shows the average compression and decompression time. Compared to compressing data directly using ZFP, PCA, SVD, and Wavelet increase the compression time by 6.5X, 16.6X, and 3.1X, respectively. The overhead is governed by the computational complexity of dimension reductions. Compared to decompressing data directly using ZFP, PCA, SVD, and Wavelet increase the decompression time by 4.9X, 6.9X, and 1.2X, respectively. It is clear that compression is more expensive than decompression, which can be attributed to the expensive matrix decomposition during compression.

*4) End-to-end Time:* The significant compression overhead naturally raises a question: will the reduction in I/O time pay off the added compression overhead for the reduced model based methods? We take *Heat3d* as an example to evaluate PCA to understand the end-to-end time including the compression and I/O times. The experiments are conducted

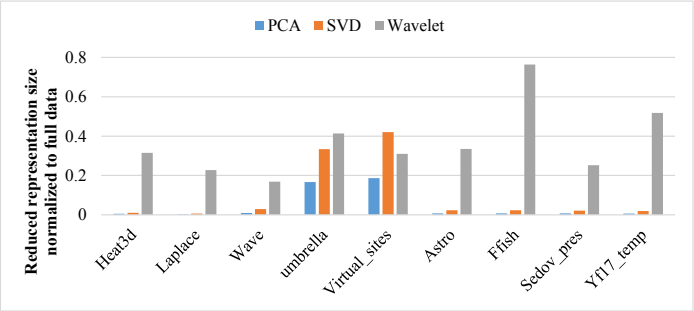



Fig. 9: The size of reduced representations.

TABLE IV: Compression and I/O time.

| Method | Compression time(s) | I/O time(s) | Total time(s) |
|---|---|---|---|
| **Baseline (I/O with no compression)** | N/A | 52.48 | 52.48 |
| **ZFP+I/O** | 12.09 | 20.39 | 32.49 |
| **SZ+I/O** | 9.72 | 19.36 | 29.09 |
| **PCA(ZFP)+I/O** | 44.87 | 9.23 | 54.11 |
| **PCA(SZ)+I/O** | 42.95 | 9.00 | 51.96 |
| **Staging+PCA+I/O** | N/A | 13.17 | 13.17 |

on the supercomputer Titan[4] and Lustre parallel file system. The number of processors is set to 64, and each processor generates 16.7 GB data. We evaluate six schemes, as shown in Table IV, the baseline (i.e., no compression), four compression methods, including direct compression using ZFP (i.e., ZFP+I/O) and SZ (i.e., SZ+I/O), PCA in conjunction with ZFP (i.e., PCA(ZFP)+I/O) or SZ (i.e., PCA(SZ)+I/O), and PCA in conjunction with data staging (Staging+PCA+I/O) [27], [28]. In particular, for the baseline, each processor of Heat3d writes its subdomain to persistent storage without compression. In contrast, for the four compression methods, each processor compresses its local subdomain prior to storing. Regarding the implementation in a parallel file system, each processor compresses and writes independently in an N-to-N fashion. The results in Table IV show that the compression time of ZFP (or SZ) is 12.09s (or 9.72s), and the I/O time is 20.39s (or 19.36s). Thus, the benefit of data reduction outweighs the increased compression time. However, for the proposed reduced model based methods, the compression overhead is significantly higher. The results show that the total time in this case is similar to that of the baseline. To address this issue, we further take advantage of data staging, which is a new paradigm

[4]https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/

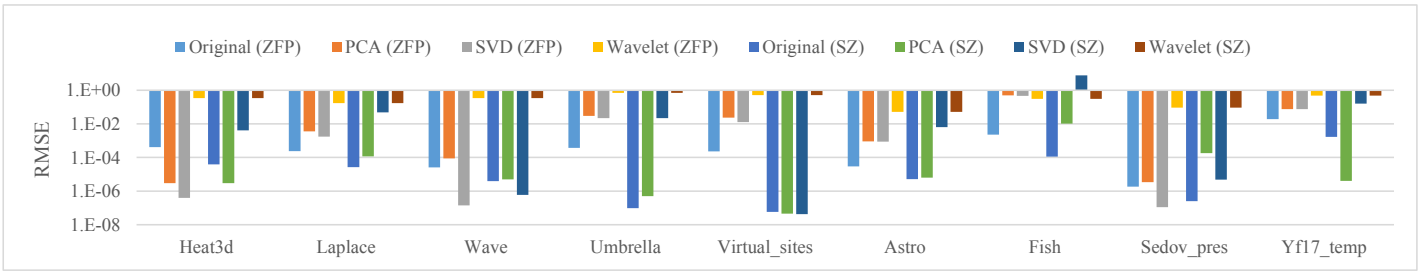



Fig. 10: Comparison of RMSE introduced by different kinds of methods, including the eight conjunctions of the three reduced models with ZFP and SZ.

