Characterization and Detection of Artifacts for Error-controlled Lossy Compressors

Pu Jiao,* Sheng Di,† Jinyang Liu,‡ Xin Liang, * Franck Cappello†
*University of Kentucky, Lexington, KY, USA
†University of California, Riverside, CA, USA
†Argonne National Laboratory, Lemont, IL, USA
pujiao@uky.edu, sdi1@anl.gov, jliu447@ucr.edu, xliang@uky.edu, cappello@mcs.anl.gov

***Abstract*—Today's scientific high-performance computing (HPC) applications are often running on large-scale environments, producing extremely large volumes of data that need to be compressed effectively for efficient storage or data transfer. Error-bounded lossy compression is arguably the most efficient way to this end, because it can get very high compression ratios while controlling the data distortion strictly based on user requirements for compression errors. However, error-bounded lossy compressors may have serious artifact issues in situations with relatively large error bound or high compression ratios, which is highly undesirable to users. In this paper, we comprehensively characterize the artifacts for multiple state-of-the-art error-bounded lossy compressors (including SZ-1.4, SZ-2.1, SZ-3.0, FPZIP, ZFP, MGARD) and provide an in-depth analysis for the root cause of these artifacts. We summarize the artifact issue into three types and also develop an efficient artifact detection algorithm for each type of artifact. We finally evaluate our artifact detection methods using four scientific datasets, which demonstrates that the proposed methods are able to detect artifact issues under linear time complexity.**

***Index Terms*—High-performance computing, scientific data, lossy compression, compression artifacts**

I. Introduction

Extreme-scale scientific applications and high-resolution instruments are producing vast amounts of data at an unprecedented speed. For instance, fusion simulation generates over 200 PB of data in a single run [1], which is expected to exceed 1 EB when exascale systems [2], [3] come into place. This creates critical challenges for storing or transferring scientific data, significantly hindering scientific discoveries.

Error-controlled lossy compression [4]–[12] has been regarded as one of the most effective ways to address such big-data challenges, as it reduces the size of scientific data while ensuring certain error control required by the scientists. Despite these error controls, compression artifacts, i.e., visible undesired pattern-based data distortion, may inevitably manifest due to the execution of lossy compression techniques.

The artifact issue of lossy reconstructed data has been widely noticed by scientists in their research work. For instance, Poppick et al. [13], [14] observed that ZFP [9] may have serious block-pattern artifacts when compressing the climate data at certain error bound, which is highly undesired for the post hoc climate data analysis. Another example is seismic imaging analysis [7], [15], which requires the stacking images generated by the lossy reconstructed data barely have visible artifacts to avoid misleading/inaccurate analysis. Moreover, according to a survey about lossy compression use-case [16], visualization is one of the most important lossy compression use-cases, where visible artifacts are undesired.

Characterizing and detecting lossy compression artifacts based on scientific datasets and understanding their causality is critical in many aspects. First, since different compressors may produce distinct artifacts, an in-depth understanding of compression artifacts can help users choose appropriate compressors to avoid the most undesired artifacts. Second, an effective artifact detection algorithm can be used to identify a proper lossy compression error settings (such as error bound), which can effectively avoid the artifact issue. Last but not least, a comprehensive understanding and detection of the artifacts can help the compressor developers further improve the lossy compression quality by developing artifact-aware compressors.

In this paper, we carefully investigate the artifacts produced by the leading scientific lossy data compressors and characterize them into three categories, using the terminology from the image compression community. In particular, we notice that each lossy compressor is dominated by one or two types of artifact, depending on the nature of the corresponding compression algorithm. We further develop several artifact detection algorithms that can effectively detect the identified artifacts based on their characteristics.

Two serious challenges exist for the characterization and detection of lossy compression artifacts. On the one hand, there have been quite a few state-of-the-art error-bounded lossy compressors developed for years, and each of them generally was designed based on largely different principles, so that the reconstructed data may exhibit largely different characteristics, causing a very high diversity in artifacts. On the other hand, accurately detecting artifacts for lossy compressors is very challenging, especially because the reconstructed dataset under the lossy compression always has certain data distortion. As such, it is non-trivial to differentiate the abnormal artifact patterns and the normal data distortion in a lossy reconstructed dataset. We aim to address the two challenges in this paper, and our contributions are summarized as follows.

- We study the decompressed data produced by error-controlled lossy compressors using four scientific datasets

Corresponding author: Xin Liang, Department of Computer Science, University of Kentucky, Lexington KY 40506 USA

2640-0316/23/$31.00 ©2023 IEEE
DOI 10.1109/HiPC58850.2023.00027

from multiple domains. We then characterize the visual artifacts into three categories and analyze their causality, which is the first attempt regarding the error-bounded lossy compressors to the best of our knowledge.
- We propose and implement artifact detection algorithms to detect lossy compression artifacts. In particular, our algorithms are designed for each type of artifact and thus are compressor-free, leading to high flexibility in practice.
- We evaluate our detection methods using the four scientific datasets. Experimental results show that the proposed methods provide effective indicators which detect the identified visual artifacts and indicate the severity.

The rest of the paper is organized as follows. In Section II, we discuss the background and related works. In Section III, we provide an overview with an introduction to the evaluated datasets and compressors. In Section IV, we carefully characterize the artifacts and analyze their causality. In Section V, we describe our methods for artifact detection. In Section VI, we present the evaluation results of the detection methods. In Section VII, we conclude with a vision for future work.

II. Related works

In this section, we review the related works on scientific lossy compressors and artifacts for lossy compression.

A. Scientific lossy compressors

The unprecedented amount and generation speed of scientific data necessitates the need for compression to achieve efficient data storage and transmission. To address the limited compression ratios of lossless compressors and unbounded errors in general lossy compressors, error-controlled lossy compressors have been proposed and developed rapidly in recent years. They are preferred in many use cases [16] and have been applied in multiple applications [17]–[20].

Error-controlled compressors can be divided into prediction-based ones and transform-based ones in general. SZ [4]–[7] is a family of prediction-based compressors. Their compression pipelines usually consist of prediction, quantization, encoding, and lossless compression. FPZIP [8] and ISABELA [12] are two other prediction-based compressors with point-wise error control. ZFP [9] is a typical transform-based compressor featuring fast compression and decompression speed. Recently, MGARD has been proposed to combine wavelet theory and finite element analysis for hierarchical compression. A more detailed introduction to the methodologies of the leading compressors will be covered in the next section.

B. Artifacts of lossy compression

Compression artifacts have been observed and studied in the image compression community for a long time, especially for the widely used JPEG [21] compressor. Many methods have been proposed in the literature to identify and/or remove artifacts in JPEG by leveraging learning-based approaches [22], the correlation of transformed coefficients [23], the quantization matrix [24], and contrast enhancement [25] etc. Nevertheless, most of these works are dedicated to JPEG and thus do not generalize to other compression methods.

