A Data-driven Approach to Harvesting Latent Reduced Models to Precondition Lossy Compression for Scientific Data

Huizhang Luo, Junqi Wang, Zhenlu Qin, Dan Huang, Qing Liu, Mengchu Zhou, *Fellow, IEEE*, and Hong Jiang, *Fellow, IEEE*

Abstract—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 lossy compressors for HPC scientific data. In particular, we aim to identify a reduced model that can be utilized to transform the original data into a more compressible form. We begin with two PDE applications 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. A mathematical proof is also presented to show how the compression ratio is improved by the reduced model. 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 ten scientific datasets, and the results show the effectiveness of our approaches. Given that there is no single method that consistently achieves the best performance, we further propose a selection strategy that guides users to select the best reduced model prior to data reduction.

Index Terms—Data reduction, data preconditioning, compressor selection, high-performance computing

1 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], [7], [8], [9], [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

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

- *Huizhang Luo and Junqi Wang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China. E-mail: luohuizhang@hnu.edu.cn, junqi.wang.87@gmail.com.*
- *Zhenlu Qin, Dan Huang, Qing Liu, and Mengchu Zhou are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA. E-mail: {zq53, qing.liu, zhou}@njit.edu, huangdan226@gmail.com.*
- *Hong Jiang is with the Computer Science and Engineering Department, University of Texas, Arlington, TX 76019 USA. E-mail: hong.jiang@uta.edu.*

Manuscript received 22 June 2022; revised 7 November 2022; accepted 28 November 2022. Date of publication 1 December 2022; date of current version 13 May 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2021YFB0301300, and in part by the National Natural Science Foundation of China under Grant 62102141. 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 Grant DE-AC05-00OR22725.
(Corresponding author: Junqi Wang.)
Recommended for acceptance by D. Puthal.
Digital Object Identifier no. 10.1109/TBDATA.2022.3225959

2332-7790 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

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 weakness 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 delta, which results in higher compression ratios. The original data can be re-computed by reconstructing the full model data and applying the delta. A key finding of this paper is that there is no single method that consistently yields the best compression ratios. Therefore, we propose a model selection strategy that selects an appropriate model prior to data reduction. This potentially can guide application scientists to choose the right reduced model without exhausting all reduced models. The major contributions of this paper are as follows:

- We synthesize the reduced model directly from the analysis output. Our case study of PDE applications suggests that this approach results in higher compression ratios than compressing data directly or using *DuoModel*. We further present a mathematical proof that the proposed methods can improve the compression ratio;
- 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. We also analyze how the compression ratio and information loss are impacted by the proposed methods;
- We propose a sampling-based model selection strategy with low overhead. This can guide application scientists to choose an appropriate method prior to compression.

The remainder of this paper is organized as follows. Section 2 provides the background. Section 3 provides the motivation. Sections 4 and 5 discuss how to identify the reduced model from analysis output using project-based methods and dimension reductions. Section 6 presents the selection method of the right reduced model. Section 7 presents the most related works. Conclusions are presented in Section 8. Our experiments for evaluation are publicly available at https://github.com/luohuizhang/ReducedModel.

2 Background

2.1 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 a 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.

Compression ratio, throughput and information loss are three of the most importance metrics that measure if a compressor is good or not [16]. In particular, compression ratio is the ratio of the original dataset size to the compressed data size. Throughput is divided into compression and decompression throughput, which defines how fast the data can be compressed and decompressed. At last, information loss defines how much information are lost when the data are decompressed by a lossy compressor. RMSE is a common metric to quantify information loss.

2.2 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

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

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

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 [18], [19]: 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], [20], [21]. 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. This reveals that a reduced model can indeed retain a similar system state, motivating us to use it to precondition data prior to compression.

3 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 ten fields of datasets 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.

3.1 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 floating-point datasets that are generated from real HPC applications. The details of the datasets are shown in Table 1[3]. 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. We first show the similarity between full model and reduced model with the three classical partial differential equation (PDE) applications, i.e., *Heat3d*, *Laplace*, and *Wave*. The reduced model applications are co-running with the full model applications at the same time, and the reduced models are obtained by scaling down the problem size. For example, for *Heat3d*, we set the number of points on each dimension of full (reduced) model as $192 \times 192 \times 192$ ($48 \times 48 \times 48$).

TABLE 1
Dataset Description

Dataset	Description
Heat3d	Distribution of heat in a given region over time, dimensions: $192 \times 192 \times 192$, type: double, 20 timesteps, 1.06 GB.
Laplace	Description of steady state situations of values distributions, dimensions: 960×960, type: double, 20 timesteps, 140MB.
Wave	Hyperbolic PDE for the description of waves, dimensions: 40961, type: double, 1024 timesteps, 320 MB.
Sedov_pres	Pressure of strong shocks in a hydrodynamical simulation, dimensions: 78144, type: double, 1024 timesteps, 610 MB.
Astro	Velocity magnitude in a supernova simulation, dimensions: 256×256, type: double, 1024 timesteps, 512 MB.
Hurricane ISABEL	Weather simulation, dimensions: $100 \times 500 \times 500$, type: single, 13 timesteps, 1.25 GB.
CESM-ATM	Climate simulation, dimensions: 1800×3600, type: single, 79 timesteps, 1.47 GB.
NYX	Cosmology: Adaptive mesh hydrodynamics + N-body cosmological simulation, dimensions: $512 \times 512 \times 512$, type: single, 6 timesteps, 2.7 GB.
XGC	Fusion simulation, dimensions: 20694×512, type: double, 9 timesteps, 1.2 GB.
S3D	Combustion simulation, dimensions: $11 \times 500 \times 500 \times 500$, type: double, 4 timesteps, 44 GB.

Fig. 1 demonstrates the similarity between the full model and reduced model for the ten 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 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.

3. The datasets are available at https://sdrbench.github.io/, and https://dournac.org/info/parallel_heat3d.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

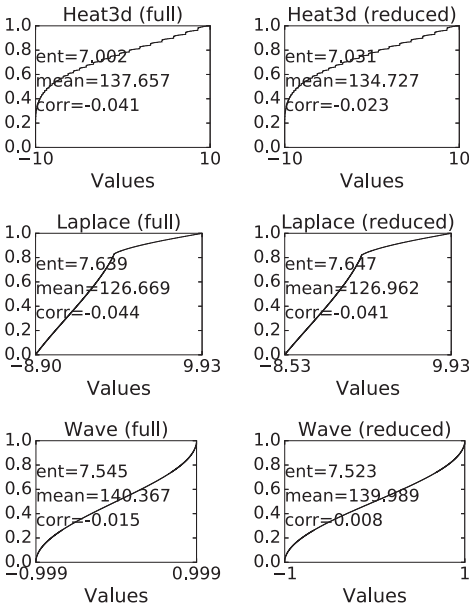


Fig. 1. Data characteristics of full model and reduced model. The curve shows the CDF of data values. ent, $mean$, and $corr$ denote the byte entropy, byte arithmetic mean, and serial correlation coefficient, respectively.

3.2 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 re-compute 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].

4 Projection-Based Reduced Model

In this section, we explore the mechanisms to build a reduced model using projection-based model reduction for PDE applications.

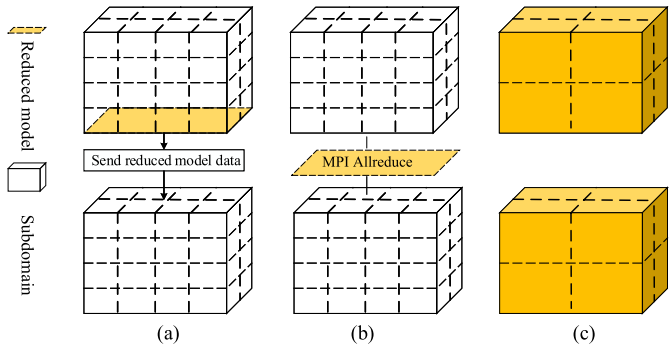


Fig. 2. Reduced models of a PDE application. a) *Mid-base*: the middle horizontal plane within the solution space is used as the reduced model; b) *Mean-base*: the horizontal plane that formed by the mean of Z-dimension of each (x, y) grid is used as the reduced model; c) *DuoModel*: a lower resolution of the original model is used.