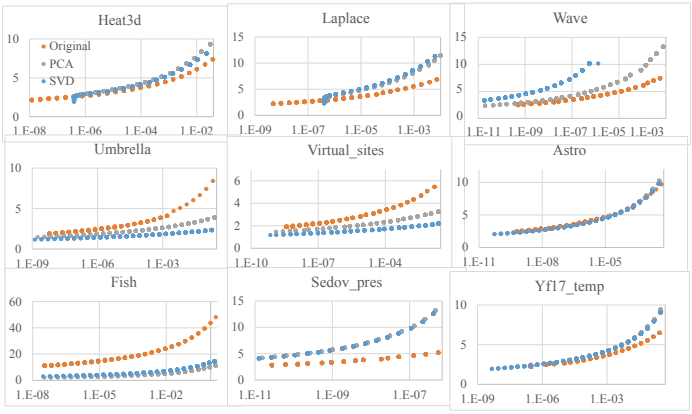
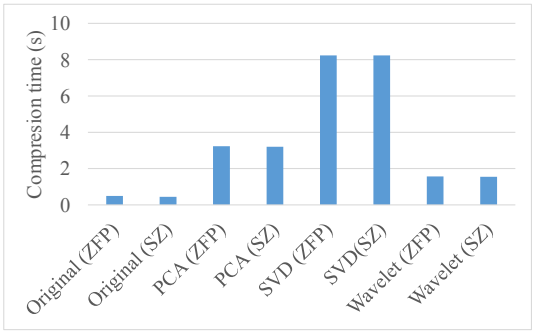
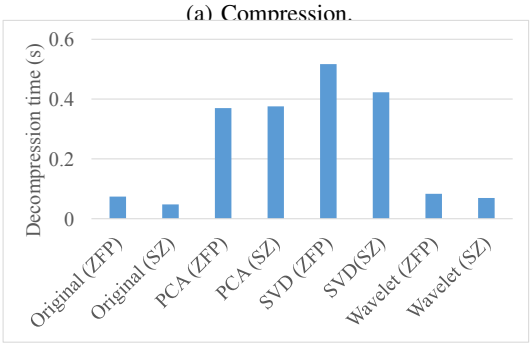



Fig. 11: Comparison of compression ratios under the same RMSE. Due to the space limit, we only take ZFP as an example to show the results. The y-axis is labeled as compression ratio, and the x-axis is RMSE.




(a) Compression.




(b) Decompression.

Fig. 12: Compression and decompression overhead. The y-axis is the average compression or decompression time across all datasets.

that allows data processing to be done asynchronously, thus greatly alleviating the demands for high I/O rates. We note that data staging, such as burst buffer, is prevalent on emerging HPC architectures, e.g., Cori and Summit. In particular, in our experiments, one additional compute node is allocated as a staging node, and the compressions and I/O operations can be done in this node asynchronously without impacting the HPC simulation. The last row of Table IV shows the total time with data staging, and it is found that total time is reduced to 13.17s, which is mainly the time spent on sending data from the application to the staging node.

## VI. Related Work

As far as we are aware, our work is the first one that uses reduced models to improve the compression ratio of existing compressors. The most related work to this paper are summarised into two categories. The first one is the Tucker decomposition. Austin *et al*. [20] showed that scientific datasets can have excellent compression rates by taking advantage of data dimensions. They proposed a parallel implementation for computing the Tucker decomposition of general dense tensors. Choi *et al*. [29] proposed an implementation and performance analysis of GPU-accelerated Tucker decomposition for dense tensors. A slice block partitioning method is used to improve performance for GPUs, and a tensor matricization layout is designed to reduce the number of all-reduce communications and matricizations. The second category is to optimize existing ZFP and SZ. Gok *et al*. proposed PaSTRI [10] for compressing integral data used in quantum chemistry. They first mined of the latent pattern features of the integral data with an in-depth analysis. Then, they utilized the patterns for the optimization of SZ with different error bounds. Tao *et al*. [5] noted that neither SZ or ZFP is the best compressor across different datasets and across different fields of a dataset. Thus, they proposed an online selection method that guides users to select the best-fit lossy compressor between SZ and ZFP. Liang *et al*. [30] focused on the trade-off between compression ratio and reconstructed data loss. The work aims to control the data distortion when reducing the data size. The main idea is to adaptively select the best-fit prediction approach with the consideration of data features in different regions of a dataset.

## VII. Conclusion and Future Work

The paper focuses on further improving the compression ratios by preconditioning data. The central idea is to identify the reduced model from the full model applications, and use it for preconditioning prior to compression. We first

illustrate that there are high similarities between the full model and reduced model, and compressing the delta between them often results in high compression ratios. As a proof of concept, we develop a projection-based model reduction in *Heat3d* to find the reduced model within the full model output itself. As such, the disadvantages of *DuoModel*, such as the resource and communication overhead, can be avoided. More general dimension reduction techniques, including PCA, SVD, and Wavelet, are then explored for the purpose of data compression. We evaluate various reduced methods on nine scientific datasets, and the results show the effectiveness of our approach. The future work consists of two directions. The first is to implement the proposed reduced methods in partitioned matrixes to further reduce the compression overhead. For the second one, we notice that there is no single reduced method that is the best of all datasets. Therefore, it is motivated to propose a model selection strategy that selects the best model prior to data reduction.