Despite recent advances in scientific lossy compressors, there is no systematic study on their compression artifacts. In this work, we fill this gap by providing a comprehensive characterization along with effective artifact detection methods. This will lead to a better understanding of the artifacts in scientific lossy compressors and, in turn, contribute to the identification of proper compression setting as well as the design and development of new compressors.

III. Overview

We aim to understand the artifacts produced by scientific data compressors with two specific goals: (1) we want to know the major artifacts in the leading compressors and their corresponding causality; (2) we seek to effectively detect these artifacts. We rely on comprehensive evaluations using state-of-the-art compressors with real-world scientific datasets below.

A. Datasets

We evaluate four scientific datasets from various domains with different dimensionalities. We include data from the 2D and 3D climate simulations (CESM [26] and Hurricane ISABEL [27], respectively), and 3D cosmological and seismic simulations (NYX [28] and RTM [29], respectively). The detailed information on these datasets is listed in Table I.

TABLE I
Benchmark datasets

Dataset	#Field	Dimension	Size
CESM	77	1800×3600	1.9 GB
Hurricane	13	$100 \times 500 \times 500$	1.3 GB
NYX	6	$512 \times 512 \times 512$	3.0 GB
RTM	3	$1008 \times 1008 \times 352$	4.1 GB

B. Compressors

We evaluate six major error-controlled lossy compressors with different decorrelation methods, including prediction-based ones [5]–[8] and transform-based ones [9], [10]. They can cover 3 different artifacts discussed in this paper. Note that we omit certain variations that build upon those prototypes because they will lead to similar results. For instance, we omit the hybrid compressor in [30] as it is the combination of SZ-2.1 and ZFP; we also omit QoZ [31] which has a very similar design to SZ-3.0 thus exhibits similar artifacts. Although SZ-1.4, SZ-2.1, and SZ-3.0 are different versions of the SZ compressor, they are designed for distinct use cases and have different kinds of visual artifacts. Therefore, they are all used in this paper for different types of artifacts. Their compression procedures are described below.

- **SZ-1.4** [5] predicts data with Lorenzo predictor [32], and then quantizes the floating-point difference into integers with a linear-scaling quantizer. The integers are then fed into a Huffman encoder and a lossless compression algorithm for size reduction.
- **FPZIP** [8] also relies on the Lorenzo predictor for decorrelation, but it enforces point-wise relative error control by a customized encoding strategy after mapping

residuals into integers. In addition, it uses an arithmetic encoder to reduce the size.

- **SZ-2.1** [6] follows the design of SZ-1.4, but improves its quality using adaptive prediction algorithm. In particular, it splits the data into uniform blocks and selects the best-fit prediction algorithm between Lorenzo and regression for each data block based on the input error bound.
- **ZFP** [9] is a transform-based compressors that split data into 4^{n_d} (n_d is the dimensionality) blocks for independent compression. In each block, it performs a near-orthogonal transform after exponent alignment and fixed-point conversion followed by an embedding encoding algorithm.
- **SZ-3.0** [7] further refines the prediction stage in SZ-2.1 with interpolation predictors. It leverages linear or cubic spline interpolations to improve prediction accuracy.
- **MGARD** [10] uses multilinear interpolation and L^2 projection to transform data into multilevel coefficients, and then quantize the data with a linear-scaling quantizer and compress the quantized values using lossless techniques.

IV. Visual Artifact Characterization

In this section, we characterize the visual artifacts produced by the aforementioned compressors. While artifacts are observed in many regions across almost all the fields, we present zoomed-in visualization of one representative field from each dataset for demonstration purposes (CLDHGH from CESM, QVAPOR from Hurricane, Velocity_x from NYX, and Pressure2000 from RTM). We compress these data fields using the six compressors by fixing the compression ratios to an extent when obvious artifacts are observed. The compression statistics, including compression ratios (CR), relative error bounds (EB), peak signal-to-noise ratios (PSNR), and structural similarity index (SSIM), are summarized in Table II.

TABLE II
Compression statistics on sample data fields

Dataset	Compressor	CR	EB*	PSNR	SSIM
CLDHGH	SZ-1.4	32.73	0.005	50.79	0.80
	FPZIP	39.42	0.067	36.91	0.71
	SZ-2.1	68.02	0.010	48.68	0.80
	ZFP	17.93	0.022	50.08	0.83
	SZ-3.0	123.12	0.010	49.20	0.81
	MGARD	121.20	0.030	50.19	0.85
QVAPOR	SZ-1.4	19.99	5E-4	70.73	0.71
	FPZIP	23.34	0.096	38.47	0.97
	SZ-2.1	76.42	0.010	50.36	0.72
	ZFP	26.46	0.006	65.56	0.74
	SZ-3.0	47.62	0.002	62.47	0.70
	MGARD	48.50	0.008	60.48	0.89
Velocity_x	SZ-1.4	57.22	0.005	50.80	0.56
	FPZIP	25.09	0.051	47.92	0.79
	SZ-2.1	942.61	0.090	52.31	0.78
	ZFP	70.36	0.026	58.02	0.88
	SZ-3.0	1091.74	0.01	53.05	0.82
	MGARD	1470.14	0.050	57.94	0.92
Pressure2000	SZ-1.4	75.78	0.002	61.14	0.55
	FPZIP	13.83	0.185	46.29	0.92
	SZ-2.1	676.76	0.05	46.38	0.53
	ZFP	184.49	0.017	61.09	0.61
	SZ-3.0	449.08	0.005	58.53	0.31
	MGARD	496.75	0.032	56.52	0.59

*Maximum relative errors are used in this table for all compressors.

We mainly identified three major types of artifacts, namely posterization, blocking, and interpolation. In the following, we carefully define these artifacts and analyze how they are produced by the error-controlled lossy compressors. The visualization of the sample fields and the zoomed-in regions are illustrated in Fig. 1. In particular, we select the $[250, 310] \times [710, 830]$ window in CLDHGH, $[70, 130] \times [50, 170]$ window in the 64th slice of QVAPOR, $[150, 210] \times [120, 240]$ window in the 100th slice of Velocity_x, $[830, 890] \times [50, 170]$ window in the 581st slice of Pressure2000, as they exhibit visible artifacts under the given compression setting. The results are displayed in Fig. 2. For better visualization quality, these zoomed-in regions are plotted using their local value ranges instead of the global ones.

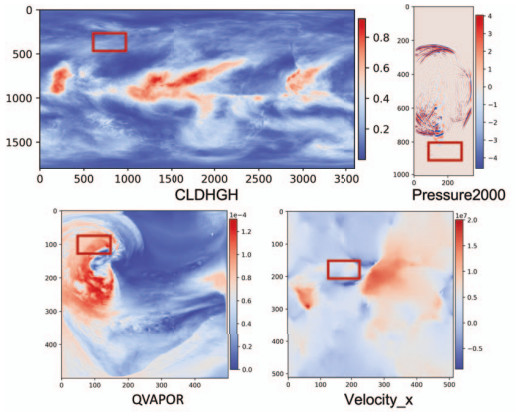

Fig. 1. Visualization of the sample fields and zoomed-in regions.