4.1 Overview

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]. For example, we subtract each horizontal plane in the original 3-D solution space by the reduced 2-D plane to calculate the residuals. To reducing the overhead of running the reduced model, we reside it in the full model output itself, without constructing another application instance which can be cumbersome.

Fig. 2 illustrates various reduced models for a PDE application. We develop two ad-hoc projection-based schemes, termed as, *mid-base* and *mean-base*, to understand how well the proposed idea can perform. For *mid-base*, we choose a horizontal mid-plane from the full model and use 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 *mean-base*, we calculate the mean plane over all the planes in Z-dimension as the reduced model. Additional communications, such as MPI Allreduce, are required to calculate the means. 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]).

4.2 Mathematical Proof

In this section, we present a mathematical proof that reduced model can improve the compression ratio for PDE applications, including *Heat3d* and *Laplace* applications. The idea of this analysis applies to all solutions to differential equations suitable for separation of variables.

4.2.1 Heat3d

Mathematically, the 3-D 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 κ 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.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

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

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

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

In simulating the evolution of heat system, we have a 3-D grid $u[i][j][k]$ of the field of temperatures at time t. We focus on the compression ratio of the heat equation evolution data. Plus, we assume that $\kappa = 1$, $(x, y, z) \in [0, 1]^3$, and the cube was originally $T_{in} = -10$ degree at $t = 0$ and the environment keeps being $T_{out} = 10$ degree on the boundary of the cube.

From partial differential equation results, we can solve the problem, by separation of variables, in terms of

$$u(t,x,y,z) = T_{out} - \sum_{k_a,k_b,k_c} \Theta(k_a,k_b,k_c)\tau(t;k_a,k_b,k_c) \cdot \eta(x;k_a)\eta(y;k_b)\eta(z;k_c)$$

with $\tau(t;k_a,k_b,k_c) = e^{-((2k_a+1)^2+(2k_b+1)^2+(2k_c+1)^2)t}$ for $k_a, k_b, k_c \in \mathbb{N} = \{0,1,2,3,\ldots\}$ and $\eta(x;k) = \sin((2k+1)\pi x)$ for $k \in \mathbb{N}$. The coefficients $\Theta(k_a,k_b,k_c) = \frac{2^6(T_{out}-T_{in})}{\pi^3(2k_a+1)(2k_b+1)(2k_c+1)}$. Let

$$\Phi(t,x) = \sum_{k=0}^{\infty} \frac{1}{2k+1} e^{-(2k+1)^2 t} \sin((2k+1)\pi x),$$

the solution can be written as

$$u(t,x,y,z) = T_{out} - 2^6(T_{out}-T_{in})\pi^{-3}\Phi(t,x)\Phi(t,y)\Phi(t,z).$$

We compress the temperature data on a grid $u(t,x,y,z)$ for a fixed t in the following steps: (1) take values of u on a grid with fixed t, in terms of a 3-D array, (2) adjust values using a 2-D data for compression, (3) apply a lossy compression method to compress it. In our analysis, we take SZ as an example to analyze the performance. In step (2), a function $f(x,y)$ over xy grid plane is chosen and subtracted from each 3-D grid point according to xy-coordinate $(v(x,y,z) = u(x,y,z) - f(x,y))$. In step (3), the lexicographical order on (z,y,x) is applied, thus most of adjacent elements in the 1-D array are adjacent in x-direction in the original 3-D array. As those adjacent pairs which differs in y or z are on the constant temperature surface (T_{out}), they have no contribution in the unhit points of SZ.

Thus, when we analyze the hit rate in SZ, the only situation to consider is those points along x-direction lines. And consider that SZ uses extrapolation up to order 2 to estimate the "next" value, the error has a leading term of order 3 in Taylor expansion, which is $v_{xxx}(t,x,y,z)\mathrm{d}x^3/6$. So we may use $|v_{xxx}(t,x,y,z)| = |-2^6(T_{out}-T_{in})\pi^{-3}\Phi(t,y)\Phi(t,z)\Phi_{xxx}(t,x) - f_{xxx}(t,x,y)|$ to estimate the error term.

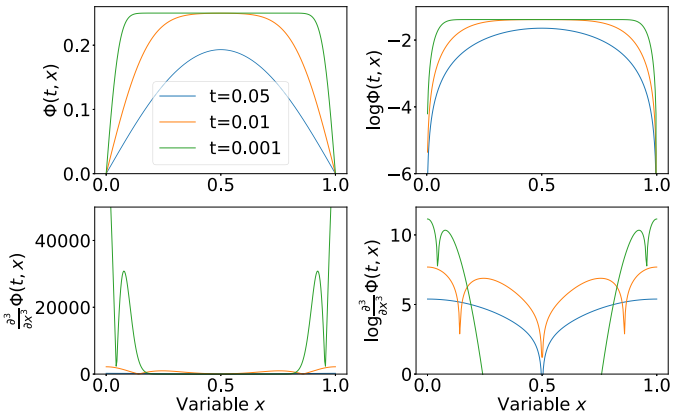


Fig. 3. The functions Φ and Φ_{xxx} for different t's.

Our task is to estimate the hit rate, equivalently, the number of points in the grid whose error term exceeds some error bound ϵ, which is a discrete version of the volume (measure) $\mathrm{Vol}(|v_{xxx}(t,x,y,z)| \geq \epsilon)$.

Fig. 3 shows $|\Phi_{xxx}(t,x)|$ and $\Phi(t,x)$ for $t = 0.001$, $t = 0.01$ and $t = 0.05$.

For a fixed t, we compare among three methods: (A) the raw data, (B) $f(x,y) = u(t,x,y,0.5)$ and (C) $f(x,y) = \int_0^1 u(t,x,y,z)dz$. Equivalently, the leading error term is proportional to

(A) $\Phi(t,y)\Phi(t,z)|\Phi_{xxx}(t,x)|$,
(B) $\Phi(t,y)|\Phi(t,z) - \Phi(t,0.5)||\Phi_{xxx}(t,x)|$,
(C) $\Phi(t,y)|\Phi(t,z) - \int_0^1 \Phi(t,z)\mathrm{d}z||\Phi_{xxx}(t,x)|$.

Take time t as a constant, we can define $F(\alpha) = \mathrm{Len}(|\Phi_{xxx}(t,x)| \geq e^\alpha) = \mathrm{Len}(\ln|\Phi_{xxx}(t,x)| \geq \alpha)$ and $G_s(\alpha) = \mathrm{Len}(|\Phi(t,x) - s| \geq e^\alpha) = \mathrm{Len}(\ln|\Phi(t,x) - s| \geq \alpha)$. And the volume we are looking for is the convolution $F * G_0' * G_s'(\ln\epsilon)$ with $s = 0$ for (A), $s = \Phi(t,0.5)$ for (B), and $s = \int_0^1 \Phi(t,z)\mathrm{d}z$ for (C). For convolutions, we have

$$\begin{aligned} F * G_0'(\alpha) &= \int_{-\infty}^{\infty} F(\alpha - l)G'(l)\mathrm{d}l \\ &= \int_{-\infty}^{\infty} \mathrm{Len}(\ln|\Phi(t,y)| \geq \alpha - l) \cdot \\ &\quad \mathrm{Len}(l \leq \ln|\Phi_{xxx}(t,x)| \leq l + \mathrm{d}l) \\ &= \mathrm{Area}(\ln|\Phi(t,y)| + \ln|\Phi_{xxx}(t,x)| \geq \alpha) \\ &= \mathrm{Area}(|\Phi_{xxx}(t,x)\Phi(t,y)| \geq e^\alpha). \end{aligned} \quad (2)$$

Convoluting again, we can conclude that $F * G_0' * G_s'(\ln\epsilon) = \mathrm{Vol}(|\Phi_{xxx}(t,x)\Phi(t,y)(\Phi(t,z) - s)| \geq \epsilon)$.

The estimated hit rate (the volume given by $1 - F * G_0' * G_s'(\ln\epsilon)$) for $t = 0.001$, $t = 0.01$, and $t = 0.05$ is shown in Fig. 4. We can see from the graph that when error bound is in certain regions, (especially when $t = 0.001$ which is closest to the simulations before) the hit rate given the middle section value is the best, and both middle section and mean value help improve the hit rate.