## ACKNOWLEDGMENTS

The authors wish to acknowledge the support from the US NSF under Grant No. CCF-1704504, CCF-1812861, and CCF-1629625, NetApp research grant, and NJIT research startup fund. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

## REFERENCES

[1] M. Burtscher, H. Mukka, A. Yang, and F. Hesaaraki, "Real-time synthesis of compression algorithms for scientific data," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016, p. 23.

[2] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Communications of the ACM*, vol. 58, no. 7, pp. 56–68, 2015.

[3] S. Ku, C.-S. Chang, M. Adams, J. Cummings, F. Hinton, D. Keyes, S. Klasky, W. Lee, Z. Lin, S. Parker *et al.*, "Gyrokinetic particle simulation of neoclassical transport in the pedestal/scrape-off region of a tokamak plasma," in *Journal of Physics: Conference Series*, vol. 46, no. 1, 2006, p. 87.

[4] I. Foster, M. Ainsworth, B. Allen, J. Bessac, F. Cappello, J. Y. Choi, E. Constantinescu, P. E. Davis, S. Di, W. Di *et al.*, "Computing just what you need: online data analysis and reduction at extreme scales," in *European Conference on Parallel Processing (EURO-PAR)*. Springer, 2017, pp. 3–19.

[5] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression rate-distortion from automatic online selection between SZ and ZFP," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[6] M. Burtscher and P. Ratanaworabhan, "Fpc: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2009.

[7] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data," in *European Conference on Parallel Processing (EURO-PAR)*. Springer, 2011, pp. 366–379.

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

[9] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 730–739.

[10] A. M. Gok, S. Di, Y. Alexeev, D. Tao, V. Mironov, X. Liang, and F. Cappello, "Pastri: Error-bounded lossy compression for two-electron integrals in quantum chemistry," in *IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 1–11.

[11] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681–1710, 2016.

[12] Q. Zheng, K. Ren, G. Gibson, B. W. Settlemyer, and G. Grider, "DeltaFS: Exascale file systems scale better without dedicated servers," in *Proceedings of the 10th Parallel Data Storage Workshop (PDSW)*, 2015, pp. 1–6.

[13] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci *et al.*, "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012, p. 49.

[14] P. Ghysels and W. Vanroose, "Hiding global synchronization latency in the preconditioned conjugate gradient algorithm," *Parallel Computing*, vol. 40, no. 7, pp. 224–238, 2014.

[15] H. Luo, Q. Liu, Z. Qiao, J. Wang, M. Wang, and H. Jiang, "Duomodel: Leveraging reduced model for data reduction and re-computation on HPC storage," *IEEE Letters of the Computer Society*, 2018.

[16] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, and Z. Qiao, "Understanding and modeling lossy compression schemes on HPC scientific data," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018, pp. 1–10.

[17] P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM review*, vol. 57, no. 4, pp. 483–531, 2015.

[18] M. Rietmann, D. Peter, O. Schenk, B. Uçar, and M. Grote, "Load-balanced local time stepping for large-scale wave propagation," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2015, pp. 925–935.

[19] Dournac, "MPI Parallelization for numerically solving the 3D Heat equation," https://dournac.org/info/parallel_heat3d, Year = 2018.

[20] W. Austin, G. Ballard, and T. G. Kolda, "Parallel tensor compression for large-scale scientific data," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 912–922.

[21] P. Forczmański and W. Maleika, "Near-lossless PCA-based compression of seabed surface with prediction," in *International Conference Image Analysis and Recognition (ICIAR)*. Springer, 2015, pp. 119–128.

[22] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "TTHRESH: Tensor compression for multidimensional visual data," *arXiv preprint arXiv:1806.05952*, 2018.

[23] D. Gupta and S. Choubey, "Discrete wavelet transform for image processing," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 3, pp. 598–602, 2015.

[24] C. Mulcahy, "Image compression using the haar wavelet transform," *Spelman Science and Mathematics Journal*, vol. 1, no. 1, pp. 22–31, 1997.

[25] J. S. Walker, *Fast fourier transforms*. CRC press, 2017.

[26] P. R. Peres-Neto, D. A. Jackson, and K. M. Somers, "How many principal components? stopping rules for determining the number of non-trivial axes revisited," *Computational Statistics & Data Analysis*, vol. 49, no. 4, pp. 974–997, 2005.

[27] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "Datastager: scalable data staging services for petascale applications," *Cluster Computing*, vol. 13, no. 3, pp. 277–290, 2010.

[28] T. Jin, F. Zhang, Q. Sun, H. Bui, M. Romanus, N. Podhorszki, S. Klasky, H. Kolla, J. Chen, R. Hager *et al.*, "Exploring data staging across deep memory hierarchies for coupled data intensive simulation workflows," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2015, pp. 1033–1042.

[29] J. Choi, X. Liu, and V. Chakaravarthy, "High-performance dense tucker decomposition on gpu clusters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*, 2018, p. 42.

[30] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *IEEE International Conference on Big Data (BigData)*, 2018, pp. 1–10.