A. Posterization artifacts

In the image processing community, posterization artifacts, also known as banding, occur when the color depth is not sufficient to accurately sample a continuous gradation of color tone. We borrow this concept and define the posterization artifacts for scientific lossy compressors as follows.

Definition 1. *Posterization artifacts are the banded structures that occur when decompressed data have an obviously smaller number of values than that of the original data.*

Observation 1: SZ-1.4, FPZIP, and ZFP are likely to exhibit posterization artifacts when the error bound becomes large.

As shown in Fig. 2 (b)(c)(e), we have observed posterization artifacts in SZ-1.4, FPZIP, and ZFP. From these figures, we can see clear banded structures in the decompressed data. They are less obvious in ZFP compared with SZ-1.4 and FPZIP, even when ZFP has larger compression ratios (e.g., Velocity_x and Pressure2000).

Causality: Since posterization artifacts indicate low degradation of values, they are mainly caused by the close approximations of data in local regions. We demonstrate how they are produced using SZ-1.4 with the 2D CLDHGH field as an example. For a specific 2D data point at (i, j), SZ-1.4 leverages a Lorenzo predictor for decorrelation, which yields the predicted value $pred(i, j) = d'(i, j-1) + d'(i-1, j) - d'(i-1, j-1)$, where d' represents the values of decompressed data (which

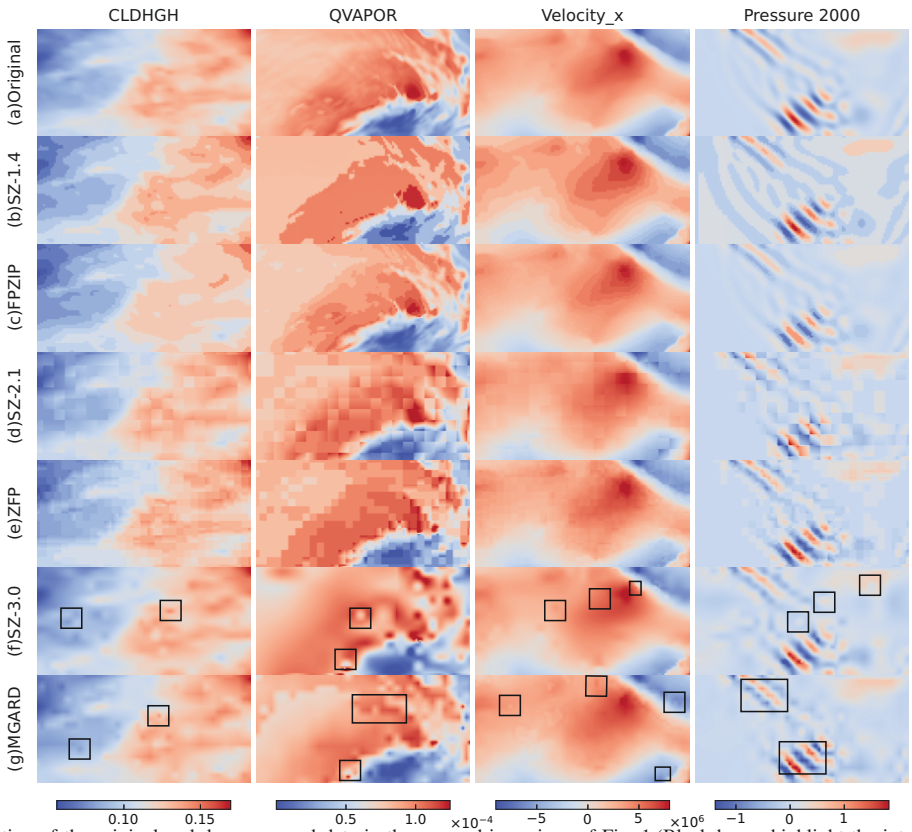

Fig. 2. Visualization of the original and decompressed data in the zoomed-in regions of Fig. 1.(Black boxes highlight the interpolation artifacts.)

are obtained during compression) at the corresponding locations and (i, j) is coordinates of a data point. Then, the linear-scaling quantization computes the quantization index $q(i,j) = (d(i,j) - pred(i,j))/(2 * eb)$, where $d(i,j)$ represents the original data at (i,j) and eb is the error bound, and produces decompressed value $d'(i,j-1) = pred(i,j) + 2 * eb * q(i,j)$. Consider a specific case when $d(i-1,j-1)$, $d(i-1,j)$, $d(i,j-1)$, $d(i,j)$ have very closed values. The former three values may have the same decompressed values, which will yield the same decompressed value for $d(i,j)$ and lead to the same values in the local region. This is validated in Fig. 3, where the same predicted values are observed for each region. While quantization corrects some of the predictions, it is only effective when the prediction error $pred(i,j) - d(i,j)$ exceeds the error bound. This creates the borders of different local regions as depicted in the last sub-figure. This analysis also works for FPZIP as it utilizes the same Lorenzo predictor. For ZFP, posterization artifacts happen because data in a local block are approximated with the same values.

B. Blocking artifacts

Blocking artifacts are very common in JPEG [21] image compression, where visible differences are observed in pixel blocks and at block boundaries. We see similar artifacts in scientific lossy data compressors and define them as follows.

Definition 2. *Blocking artifacts are visible and periodic discontinuities in the decompressed data.*

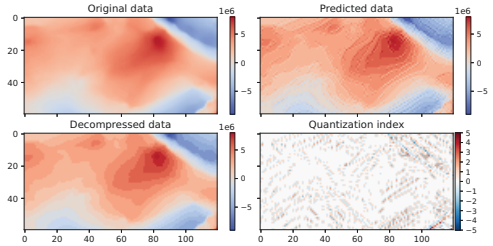

Fig. 3. Velocity_x Original data, SZ1.4 decompressed data, predicted data and quantization index.

Observation 2: Blocking artifacts appear only in compressors with a block-wise design such as SZ-2.1 and ZFP.

We found that blocking artifacts only appear in compressors with a block-wise design, namely SZ-2.1 and ZFP in the evaluated compressors, as shown in Fig. 2 (d)(e). ZFP split data into blocks of 4^{n_d} so the blocking artifacts occur with a period of 4. SZ-2.1 uses a block of 12×12 for 2D data and $6 \times 6 \times 6$ for 3D data, and we can see the blocking artifacts of the corresponding sizes. Note that SZ-2.1 exhibits the "partial blocking effect" as shown in the Velocity_x field of Fig. 2 (b), where blocking artifacts disappear in certain regions of the data. This is because SZ-2.1 employs an adaptive design that selects the better predictor between regression and Lorenzo predictors. As such, data blocks compressed with the latter incur no blocking artifacts.