4.2.2 Laplace

Applying same method in the case of Laplace equation-problem, we analytically solve the equation $\Delta u(x,y) = 0$

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

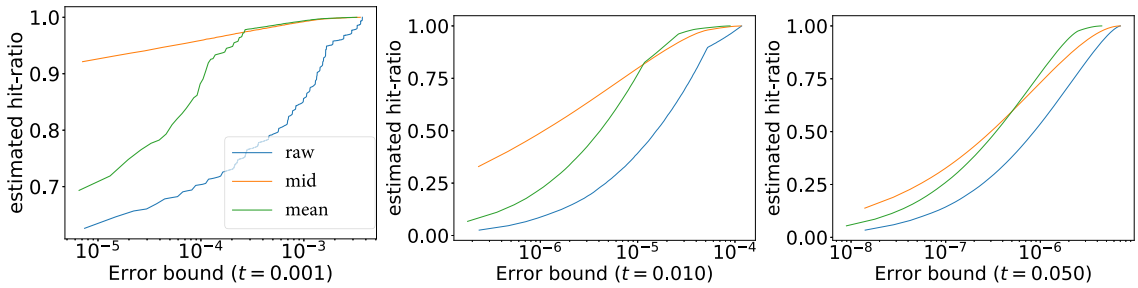


Fig. 4. Estimated hit rate versus error bounds, in three different t-values.

on $[0,1] \times [0,1]$ with boundary conditions $u(1,y) = 10$ and $u(0,y) = u(x,0) = u(x,1) = 0$. The solution is

$$u(x,y) = \sum_{k>0,k \text{ odd}} \frac{C \sinh(k\pi x) \sin(k\pi y)}{k \sinh(k\pi)} \quad (3)$$

with $C = 40/\pi$.

We compare two different ways of sequentializing the grid of values $u[x_i][y_j]$, by connecting the pieces along direction of x-axis (*x-first plan*) or along that of y-axis (*y-first plan*), on the hit rate of SZ compression algorithm. As before, the hit rate of SZ is determined by the numerical third derivative of the sequence. We have

$$u_{xxx}(x,y) = \sum_{k>0,k \text{ odd}} \frac{k^2\pi^2 C}{\sinh(k\pi)} \cosh(k\pi x) \sin(k\pi y) \quad (4)$$

$$u_{yyy}(x,y) = -\sum_{k>0,k \text{ odd}} \frac{k^2\pi^2 C}{\sinh(k\pi)} \sinh(k\pi x) \cos(k\pi y) \quad (5)$$

Fig. 5 shows the third derivatives. From the graphs of u_{xxx} and u_{yyy}, the estimated errors of SZ compression whose distribution determines hit rate range mostly from 1 to 100 in their absolute values, making the possible reduce-methods most effective in the range $(stepSize^{-3}, 100 stepSize^{-3})$ where $stepSize$ is the dimension on one direction (in this problem, $stepSize \approx 10^3$, thus the range becomes $(10^{-9}, 10^{-7})$). Since the solution is no longer factorizable into functions on x and y, we calculate the influence of the mid-base and mean-base methods to the estimated error distribution and plot them in CDF sense in Fig. 6.

The estimation suggests that the mid-base reduction on x-first plan is the best choice among all 6 candidates. Moreover, it is the best one for all error bounds, with most improvement occur in the region proposed before.

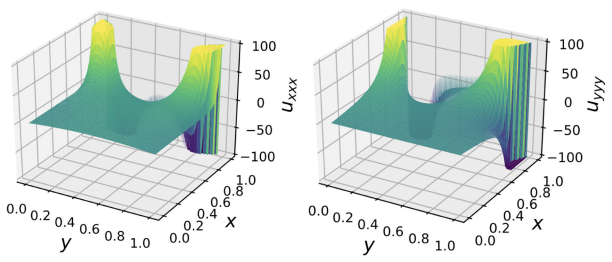


Fig. 5. Third derivatives with a cutoff range $(-100, 100)$.

4.3 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 two lossy compressors to understand more broadly how much projection-based reduced models can improve compression ratios. In particular, we use SZ 2.1.8 with the default mode and absolute error bound. The number of quantization intervals is set to 0, such that SZ will automatically search for an optimized setting. For ZFP, we use the newest version ZFP 0.5.5 with absolute error bound. To be able to test one error bound that covers the entire range of a dataset, we use the product of the relative error bound and the range of each dataset to determine the real absolute error bound for the dataset. Thus, the absolute error bound is given as: $\eta \cdot (max - min)$, where η is the relative error bound, and max and min are the maximum and minimum values. In the experiments, the default relative error bound is 10^{-5} [22].

Fig. 7 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*. On average, the compression ratios of ZFP increase from 4X to greater than 10X for *mid-base*. Using SZ to compress the preconditioned data, the compression ratios increase from 40X to greater than 90X for *mid-base* and greater than 55X for *mean-base*. In addition, *mid-base* and *mean-base* outperform *DuoModel*. The reason is that the delta generated by the *mid-base* and *mean-base* are with more smoothness than those generated by *DuoModel*.

To better study how the reduced model improves the compression ratio, we collect the values of intermediate metrics of ZFP and SZ during the compressions. For the convenience of discussion, the description of the metrics are listed in Table 2. For ZFP, the size of compressed data is the sum of sizes of all compressed blocks, given that ZFP partitions the entire dataset into fixed-sized blocks. The important metrics are $ZeroCnt$, $BitsPerBitplane$, $MaxPrec$, and $BlockSize$. In particular, if a block is with all zeros, $BlockSize$ is one since only a single bit of zero is written. Otherwise, $BlockSize$ is the sum of $BitsBitplane$ and $BitsExp$. For a non-zero block, $BitsBitplane = \sum_{j=0}^{MaxPrec-1} BitsPerBitplane_j$, where $BitsPerBitplane_j$ is the jth bit plane to encode for the block, and $MaxPrec$ is calculated as $MaxPrec = min(64, MaxExp - log_2(Errorbound) + 2(dim + 1))$, where dim is the number of dimensions. For SZ, the size of compressed data is the sum of size of the

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

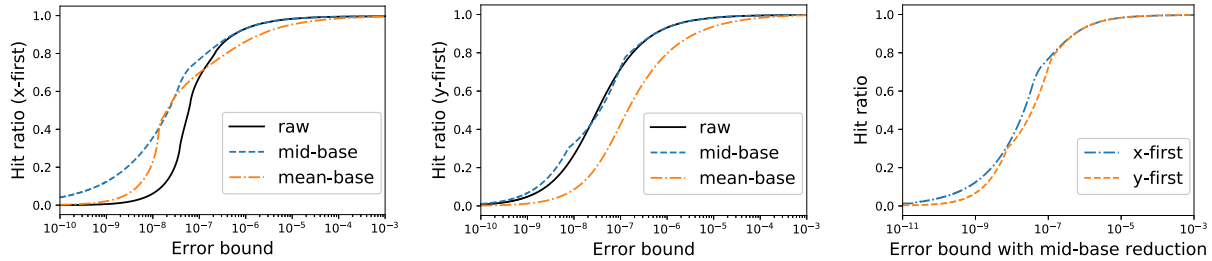


Fig. 6. The hit rates of the three methods in x-first plan and y-first plan and the comparison between two mid-base reductions.

Huffman tree, size of Huffman coding, and size of curve-missed points. Thus, the important metrics are $HitRatio$, $QuantIntv$, $TreeSize$, $EncodeSize$, and $OutlierSize$. In particular, all the other four metrics are related to $HitRatio$. Further details regarding the compression algorithms can be found in prior work [8], [9].

Fig. 8 measures the internal metrics of ZFP. Due to the space limit, we only show the results of *Heat3d*. As compared to the raw data, *mid-base* increases $ZeroCnt$ from 0 to 691,368. However, with the number of total data points is 7,077,888, the contribution to the overall improvement of compression ratio is only about 1%. The $BitsBitplane$ of *mid-base* has a substantial reduction from 55 to 34. The reason is that *mid-base* improves the data smoothness, and more bits in a bit plane after the orthogonal transform will likely be zero, resulting in the reduced $BitsBitplane$. At last, it is also found that for *mean-base* and *DuoModel*, their reductions of $ZeroCnt$, $BitsBitplane$, and $BlockSize$ are not obvious as *mid-base*.

Fig. 9 measures the internal metrics of SZ. As compared to the raw data, $HitRatio$ is improved for *mid-base* and *mean-base* as a result of the improved smoothness. For example, for Z-order, $HitRatio$ is improved from 0.689 to 0.938 and 0.886 for *mid-base* and *mean-base*, respectively, further detailed in Fig. 10. This in turn lowers $OutlierSize$, which is the size of curve-missed points that are typically hard to compress. For example, $OutlierSize$ of *mid-base* is reduced from 2,199,132 by more than 50%. It is also found that all the four methods have similar $TreeSize$. The reason is that SZ applies the same $QuantIntv$. The number of tree nodes in the Huffman tree is $2 \cdot QuantIntv - 1$, and therefore we observe similar size of the Huffman tree ($TreeSize$). At last, with the improved data smoothness, $EncodeSize$ of *mid-base* and *mean-base* are substantially smaller as compared to raw data. Overall, *mid-base* and *mean-base* outperforms the original and *DuoModel* in terms of the reductions of $EncodeSize$ and $OutlierSize$.

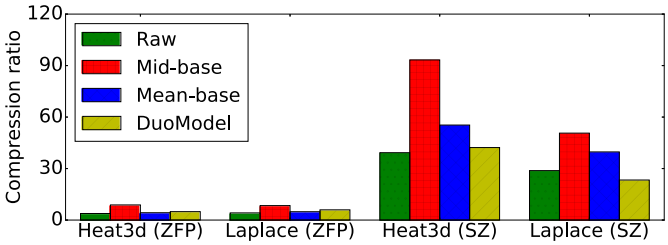


Fig. 7. Compression ratios using projection-based methods. The x-axis is the test setup (i.e., dataset name and compressor used. In this paper, raw means to use the compressors directly).

TABLE 2
A List of Internal Compression Metrics

Symbols	Description
	ZFP
BlockCnt	Number of blocks
ZeroCnt	Number of blocks that are with all zeros
MaxExp	The common (largest) exponent of each block
MaxPrec	Maximum number of bit planes to encode in order to meet the accuracy demand
BitsBitplane	Number of bits used in encoding bitplane
BitsExp	A fix number of bits represent the exponents 11 for double-precision floating-point data 8 for single-precision floating-point data
BlockSize	Size of each block data (in bits)
	SZ
HitRate	Curve-fitting hit rate
QuantIntv	Number of quantization intervals
NodeCnt	Number of Huffman tree nodes
TreeSize	Size of Huffman tree (in bytes)
EncodeSize	Total size of Huffman coding (in bytes)
OutlierSize	Total size of curve-missed points (in bytes)

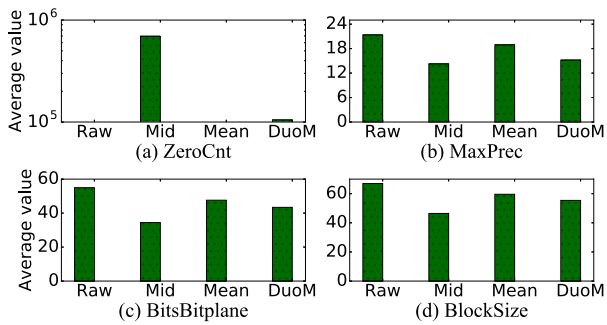


Fig. 8. ZFP metrics.

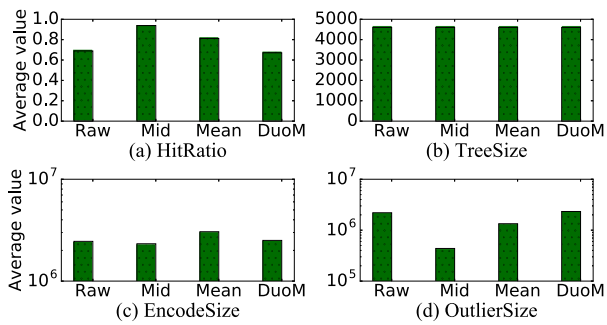


Fig. 9. SZ metrics.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

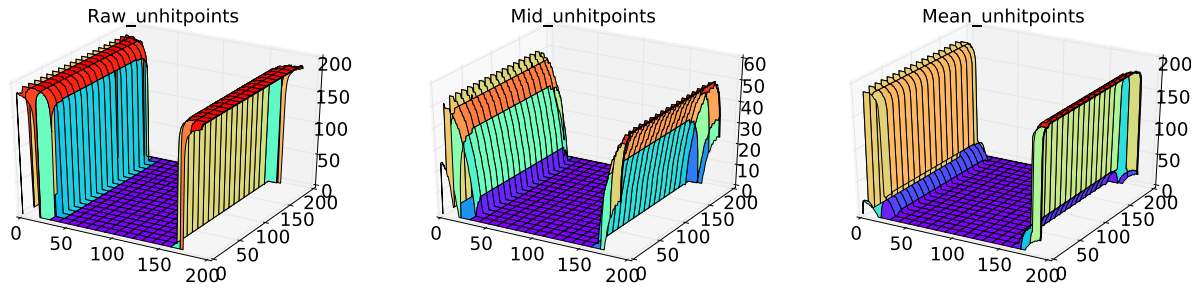


Fig. 10. Number of unhit points of the Z-dimension. Different colors represent the distribution of the unhit points of the Z-dimension on XY plane.

We also evaluate how the reduced model impact the hit rate of SZ. As shown in Fig. 10, we track if a data point is hit or unhit by the curve-fitting in SZ. Fig. 10 shows cumulative graph of the unhit cases of whole cube projection on xy plane. It is found that all the unhit cases are distributed in surrounding locations, the reason is that these locations have most dramatic change, which is shown in the Fig. 3. For the comparison among the three methods, both *mid-base* and *mean-base* helps improve the hit-ratio, and *mid-base* has the lowest unhit rate.

5 Exploring General Reduced Models

For datasets that model sophisticated problems with complex geometry, identifying the reduced model may not be as straightforward as *mid-base* and *mean-base* in Section 4.1. Fortunately, scientific data are naturally multi-dimensional, capturing physical quantities in both space and time[23]. A straightforward idea is to further explore common dimension reduction techniques to extract reduced model. Unlike *mid-base*, whose data outputs of the reduced model 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. 11 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.

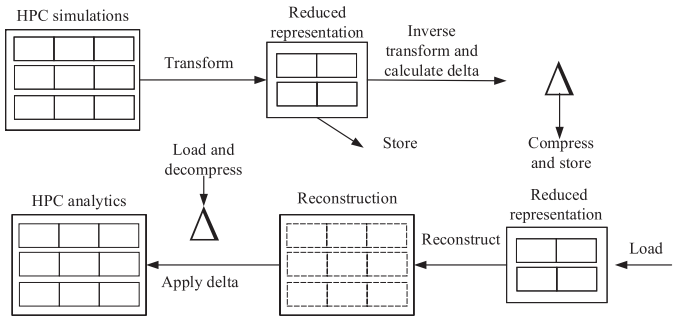


Fig. 11. Data reduction and reconstruction using dimension reduction techniques. Different colors represent the distribution of the unhit points of the Z-dimension on x0y planes.

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.

5.1 Adopting Dimension Reduction Techniques

5.1.1 PCA

PCA uses an orthogonal transformation to convert correlated variables into a set of uncorrelated variables, called principal components [24]. 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.

5.1.2 SVD

In contrast to PCA that only focuses on the column space, SVD focuses on both the column and row space [25]. The singular-value decomposition of a matrix A can be represented as $U \cdot \Sigma \cdot V^*$, and the diagonal entries σ_i of Σ 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 columns of U and k rows of V^*.