Causality: Blocking artifacts are usually caused by block-

independent data decorrelation and quantization. We omit the detailed analysis as it is similar to that of JPEG. Both the regression predictor in SZ-2.1 and transform in ZFP are performed in individual blocks, which forms the major causes. This is further validated by the partial blocking effect in SZ-2.1 described above.

C. Interpolation artifact

In addition to posterization and blocking artifacts that are very similar to the artifacts in image compression, we also identify a new type of artifact in scientific lossy compressors. We name them interpolation artifacts as they are most likely caused by the application of interpolation methods during compression, as will be analyzed in the rest of this section. Formally, we define interpolation artifacts as:

Definition 3. *Interpolation artifacts represent the radial patterns in the decompressed data which form either rigid edges or spots.*

Observation 3: Compressors that leverage interpolation methods for decorrelation (e.g., SZ-3.0 and MGARD) generate interpolation artifacts.

Interpolation artifacts are observed mainly in SZ-3.0 and MGARD. As highlighted in Fig. 2 (f)(g), there are multiple distorted regions with visible edges and spots. We can further notice that artifacts in SZ-3.0 usually span wider regions, while those in MGARD are more concentrated in the center. This is because SZ-3.0 usually leverages cubic interpolation that involves more adjacent data points than the multilinear interpolation used in MGARD.

Causality: The interpolation artifacts are mainly caused by the error propagation in the hierarchical interpolations. We analyze the causality using SZ-3.0 as an example. Interpolations in SZ-3.0 are performed in a hierarchical fashion, where a data point in the same level is interpolated using adjacent data with a constant stride. The stride decreases by half when the level decreases by 1, and they both reach 1 at the final level. Fig. 4 depicts how a single error at level 6 propagates to a visible interpolation artifact spanning across a region at level 1. In the beginning, we can observe a large error only in the highlighted point at level 6. As the interpolation procedure continues, the error gradually propagates to a larger region in level 5 and stabilizes in level 4. In the end, it leads to a visible artifact at the final level, which corresponds to the decompressed data.

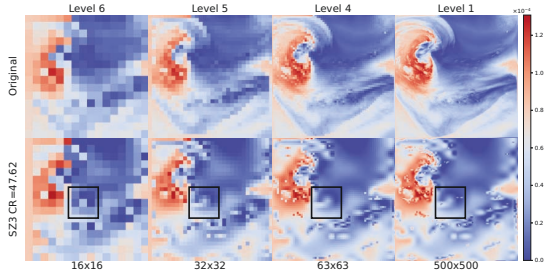

Fig. 4. Forming process of interpolation artifacts in SZ3 (QVAPOR). The number at the bottom indicates the resolution of the level.

V. Artifact Detection

In this section, we present our methods to detect the characterized artifacts in error-controlled scientific lossy compressors. This is important to multiple use cases including performing quality checks for decompressed data and identifying proper compression configuration (e.g., error bounds). We introduce our methods with 2D data for demonstration purposes, but the proposed algorithms easily generalize to cases with 3D or higher dimensional data.

A. Detection of posterization artifacts

A significant observation from the posterization artifact characterization is that a local region can be flushed into the same or close values during compression, resulting in the banded structures in the global view. This feature inspires us to take advantage of connected components [33] to evaluate the degree of posterization, as each banded region can be viewed as a separate connected component. While connected components are employed to detect this artifact in the image compression community [34], [35], most of them are relatively costly and/or only apply to integer data. Instead, we use a simple metric here to achieve high performance while generalizing it to scientific data usually in floating-point formats.

We present our detection method for posterization artifacts in Algorithm 1. The key function `unique_labels` computes the number of connected components of the input data under the given tolerance τ. In the beginning, each data point is initialized with a unique label (lines 1-3). Then, we iterate all the data points one by one and union two neighboring sets if the difference between the two root values are less than τ (lines 5-12). After that, the number of unique labels in the disjoint set is returned, which represents the number of connected components. We perform this algorithm using both decompressed data and original data, and their respective ratio (named *label ratio* in the rest of the paper) is used as an indicator for the severity of posterization artifacts (line 15). Generally speaking, if the *label ratio* (LR) is close to 1, we can conclude that there are almost no posterization artifacts; if the *label ratio* is small, say 0.1, we can anticipate the presence of posterization artifacts because the number of representative values is reduced a lot.

Algorithm 1 Posteriztion artifact detection

Input: decompressed data d', input data d, posterization threshold τ
Output: indicator for posterization artifacts

1: **function** `unique_labels`(d, τ)
2: **for** $(i, j) \in n_1 \times n_2$ **do**
3: $DS[i, j] \leftarrow (i, j)$ /*Init the disjoint set*/
4: **for** $(x, y) \in n_1 \times n_2$ **do**
5: **if** $|d_{i+1,j} - d_{DS[i,j]}| \leq \tau$ **then**
6: $DS[i+1, j] = DS[i, j]$ /*Union with bottom point if necessary*/
7: **if** $|d_{i,j+1} - d_{DS[i,j]}| \leq \tau$ **then**
8: $DS[i, j+1] = DS[i, j]$ /*Union with right point if necessary*/
9: **return** `unique_labels`(d', τ) / `unique_labels`(d, τ)

Fig. 5 illustrates the results of Algorithm 1 on both the original data and the decompressed data produced by SZ-1.4 using the CLDHGH field. According to this figure, it is

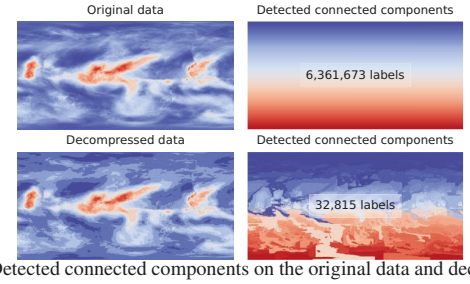

Fig. 5. Detected connected components on the original data and decompressed data of SZ-1.4, (CLDHGH, CR=158.11).

observed that the decompressed data generates a significantly lower ratio compared to that of the original data, which indicates the presence of visible posterization artifacts.

B. Detection of blocking artifacts

The most determining factor of blocking artifacts is the periodic discontinuities characterized in the previous section. Motivated by this fact, we investigate the derivatives that best describe the continuity of functions. Generally speaking, blocking artifacts indicate a rapid change of derivatives on block borders but smooth derivatives inside blocks. This corresponds to large second-order derivatives in the block borders and small ones elsewhere. Fig. 6 depicts the second-order derivatives on a specific row in the original data and the decompressed data of ZFP from the CLDHGH field. It is observed that the second-order derivatives exhibit the expected periodic patterns with a block size 4: there is a large value at one data point (the left border), followed by two small values (inside the block) and another large value (the right border). This inspires us to detect blocking artifacts using the patterns in the second-order derivatives.