5.1.3 Wavelet

Discrete wavelet transform [26] decomposes a signal into mutually orthogonal sets of wavelet basis functions. In this work, we apply Haar Wavelet [27] 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 θ to zeros.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

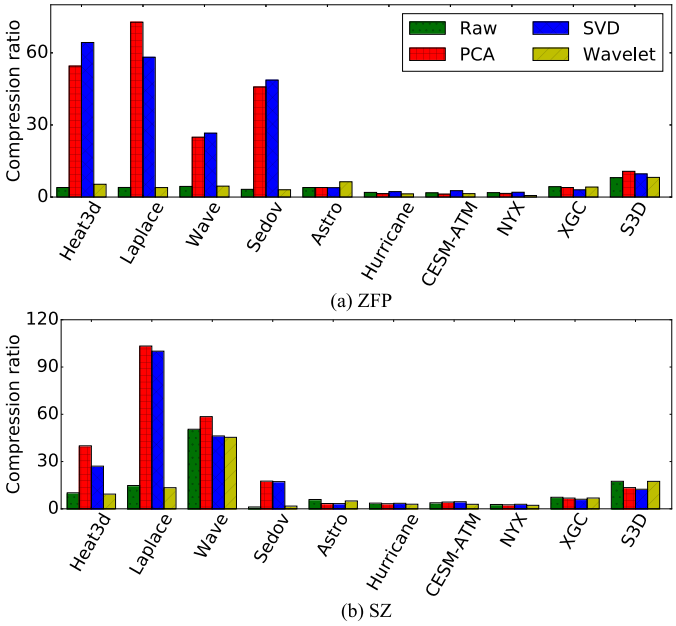


Fig. 12. Comparison of compression ratios. The legends show the conjunctions of the three dimension reduction techniques with ZFP and SZ, respectively.

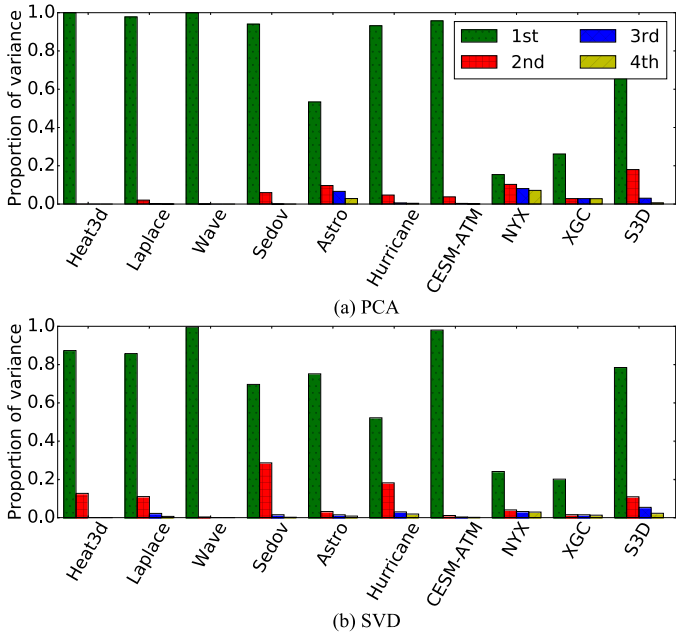


Fig. 13. PCA (or SVD) proportion of variance of the largest four primary components (or singular values).

The new matrix in principle is sparser and can be stored more efficiently. The new sparse matrix is regarded as the reduced representation.

5.2 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. For the runtime results, the experiments were conducted on a more powerful machine, a Linux server with Intel(R) $Core^{TM}$ i7-9700 that has 8 cores with a frequency of 3.00GHz and 32 GB of memory. The operating system is Ubuntu 20.04.1 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 4.3. We select the number of components, k, such that $\sum_{i=1}^{k}\sigma_i/\sum_{i=1}^{n}\sigma_i \geq 95\%$[28], in which σ_i is the variance of the ith largest primary components (or singular values) for PCA (or SVD). For the impact on data loss, the higher the value of k is, the less the loss of information in a reduced model is. Finally, for Wavelet, we set the threshold θ to 5% of the maximum value in the transformed matrix.

5.2.1 Compression Ratio

Fig. 12 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*, and *Sedov_pres*. However, the improvement for other datasets is insignificant. In particular, for *XGC* data, PCA, SVD, and Wavelet all result in lower compression ratios than compressing it directly. To further understand the different outcomes across datasets, Fig. 13 shows 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.

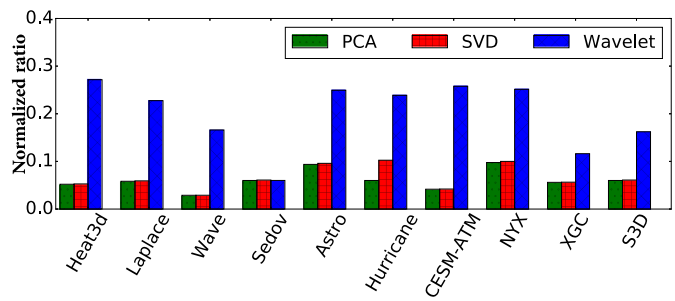


Fig. 14. The size of reduced representations normalized to the full datasets.

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. 14 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 θ to further reduce the reduced representation size of Wavelet. However, the associated delta will become less compressible, which offsets the overall improvement.

We also show the impact of the reduced model techniques on the ZFP and SZ compressors. Fig. 15 shows the compression ratio brought by PCA and SVD only. As shown in the figure, it is found that for some datasets, such as *Heat3d* and *Laplace*, the raw PCA and SVD have 50% of the contribution, and the reduced model will enlarge the compression ratio by

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

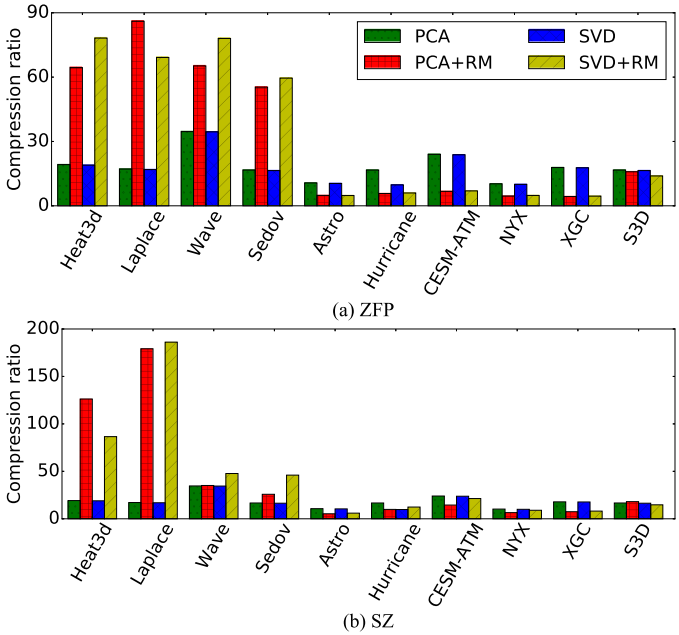


Fig. 15. The impacts of raw PCA and SVD. RM means reduced model.

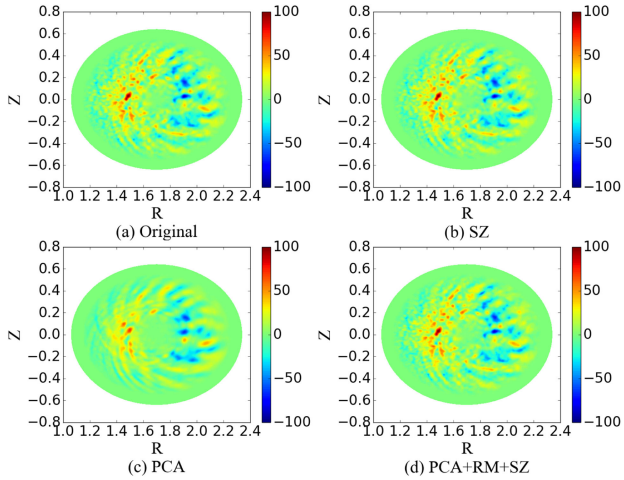


Fig. 16. Visualization of decompressed XGC dataset at the compression ratio of 20X.