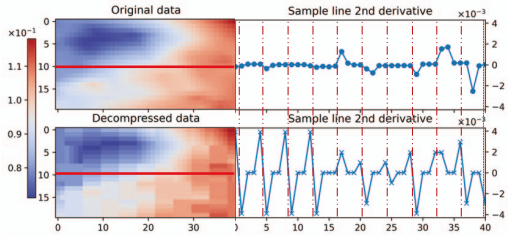

Fig. 6. Blocking artifact detection demonstration

We present our detection algorithm for row-wise blocking artifacts in Algorithm 2, and the same procedure easily generalizes to column-wise blocking artifacts. We first introduce the key function `test_block_size` which computes the number of possible blocks with artifacts and that of effective blocks (i.e., blocks with at least one non-negligible value) on the second-order derivatives. In particular, we initialize the two numbers to 0 (line 2) and increment them while iterating all the blocks of size B (lines 3-11). We validate if a block is effective by comparing its max absolute value with τ_1 (line 5), and if a block has artifacts by comparing the portion of border values over the L_1 norm of the block with τ_2. To this end, the number of artifact blocks and effective blocks is returned. We then detect the most possible block size for blocking artifacts and its indicator using this function. In particular, we sample a certain number of rows from the decompressed data (line 13), and iterate all the candidate rows to find the one with maximal indicator (lines 14-22). The same procedure can be applied to the original data. If a small indicator is returned for the original data while a large one is returned for the decompressed, it is highly possible that the blocking artifacts are introduced by lossy compression.

Algorithm 2 BLOCKING ARTIFACT DETECTION ON ROWS

Input: input data d, effectiveness threshold τ_1, and detection threshold τ_2
Output: the possible block size and an indicator for the severity

1: **function** `test_block_size`(d'', n, B, τ_1, τ_2)
2: $C_{effective} \leftarrow 0$, $C_{artifact} \leftarrow 0$
3: **for** $i = 1 \rightarrow n/B$ **do**
4: $b \leftarrow d''[(i-1) * B : i * B]$ /*Extract data block*/
5: **if** $\|b\|_\infty > \tau_1$ **then**
6: $C_{effective} \leftarrow C_{effective} + 1$ /*Block is effective*/
7: $flag_1 \leftarrow (|b_0| > \tau_2$ **and** $|b_{B-1}| > \tau_2)$ /*Rapid change on borders*/
8: $flag_2 \leftarrow (|b_0| + |b_{B-1}|)/(\sum_i |b_i|) > \tau_3$ /*dominant values on borders*/
9: **if** $flag_1$ **and** $flag_2$ **then**
10: $C_{artifact} \leftarrow C_{artifact} + 1$ /*Block may have artifacts*/
11: **return** $C_{artifact}, C_{effective}$
12:
13: sample_rows $\leftarrow$ `random`(0, n_1)
14: **for** B in candidate block sizes **do**
15: $C_1 \leftarrow 0$, $C_2 \leftarrow 0$
16: **for** $r \in$ sample_rows **do**
17: $C_{1r}, C_{2r} \leftarrow$ `test_block_size`(2nd_order_derivative($d_{r,:}$), n_2, B, τ_1, τ_2) /*Evaluate each sampled row*/
18: $C_1 \leftarrow C_1 + C_{1r}$, $C_2 \leftarrow C_2 + C_{2r}$ /*Aggregate the results*/
19: $P \leftarrow C_1/C_2$
20: **if** $P >$ max_P **then**
21: max_P $\leftarrow P$, block_size $\leftarrow B$
22: **return** block_size, max_P

C. Detection of interpolation artifacts

As characterized in the last section, interpolation artifacts exhibit two properties: 1) it is centered on a data point that is usually local maximum or minimum; 2) it has a radial pattern that radiates from the center point to the local regions. We first define a new terminology "relative sign", which can be used to describe the two properties. Specifically, the relative sign of a data point d_1 toward another data point d_2 is $+1$ if $d_1 > d_2$ and -1 otherwise. Then the summed relative sign of a data point is defined as the summation of the relative signs toward all its adjacent neighbors. Using this definition, local maximums and minimums will always have summed relative signs of $+4$ and -4, respectively, in the 2D case. For the radial patterns, they will start at a local maximum or minimum with an absolute summed relative sign 4 and propagate along the two axes. Assuming the centering point is a local maximum. Any data points located on the propagation paths will be smaller than the data point from which it is propagated but larger than all the data points in the other three directions, yielding a summed relative sign of $-1 + 3 = 2$. Similarly, any other points in the rectangular region will have a summed relative sign of 0. One example is illustrated in Fig. 7, where one can see clear patterns of summed relative signs in the decompressed data but no patterns in the original data.

We summarize our detection method for interpolation artifacts in Algorithm 3, with a function `relative_sign`

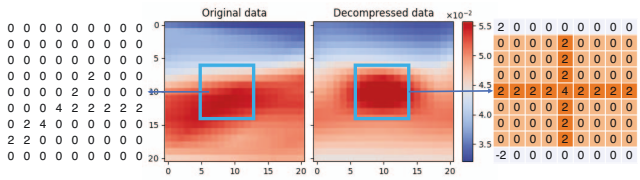

Fig. 7. Demonstration of the summed relative signs in a region with interpolation artifacts produced by SZ-3.0 (CLDHGH).

representing the computation of the relative sign between two data points. The function `sum_of_relative_sign` (line 1-3) computes the summed relative sign of a data point at index (i, j), and the function `compute_padding` computes a boolean flag indicating the presence of artifacts along with the length of the artifact along x and y axes based on the pattern mentioned above. For the example in Fig. 7, it will yield $flag = true$, $l_x = 3$, and $l_y = 4$ if the value range of the region exceeds the prescribed threshold τ. Our main detection function starts at line 19, where we initialize an empty set to collect detected artifacts. Then we compute the summed relative signs for each point in the decompressed data (lines 20-21). After that, we iterate all the decompressed data and check if it leads to a possible artifact (lines 23-32). Specifically, we first check if the first property (local maximum or minimum) is satisfied (line 24), and then compute the boolean indicator and corresponding padding for the second property (radial pattern) (line 25). If the current decompressed data exhibits a pattern that is similar to the artifact, we check if such a pattern exists in the original data (line 28-29). If not, we mark it as an artifact and append it to our result (lines 29-31). Note that we will omit a potential artifact if the value range of the region is less than a prescribed threshold τ (lines 26-27), which indicates such an artifact is not visible under the given threshold.

Algorithm 3 INTERPOLATION ARTIFACT DETECTION

Input: original data d and decompressed data d', range threshold τ
Output: Artifact location index and its length along two axes

1: **function** `sum_of_relative_sign`(d, i, j)
2: $sum \leftarrow$ `relative_sign`$(d_{i,j},\ d_{i-1,j})$ + `relative_sign`$(d_{i,j},\ d_{i+1,j})$ + `relative_sign`$(d_{i,j},\ d_{i,j-1})$ + `relative_sign`$(d_{i,j},\ d_{i,j+1})$ /*Sum the relative signs of four directions in 2D*/
3: **return** sum
4:
5: **function** `compute_padding`(i, j, s)
6: $l_x \leftarrow 1, l_y \leftarrow 1$
7: **while** $|s_{i,j+l_y}| == |s_{i,j-l_y}| == 2$ **and** $s_{i,j+l_y} * s_{i,j} > 0$ **do**
8: $l_y \leftarrow l_y + 1$ /*Increment length along y if condition holds*/
9: **while** $|s_{i+l_x,j}| == |s_{i+l_x,j}| == 2$ **and** $s_{i+l_x,j} * s_{i,j} > 0$ **do**
10: $l_x \leftarrow l_x + 1$ /*Increment length along x if condition holds*/
11: $l'_x = l_x - 1, l'_y = l_y - 1$
12: **if** $l'_x > 0$ **and** $l'_y > 0$ **then**
13: **for** $(m, n) \in \{(m, n) | -(l_x - 1) \leq m \leq (l_x - 1), -(l_y - 1) \leq n \leq -(l_y - 1), m \neq 0, n \neq 0\}$ **do**
14: **if** $s_{i+m,j+n} \neq 0$ **then**
15: $l'_x = \min(|m| - 1, l'_x)$, $l'_y = \min(|n| - 1, l'_y)$
16: flag = $(l'_x > 0)$ **and** $(l'_y > 0)$ /*Flag artifact if found*/
17: **return** flag, l'_x, l'_y
18:
19: artifacts $\leftarrow \{\}$
20: **for** $(i, j) \in n_1 \times n_2$ **do**
21: decompressed_sign$[i, j] \leftarrow$ `sum_of_relative_sign`(d', i, j)
22: **for** $(i, j) \in n_1 \times n_2$ **do**
23: **if** | decompressed_sign$[i, j]$ | == 4 **then**
24: d_flag, l''_x, $l''_y \leftarrow$ `compute_padding`$(i, j,$ decompressed_sign$)$
25: **if** `range`$(d_{[i-l''_x:i+l''_x, j-l''_y:j+l''_y]}) < \tau$ **then**
26: **continue**
27: **if** d_flag **then**
28: original_sign = `sum_of_relative_sign`(d, i, j)
29: flag = (original_sign≠decompressed_sign$[i, j]$)
30: **if** flag==**true** **then**
31: artifacts.append$(\{i, j, l''_x, l''_y\})$
32: **return** artifacts

VI. Evaluation

In this section, we present our evaluation results on artifact detection with the compressors and datasets used in the previous sections. We normalize all the data to the range $[0, 1]$ for a unified setting of the necessary thresholds. All of the experiments are performed on a node of a high-performance cluster MCC [36], where each node is equipped with two 2 AMD EPYC 7702 processors and 512GB DDR4 memory. For all three kinds of artifacts, detection is performed on all the four fields and CLDHGH is used to provide further visualization demonstration.

A. Detection of posterization artifacts

Given the original data, we can evaluate the posterization ratio on both the original data and the decompressed data with the same posterization threshold. The posterization threshold τ used for the following experiments is 1E-5. The experiment results are shown in Fig. 8. In the results, we use the *Normalized Label Ratio* (NLR), which is obtained from dividing the label ratio of decompressed data by the label ratio of the original data. If NLR is close to 1, we have a similar amount of connected components; if NLR is larger than 1, we have extra connected components, which could be caused by the rigid edges introduced by interpolation or block-wise compression; and if NLR is less than 1, we lose some connected components due to the compression, thus causing posterization artifacts.

For each test case, we choose the bitrate range where the NLR of SZ-1.4 spans from about 0.1 to the label ratio of the original dataset. Other compressors are tuned to this bitrate range for fair comparison. We can see that the NLR of SZ-1.4 stably increases as we decrease the compression ratio. The decompressed data of ZFP and SZ-2.1 first suffer from posterization artifact when the compression ratio is large and then may introduce extra edges because of the blocking artifact on the decompressed data. MGARD and SZ-3.0 results show that they introduce extra labels when the compression ratio is large and their NLR decrease as we lower the compression ratio, compared to the original data. This corresponds with the observation of the interpolation artifact: isolated and abrupt change in a region that is smooth on the original data.

Fig. 9 and 10 show the association between the evaluation metrics and the visualization on the original data and the decompressed data from SZ-1.4 and FPZIP. As the visualization deteriorates, we see a decline in the NLR,SSIM and PSNR. SSIM is a perceptual index, but it is not as sensitive as NLR

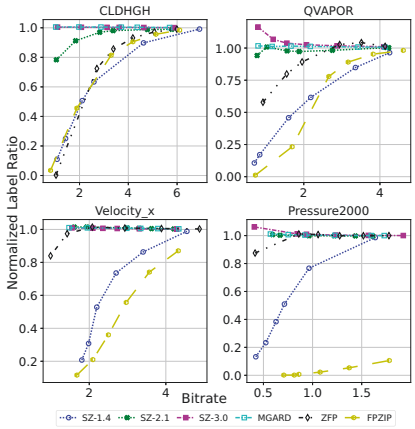

Fig. 8. Detection results of posterization artifacts.

in terms of posterization artifacts in the windowed region.

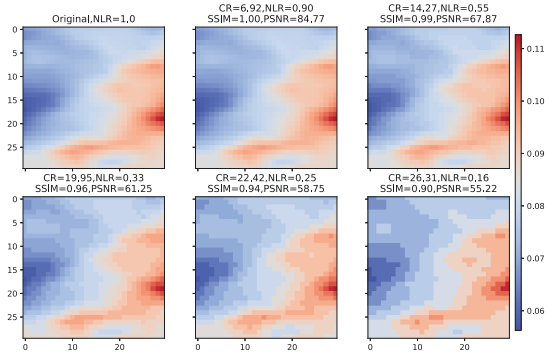

Fig. 9. Visualization of different label ratios (LR) in SZ-1.4 on CLDHGH

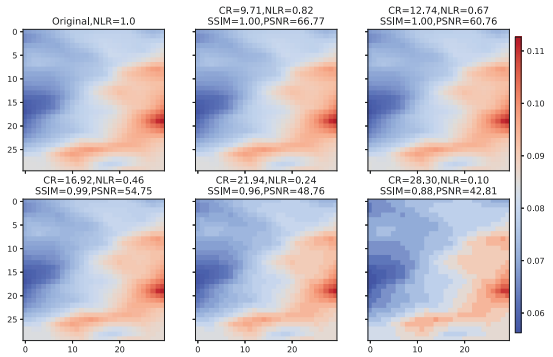

Fig. 10. Visualization of different label ratios (LR) in FPZIP on CLDHGH.