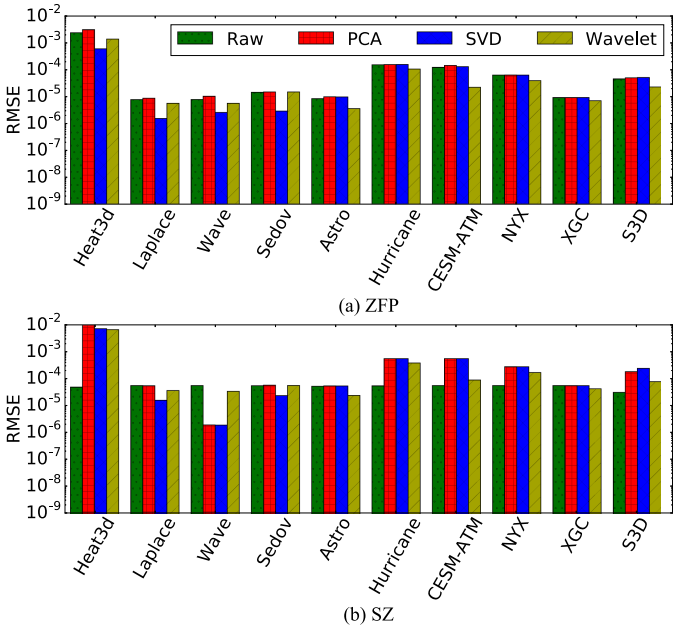


Fig. 17. Comparison of RMSE introduced by different kinds of methods, including the eight conjunctions of the three reduced models with ZFP and SZ.

further compressing the reduced representation. But for other datasets, such as NYX and XGC, compression ratio is less than the raw PCA and SVD, the reason is that PCA/SVD leads to too much information loss. In this case, our methods will make compensation by storing the deltas.

5.2.2 Evaluation of Information Loss

Information loss is another importance metrics to measure the effectiveness of the proposed methods. We first present the quality of visualization from decompressed data. Then, we evaluate the RMSE results of the proposed methods.

Fig. 16 illustrates the visualization results of the decompressed data using raw SZ, raw PCA, and the proposed reduced model with SZ (PCA+RM+SZ). It is found under the compression ratio of 20X, the visualization effect of SZ and PCA+RM+SZ are almost the same as the original data. However, the visualization effect of raw PCA is obviously different from the original data. This means that if PCA is simply applied, the fidelity loss caused by PCA has obvious impact at application level.

Fig. 17 shows the RMSE results of PCA, SVD, and Wavelet, respectively. Overall Wavelet yields a similar RMSE with compressing data directly for most datasets. Given the insignificant compression ratios, 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.

5.2.3 Overhead Analysis

Fig. 18 shows the average compression and decompression time. Compared to compressing data directly using ZFP, PCA, SVD, and Wavelet increase the compression time by 10.5X, 16.6X, and 2.1X, 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 1.5X, 1.7X, and 1.1X, respectively. It is clear that compression is more expensive than decompression, which can be attributed to the expensive matrix decomposition during compression.

5.2.4 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 on the

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

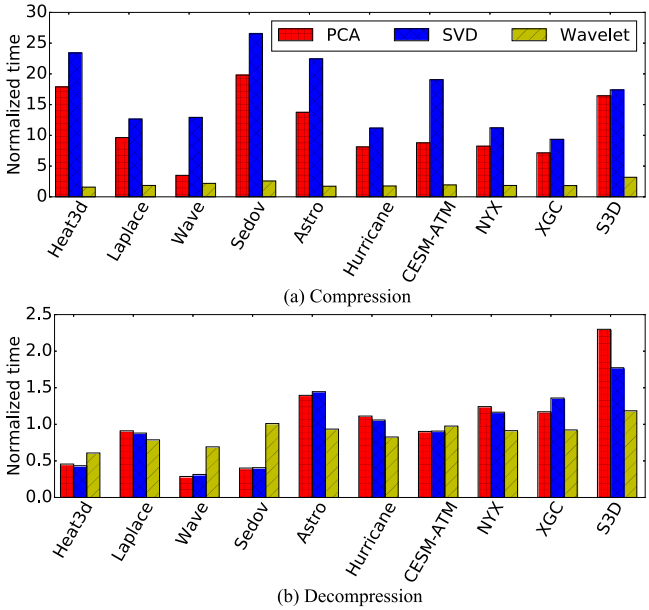


Fig. 18. Compression and decompression overhead.

supercomputer Titan[4] with 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 3, 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) [29], [30]. 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 3 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 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 3 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.

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

TABLE 3
Compression and I/O Time

Time	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

6 Model Selection

As shown in Fig. 12, there is no single method that can consistently yield the best performance. For example, PCA in conjunction with SZ provides the best compression ratios for *Laplace* and *Wave*. However, for *Heat3d*, SVD with ZFP achieves the best performance. For *S3D*, it is desirable to compress data directly using SZ. In reality, domain scientists need guidance to select the best method prior to reducing their data. One could in theory exhaust all combinations for a particular dataset and select the best one. However, this can be very computationally expensive for large datasets. This section tackles this problem using a sampling-based model selection, with the goal of enabling users to select the best method without trial-and-error.

Algorithm 1. Reduced Model and Compressor Selection

Require: Dataset A.
Ensure: The selected reduced model and/or compressor.
1: Generate the sampled data A_{sp} with a given sampling ratio R_{sp}.
2: Identify the reduced model using PCA, SVD, and Wavelet based upon A_{sp}.
3: Calculate the reduced representation size ratios normalized to the original data of the three methods as φ_1, φ_2, and φ_3, respectively.
4: Calculate the sums of variance of primary component of PCA and SVD as ξ_1 and ξ_2.
5: Calculate the data loss as $loss_{i=1,2} = (1 - \xi_i^2)$. For Wavelet, the data loss is defined as $loss_3 = \frac{\theta}{MaxValue}$.
6: Choose the reduced model with minimal values of $min\{loss_i \cdot \varphi_i\}$.
7: Obtain the compression ratios of A_{sp} of ZFP and SZ with and without the selected reduced model.
8: Output the reduced model and/or the compressor that yield a higher compression ratio.

6.1 Method

Our method is based upon the following two observations. First, sampling is generally a good strategy to extract the properties of data. A key advantage of using sampling in the context of this work is that the size of sampled data can be significantly smaller than that of the full data, and therefore estimation on top of the sampled data can be computationally cheaper than directly operating on the full data. Based upon this observation, we first down-sample the full data, and then use the compression performance of the sampled data to

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

TABLE 4
Average Relative Error of Information Loss $loss_i$ in the Reduced Models, Where $loss_i$ is Defined in Algorithm 1

Method	Ratio	$\mathbf{R_{sp} = 1\%}$			$\mathbf{R_{sp} = 5\%}$			$\mathbf{R_{sp} = 10\%}$		
		PCA	SVD	Wavelet	PCA	SVD	Wavelet	PCA	SVD	Wavelet
PS		0.4870	0.3598	0.6343	0.1376	0.3158	0.6318	0.1388	0.3428	0.5583
US		0.2359	0.2505	0.5051	0.0814	0.1759	0.3436	0.0717	0.1675	0.1037
RS		0.0823	0.1885	0.4146	0.0321	0.1488	0.1052	0.0631	0.1402	0.1079

extrapolate that of the full data. Second, intuitively whether a reduced model is good or not depends on the following factors: 1) the amount of information that can be retained by the reduced model. The higher the similarity between the reduced model and full model is, the higher the reduction ratio can be. For PCA (or SVD), it refers to the portion of variances of the chosen primary components (or singular values). For Wavelet, it refers to the remaining information of the transformed matrix after the thresholding; 2) the ratio of the size of the reduced representation to that of the full model. With the same amount of information captured, the smaller the reduced representation is, the better the reduced model is. The sizes of the reduced representations of PCA, SVD and Wavelet refer to the size of primary component eigenvectors, the total size of three refactored matrices, and the number of nonzero entries in the sparse matrix, respectively.

Our model selection algorithm adopts information loss and the reduced representation size (normalized to the original data size) as the two metrics to select the best reduced model. In particular, a method that yields the minimum product of information loss and reduced representation size is chosen. We use $loss_i$ to denote the information loss incurred by method i, where $i = 1, 2, 3$ for PCA, SVD, and Wavelet, respectively. φ_i denotes the normalized size of the reduced representation. For PCA and SVD, the sum of variance for the primary components are denoted as ξ_1 and ξ_2, respectively.