B. Detection of blocking artifacts

In this section, we use prior knowledge that ZFP works on separated blocks of block size of 4 and set the block size of SZ-2.1 to 6 for the experiments. To better assist the readers, we use the compression configurations shown in Table. II so readers can find the visualization in Fig.2.

We try the decompressed data with block sizes from 4 to 16, and the detection results are shown in Fig. 11. When we look at the block size of 4, ZFP has the highest block ratio (BR) among all the compressors for 4 different fields. Similarly, when we look at the block size of 6, we find that SZ-2.1 has the highest BR. If we detect the ZFP decompressed data with different block sizes, we find that a block size of 4 has the highest BR. This is the same for SZ-2.1. We demonstrate that our algorithm is sensitive to compressors that can cause block artifacts, and it can be used to detect the possible block size of the blocking artifact.

Next, we show the relation among the compression ratio(CR), block ratio(BR), and visual quality, shown in Fig. 11. CLDHGH is compressed with ZFP and SZ-2.1. We show the BR of block size of 4 for ZFP results and 6 for SZ-2.1. In the visualization, all the figures use the value range of the windowed value range of the original data.

From the results of ZFP instances, we can see the BR first increases with the CR and then decreases. From the visual quality, when we have a high BR, we are able to see blocking artifacts in the current window. When we get a lower BR, either the decompressed data is close to the original data or we have severe posterization artifacts in the decompressed data. We have the same observation on the results of SZ-2.1. In practice, we can use block detection with posterization detection together to make the decision on the artifact present in the decompressed data.

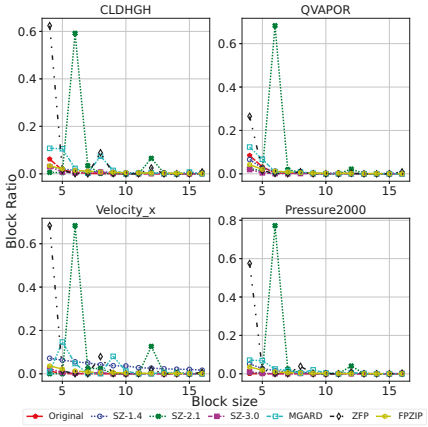

Fig. 11. Blocking detection on the four fields

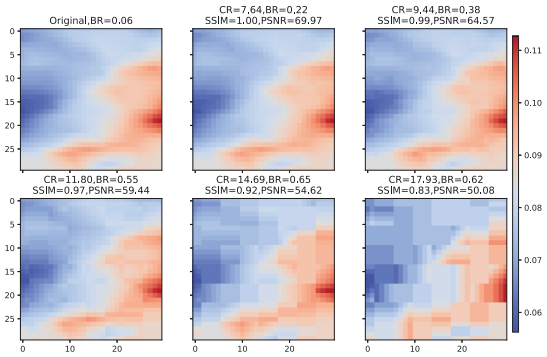

Fig. 12. Visualization of different blocking ratios (BR) in ZFP on CLDHGH

C. Detection of interpolation artifacts

The results of the interpolation give the number of locations that match the interpolation artifact pattern defined in Fig. 7. This information can be used as a reference by the users to examine the specific locations for further evaluation. We

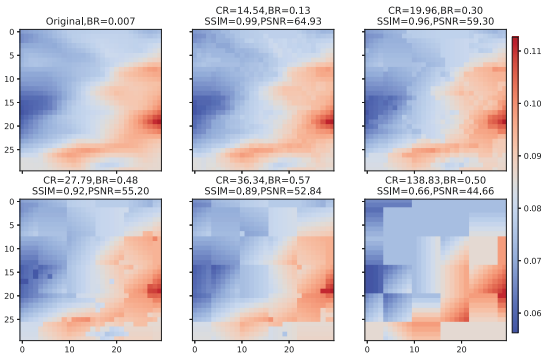

Fig. 13. Visualization of blocking detection in SZ-2.1 on CLDHGH

use the number of detected artifact locations for interpolation artifacts, denoted as "count" in Fig. 14, 15, and 16.

As Fig. 14 shows, our method can detect the interpolation artifacts defined in Fig. 7 in the MGARD and SZ-3.0 decompressed data. As bitrate decreases, we find more interpolation artifacts from MGARD and SZ-3.0 decompressed data. We also notice that our method gives a small amount false positive interpolation artifacts on the decompressed data from other compressors. This is another reason that the results of interpolation detection should only be used for visual defect reference rather than ground truth.

Next, we check the association among the compression ratio, interpolation artifact count, and artifact visualization in Fig. 15 and 16. As the compression ratio increases, we can see more and more interpolation artifacts on the decompressed data from the visualization, and we also see an increase in the interpolation artifact location count on both MGARD and SZ-3.0 instances.

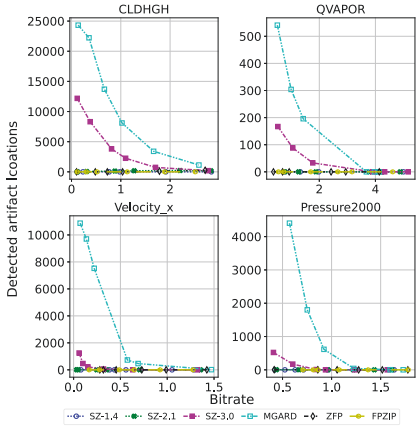

Fig. 14. Interpolation detection on the four fields

D. Performance

We evaluate the performance of the proposed algorithms on all fields of the four datasets, using the decompressed data of SZ-3.0 as an example. The results are presented in Table III, where the compression speed (S_C) and decompression speed (S_D) of SZ-3.0 are reported for reference. All the statistics provided in the table are the aggregated results for each dataset. Each experiment is run 3 times to get the average execution time.

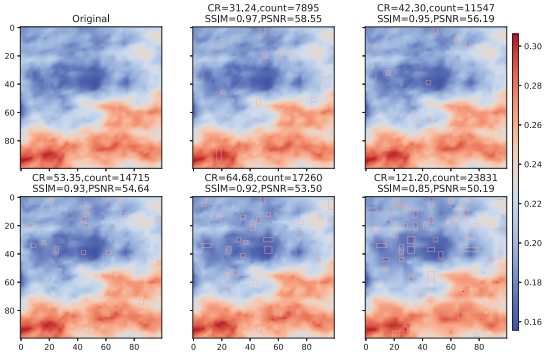

Fig. 15. MGARD interpolation artifact detection visualization

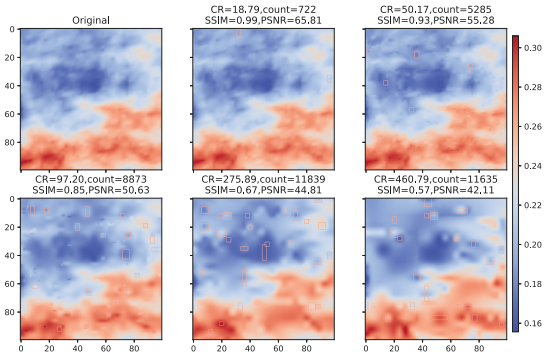

Fig. 16. SZ-3.0 interpolation artifact detection visualization

Among the three detection methods, blocking detection is the most efficient because it works on sampled data instead of the entire field. Posterization detection requires the construction of the segmentation map for original data and decompressed data. Interpolation detection requires the construction of the "relative sign" map on the decompressed data. Both posterization and interpolation detection are working on the entire field, which makes them slower than blocking detection.