Our algorithm, as shown in Algorithm 1, consists of three major steps. First, we sample the dataset A with a given sampling ratio R_{sp} (Line 1, with more details in Section 6.2). Next, based upon the sampled data A_{sp}, we determine the reduced model to be adopted by choosing the one that minimizes the product of information loss and reduced representation size ratio, i.e., $min\{loss_i \cdot \varphi_i\}$ (Lines 2-6). For PCA or SVD, the information loss is defined as $loss_i = 1 - \xi_i^2, i = 1, 2$. For Wavelet, we use $loss_3 = \frac{\theta}{MaxValue}$, where $MaxValue$ is the maximum value of the original matrix. To calculate the normalized size of the reduced representation, we assume that A_{sp} is a $m \times n$ matrix. For PCA, we have $A_{sp,m\times n} \approx X_{m\times k} C_{k\times n}$. The normalized reduced representation size is $\varphi_1 = k(m+n)/mn$. For SVD, we have $A_{sp,m\times n} \approx U_{m\times k}\Sigma_{k\times k}V^*_{k\times n}$. To obtain the reduced representation, we only retain k largest singular values, and the corresponding k columns of U and k rows of V^*. Thus, the normalized reduced representation size is $\varphi_2 = k(m+k+n)/mn$. For Wavelet, we have the sparse matrix as the reduced representation. The normalized reduced representation size is $\varphi_3 = Count\_nonzero(A_{sp})/mn$, where $Count\_nonzero(A_{sp})$ counts the number of non-zero values in the matrix A_{sp}. The last step is to identify the backend compressor to be further used in conjunction with the selected reduced model, or be used independently without the reduced model (Lines 7-8).

6.2 Sampling Method Analysis

Prefix sampling (PS), uniform sampling (US) and random sampling (RS) are commonly used to sub-sample data. In particular, PS selects the front part of data until a predefined sampling ratio is met. In contrast, US sub-samples data uniformly with a given interval, while RS selects each element with equal probability. To evaluate the effectiveness of these sampling methods, we evaluate the information loss in the reduced model and measure the compression ratio of the sampled data using ZFP and SZ. Table 4 shows the relative error of information loss under PCA, SVD and Wavelet with different sampling ratios. This metric is defined as $\eta_1 = |loss_{full} - loss_{sp}|/loss_{full}$, where $loss_{full}$ and $loss_{sp}$ are the information loss in the reduced model from the full and sampled data, respectively. A single result in the table is the average relative error across ten datasets.

With the sampling ratio of 1%, all three sampling methods have high errors for PCA, SVD, and Wavelet. However, as the sampling ratio increases, the relative error of information loss decreases. Among PCA, SVD and Wavelet, the relative error of PCA and SVD are more sensitive to the sampling ratio than that of Wavelet. Another finding is that US and RS outperform PS, especially when the sampling ratio increases to 10%. The reason is that PS focuses on front portion of data, thus mostly likely lose important information as compared to US and RS.

Table 5 shows the average relative error of compression ratio using SZ and ZFP. Similarly, the relative estimation error is defined as $\eta_2 = |CR_{full} - CR_{sp}|/CR_{full}$, where CR_{full} and CR_{sp} are the compression ratios of the full data and sampled data directly using ZFP or SZ, respectively. It is shown that the sampling-based compression ratio estimation works well for ZFP. However, it results in large estimation errors for SZ. This discrepancy between the two compressors was reported and explained in prior work [16].

6.3 Results

Table 6 shows the results of the sampling-based selection method, including the relative error of compression ratio and the method selection hit rate. The relative estimation error is

TABLE 5
Average Relative Error of ZFP and SZ Compression Ratio Estimation

	$\mathbf{R_{sp} = 1\%}$		$\mathbf{R_{sp} = 5\%}$		$\mathbf{R_{sp} = 10\%}$	
	ZFP	SZ	ZFP	SZ	ZFP	SZ
PS	0.4424	0.7444	0.1851	0.7051	0.1123	0.5342
US	0.0323	0.3855	0.0234	0.2104	0.0188	0.1772
RS	0.0513	0.3761	0.0212	0.2249	0.0206	0.1669

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

TABLE 6
Estimation Error and Method Selection Hit Rate

	$R_{sp} = 1\%$		$R_{sp} = 5\%$		$R_{sp} = 10\%$	
	ZFP	SZ	ZFP	SZ	ZFP	SZ
PS+PCA	0.5739	0.4244	0.3442	0.3939	0.2668	0.3021
PS+SVD	0.5037	0.5581	0.3359	0.4572	0.3110	0.3887
PS+Wavelet	0.6818	0.6437	0.2581	0.4620	0.1545	0.3963
US+PCA	0.3736	0.4904	0.2440	0.4184	0.2702	0.3677
US+SVD	0.3491	0.5019	0.3243	0.3702	0.3373	0.3393
US+Wavelet	0.1703	0.4371	0.1241	0.2995	0.0938	0.2542
RS+PCA	0.4056	0.4558	0.2817	0.4170	0.2262	0.3543
RS+SVD	0.3459	0.5067	0.3379	0.3774	0.3387	0.3379
RS+Wavelet	0.1605	0.4656	0.1089	0.3033	0.0883	0.2622
RS Hit Rate	61.0%		69.6%		73.2%	

defined as $\eta_3 = |CR_{full} - CR_{sp}|/CR_{full}$, where CR_{full} and CR_{sp} are the final compression ratios of the full data and sampled data for a given reduced model, respectively. The hit rate is defined as the fraction of correct selections, which means the model selected based upon the sampled data is consistent with the model selected based upon the full data. The first ten rows show the average relative error of the final compression ratio across the ten datasets for a reduced model along with a sampling method. We notice that US has the best performance for information loss as well as the most accurate compression estimation. Therefore, we choose US as the sampling method in Algorithm 1. The last row in Table 6 shows the effectiveness of model selection. With a decent sampling ratio (e.g., 5% or higher), the hit rate of RS can be a high as 69.6%. This suggests that sampling-based estimation can provide a good indicator for the potential model to be used, despite the substantial loss of data.

7 Related Work

As far as we are aware, DuoModel [15] is the first one that uses reduced models to improve the compression ratio of existing compressors. This work is proposed to address the general issues of DuoModel. The most related work to this paper are summarised into two categories. The first one is preconditioning the data. Hübbe et al. [31] proposed MAFISC to reduce the HPC-datastorage footprint. The authors developed and tested different filters to precondition the climate data for lossless compressors. Lakshminarasimhan et al. [7] found that the scientific data are random. The compression ratio can be improved if the data are sorted. Thus, they proposed ISABELA to sort the data and keep tracking the new position indexes to recover the original order. Schendel et al. [32] proposed ISOBAR, which separates a dataset into compressible and incompressible segments. The compressible and incompressible segments are treated differently. Metadata are introduced to recover the original data. Bicer et al. [33] developed a new compression methodology, which sorts all floating point values and calculated the delta of neighbors. The compression and decompression schemes support bitwise operations such that compression and decompression are in high speeds. Austin et al. [23] 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. [34] 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. [35] 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.

8 Conclusion

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 PDE applications 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. We provide a mathematical proof that our method can improve the local smoothness of a dataset that are calculated by PDEs. More general dimension reduction techniques, including PCA, SVD, and Wavelet, are then explored for the purpose of data compression. Besides, time series data are also explored in the paper. We evaluate various reduced methods on ten scientific datasets, and the results show the effectiveness of our approach. However, the results also show that there is no a single reduced model method is best of all the datasets. Therefore, we propose a model selection strategy that selects an appropriate model prior to data reduction.

References

[1] M. Burtscher, H. Mukka, A. Yang, and F. Hesaaraki, "Real-time synthesis of compression algorithms for scientific data," in *Proc. IEEE Int. Conf. High Perform. Comput., Netw. Storage Anal.*, 2016, pp. 264–275.
[2] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Commun. ACM*, vol. 58, no. 7, pp. 56–68, 2015.
[3] S. Ku et al., "Gyrokinetic particle simulation of neoclassical transport in the pedestal/scrape-off region of a tokamak plasma," in *Proc. J. Phys. Conf. Ser.*, 2006, Art. no. 87.
[4] I. Foster et al., "Computing just what you need: Online data analysis and reduction at extreme scales," in *Proc. Eur. Conf. Parallel Process.*, 2017, pp. 3–19.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.

[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 Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1857–1871, Aug. 2019.
[6] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 18–31, Jan. 2009.
[7] S. Lakshminarasimhan et al., "Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data," in *Proc. Eur. Conf. Parallel Process.*, 2011, pp. 366–379.
[8] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 2674–2683, Dec. 2014.
[9] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 730–739.
[10] A. M. Gok et al., "PaSTRI: Error-bounded lossy compression for two-electron integrals in quantum chemistry," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2018, pp. 1–11.
[11] W. Xia et al., "A comprehensive study of the past, present, and future of data deduplication," *Proc. IEEE*, vol. 104, no. 9, pp. 1681–1710, Sep. 2016.
[12] Q. Zheng, K. Ren, G. Gibson, B. W. Settlemyer, and G. Grider, "DeltaFS: Exascale file systems scale better without dedicated servers," in *Proc. 10th Parallel Data Storage Workshop*, 2015, pp. 1–6.
[13] J. C. Bennett et al., "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *Proc. IEEE Int. Conf. High Perform. Comput., Netw. Storage Anal.*, 2012, pp. 1–9.
[14] P. Ghysels and W. Vanroose, "Hiding global synchronization latency in the preconditioned conjugate gradient algorithm," *Parallel Comput.*, 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 Lett. Comput. Soc.*, vol. 1, no. 1, pp. 5–8, Jan.-Jun. 2018.
[16] T. Lu et al., "Understanding and modeling lossy compression schemes on HPC scientific data," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 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 Rev.*, vol. 57, no. 4, pp. 483–531, 2015.
[18] W. H. Schilders, H. A. Van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, vol. 13. Berlin, Germany: Springer, 2008.
[19] A. Antoulas, "Approximation of large-scale dynamical systems: An overview," *IFAC Proc. Volumes*, vol. 37, no. 11, pp. 19–28, 2004.
[20] T. Cui, Y. M. Marzouk, and K. E. Willcox, "Data-driven model reduction for the Bayesian solution of inverse problems," *Int. J. Numer. Methods Eng.*, vol. 102, no. 5, pp. 966–990, 2015.
[21] J. Hernández, J. Oliver, A. E. Huespe, M. Caicedo, and J. Cante, "High-performance model reduction techniques in computational multiscale homogenization," *Comput. Methods Appl. Mechanics Eng.*, vol. 276, pp. 149–189, 2014.
[22] J. Tian et al., "waveSZ: A hardware-algorithm co-design of efficient lossy compression for scientific data," in *Proc. 25th ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, 2020, pp. 74–88.
[23] W. Austin, G. Ballard, and T. G. Kolda, "Parallel tensor compression for large-scale scientific data," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 912–922.
[24] P. Forczmański and W. Maleika, "Near-lossless PCA-based compression of seabed surface with prediction," in *Proc. Int. Conf. Image Anal. Recognit.*, 2015, pp. 119–128.
[25] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "TTHRESH: Tensor compression for multidimensional visual data," *IEEE Trans. Visualization Comput. Graph.*, vol. 26, no. 9, pp. 2891–2903, Sep. 2019.
[26] D. Gupta and S. Choubey, "Discrete wavelet transform for image processing," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 4, no. 3, pp. 598–602, 2015.
[27] C. Mulcahy, "Image compression using the haar wavelet transform," *Spelman Sci. Math. J.*, vol. 1, no. 1, pp. 22–31, 1997.
[28] 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," *Comput. Statist. Data Anal.*, vol. 49, no. 4, pp. 974–997, 2005.
[29] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "DataStager: Scalable data staging services for petascale applications," *Cluster Comput.*, vol. 13, no. 3, pp. 277–290, 2010.
[30] T. Jin et al., "Exploring data staging across deep memory hierarchies for coupled data intensive simulation workflows," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2015, pp. 1033–1042.
[31] N. Hübbe and J. Kunkel, "Reducing the HPC-datastorage footprint with MAFISC–multidimensional adaptive filtering improved scientific data compression," *Comput. Sci. Res. Develop.*, vol. 28, no. 2, pp. 231–239, 2013.
[32] E. R. Schendel et al., "Isobar preconditioner for effective and high-throughput lossless data compression," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 138–149.
[33] T. Bicer, J. Yin, D. Chiu, G. Agrawal, and K. Schuchardt, "Integrating online compression to accelerate large-scale data analytics applications," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process.*, 2013, pp. 1205–1216.
[34] J. Choi, X. Liu, and V. Chakaravarthy, "High-performance dense tucker decomposition on GPU clusters," in *Proc. IEEE Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2018, pp. 543–553.
[35] X. Liang et al., "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 1–10.

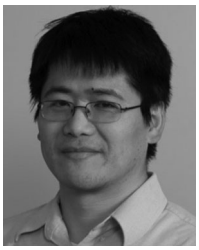
Huizhang Luo received the BS and Ph.D degrees in computer science from Chongqing University, China, in 2012 and 2017, respectively. He is currently an associate professor in Hunan University. Before that, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering with NJIT. His research interests include memory systems, high-performance computing, and non-volatile memory.

Junqi Wang received the BS degree in physics with Zhejiang University, in 2010 and the PhD degree in mathematics with Rutgers University - Newark, in 2018. He is a visiting scholar with Hunan University. He was a postdoctoral associate at Department of Mathematics and Computer Science with Rutgers University - Newark. His current research interests are cooperative machine learning, Bayesian theory and optimal transport theory.

Zhenlu Qin received the BS and MS degree in computer science and technology from the Hefei University of Technology, China, in 2014 and 2017, respectively. He is currently working toward the PhD degree in Department of Electrical and Computer Engineering, New Jersey Institute of Technology.

Dan Huang received the bachelor's and master's degrees from Jilin University and Southeast University, respectively, and PhD degree in computer engineering with University of Central Florida. He is a research associate in NJIT. His research interests are distributed storage systems, virtualization technology and the I/O of distributed system.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.



Qing Liu received the PhD degree in computer engineering from the University of New Mexico, in 2008. He is an associate professor in the Department of Electrical and Computer Engineering with NJIT. He has joint faculty appointment with Oak Ridge National Laboratory. Prior to that, he was a staff scientist with Science Data Group, Oak Ridge National Laboratory. His areas of interest include high-performance computing, data science, and networking. In 2013 He won an R&D 100 Award for the development of the Adaptable I/O System for Big Data. He has published numerous papers in high-quality conferences and journals, including SC, SIGMETRICS, IPDPS, Cluster, ICDCS, HPDC, TPDS, TC, and TOS.

Mengchu Zhou (Fellow, IEEE) received the PhD degree from Rensselaer Polytechnic Institute, Troy, NY, in 1990 and then joined New Jersey Institute of Technology where he is now a Distinguished Professor. His interests are in Petri nets, automation, Internet of Things, cloud/edge computing, and AI. He has 1000+ publications including 12 books, 700+ journal papers (600+ in IEEE transactions), 31 patents and 30 book-chapters. He is Fellow of IFAC, AAAS, CAA, and NAI.

Hong Jiang (Fellow, IEEE) received the BSc degree in computer engineering from the Huazhong University of Science and Technology, Wuhan, China, the MASc degree in computer engineering from the University of Toronto, Toronto, Canada, and the PhD degree in computer science from the Texas A&M University, College Station, Texas, USA. He is currently Chair and Wendell H. Nedderman Endowed Professor of Computer Science and Engineering Department with the University of Texas at Arlington. Prior to joining UTA, he served as a Program Director with National Science Foundation (2013.1-2015.8) and he was with University of Nebraska-Lincoln since 1991, where he was Willa Cather Professor of Computer Science and Engineering. His present research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, performance evaluation. He is an associate editor of the *IEEE Transactions on Computers* and recently served as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has more than 300 publications in major journals and international Conferences in these areas, including IEEE-TPDS, IEEE-TC, Proceedings of IEEE, ACM-TACO, ACM-ToS, USENIX ATC, FAST, EUROSYS, ISCA, MICRO, SOCC, LISA, SIGMETRICS, ICDE, DATE, ICDCS, IPDPS, MIDDLEWARE, OOPLAS, ECOOP, SC, ICS, HPDC, INFOCOM, ICPP, etc., and his research has been supported by NSF and industry. He is a Member of ACM.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: University of Texas at Arlington. Downloaded on August 24,2023 at 15:51:50 UTC from IEEE Xplore. Restrictions apply.