TABLE III
ATIFACT DETECTION SPEED (MB/S)

Dataset	CR	S_C	S_D	Blocking	Posterization	Interpolation
CESM	57.67	158.0	559.55	565.93	232.35	297.99
Hurricane	59.41	175.91	582.92	1641.04	131.19	165.47
NYX	173.12	92.66	50.87	255.94	113.85	160.57
RTM	140.6	171.69	465.03	1511.41	127.01	154.48

VII. Conclusion and Future Work

In this paper, we characterize the artifact of different compressors and propose methods to detect these artifacts. Specifically, we have categorized the artifacts of state-of-the-art error-bounded lossy compressors into three kinds and carefully analyzed the causality. Our detection methods can successfully identify these artifacts given the decompression data and indicate the severity. In the future, we will study how to design efficient compression algorithms to mitigate artifacts.

Acknowledgment

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for

the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709, OAC-2104023, and OAC-2330367. We acknowledge the computing resources provided by the Center for Computational Science of the University of Kentucky.

REFERENCES

[1] "Team at princeton plasma physics laboratory employs doe supercomputers to understand heat-load width requirements of future iter device," https://www.olcf.ornl.gov/2021/02/18/scientists-use-supercomputers-to-study-reliable-fusion-reactor-design-operation, 2021, online.

[2] "Aurora exscale system," https://www.alcf.anl.gov/support-center/aurora.

[3] "Frontier exscale supercomputer," https://www.olcf.ornl.gov/frontier.

[4] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium*. Chicago, IL, USA: IEEE, 2016, pp. 730–739.

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

[6] 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 *2018 IEEE International Conference on Big Data*. IEEE, 2018, pp. 438–447.

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

[8] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[9] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[10] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

[11] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "Tthresh: Tensor compression for multidimensional visual data," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 9, pp. 2891–2903, 2019.

[12] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Isabela for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 524–540, 2013.

[13] A. Poppick, J. Nardi, N. Feldman, A. Baker, and D. Hammerling, "A statistical analysis of compressed climate model data," *Proc. DRBSD*, pp. 1–6, 2018.

[14] A. Poppick, J. Nardi, N. Feldman, A. H. Baker, A. Pinard, and D. M. Hammerling, "A statistical analysis of lossily compressed climate model data," *Computers & Geosciences*, vol. 145, p. 104599, 2020.

[15] Y. Huang, K. Zhao, S. Di, G. Li, M. Dmitriev, T.-L. D. Tonellot, and F. Cappello, "Towards improving reverse time migration performance by high-speed lossy compression," in *23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (AMC CCGrid2023)*, 2023.

[16] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong, "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.

[17] 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 *2018 IEEE International Conference on Cluster Computing*. IEEE, 2018, pp. 1–11.

[18] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne, "Toward a multi-method approach: Lossy data compression for climate simulation data," in *International Conference on High Performance Computing*. Springer, 2017, pp. 30–42.

[19] E. Agullo, F. Cappello, S. Di, L. Giraud, X. Liang, and N. Schenkels, "Exploring variable accuracy storage through lossy compression techniques in numerical linear algebra: a first application to flexible gmres," Ph.D. dissertation, Inria Bordeaux Sud-Ouest, 2020.

[20] Q. Gong, X. Liang, B. Whitney, J. Y. Choi, J. Chen, L. Wan, S. Ethier, S.-H. Ku, R. M. Churchill, C.-S. Chang, M. Ainsworth, O. Tugluk, T. Munson, D. Pugmire, R. Archibald, and S. Klasky, "Maintaining trust in reduction: preserving the accuracy of quantities of interest for lossy compression," in *Smoky Mountains Computational Sciences and Engineering Conference*. Springer, 2021.

[21] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[22] Y. Li, F. Guo, R. T. Tan, and M. S. Brown, "A contrast enhancement framework with jpeg artifacts suppression," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. Springer, 2014, pp. 174–188.

[23] D. Bhardwaj and V. Pankajakshan, "A jpeg blocking artifact detector for image forensics," *Signal Processing: Image Communication*, vol. 68, pp. 155–161, 2018.

[24] M. Ehrlich, L. Davis, S.-N. Lim, and A. Shrivastava, "Quantization guided jpeg artifact correction," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*. Springer, 2020, pp. 293–309.

[25] X. Fu, Z.-J. Zha, F. Wu, X. Ding, and J. Paisley, "Jpeg artifacts reduction via deep convolutional sparse coding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2501–2510.

[26] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay *et al.*, "The community earth system model: a framework for collaborative research," *Bulletin of the American Meteorological Society*, vol. 94, no. 9, pp. 1339–1360, 2013.

[27] H. I. dataset, http://sciviscontest-staging.ieeevis.org/2004/data.html, online.

[28] "Nyx: An adaptive mesh, cosmological hydrodynamics simulation code," https://amrex-astro.github.io/Nyx/, 2021, online.

[29] S. N. Kayum, T. Tonellot, V. Etienne, A. Momin, G. Sindi, M. Dmitriev, and H. Salim, "Geodrive-a high performance computing flexible platform for seismic applications," *First Break*, vol. 38, no. 2, pp. 97–100, 2020.

[30] X. Liang, S. Di, S. Li, D. Tao, B. Nicolae, Z. Chen, and F. Cappello, "Significantly improving lossy compression quality based on an optimized hybrid prediction model," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–26.

[31] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Dynamic quality metric oriented error bounded lossy compression for scientific datasets," in *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 2022, pp. 892–906.

[32] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 343–348.

[33] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao, "The connected-component labeling problem: A review of state-of-the-art algorithms," *Pattern Recognition*, vol. 70, pp. 25–43, 2017.

[34] Y. Wang, S.-U. Kum, C. Chen, and A. Kokaram, "A perceptual visibility metric for banding artifacts," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 2067–2071.

[35] S. Bhagavathy, J. Llach, and J. fu Zhai, "Multi-scale probabilistic dithering for suppressing banding artifacts in digital images," in *2007 IEEE International Conference on Image Processing*, vol. 4. IEEE, 2007, pp. IV–397.

[36] "Morgan Compute Cluster," https://www.ccs.uky.edu, 2023.