Optimizing Error-Bounded Lossy Compression for Scientific Data with Diverse Constraints

Yuanjian Liu, Sheng Di*, *Senior Member, IEEE*, Kai Zhao, Sian Jin, Cheng Wang, Kyle Chard, Dingwen Tao, Ian Foster, *Fellow, IEEE*, Franck Cappello, *Fellow, IEEE*

Abstract—Vast volumes of data are produced by today's scientific simulations and advanced instruments. These data cannot be stored and transferred efficiently because of limited I/O bandwidth, network speed, and storage capacity. Error-bounded lossy compression can be an effective method for addressing these issues: not only can it significantly reduce data size, but it can also control the data distortion based on user-defined error bounds. In practice, many scientific applications have specific requirements or constraints for lossy compression, in order to guarantee that the reconstructed data are valid for post hoc analysis. For example, some datasets contain irrelevant data that should be isolated in particular and users often have intuition regarding value ranges, geospatial regions, and other data subsets that are crucial for subsequent analysis. Existing state-of-the-art error-bounded lossy compressors, however, do not consider these constraints during compression, resulting in inferior compression ratios with respect to user's post hoc analysis, due to the fact that the data itself provides little or no value for post hoc analysis. In this work we address this issue by proposing an optimized framework that can preserve diverse constraints during the error-bounded lossy compression, e.g., cleaning the irrelevant data, efficiently preserving different precision for multiple value intervals, and allowing users to set diverse precision over both regular and irregular regions. We perform our evaluation on a supercomputer with up to 2,100 cores. Experiments with six real-world applications show that our proposed diverse constraints based error-bounded lossy compressor can obtain a higher visual quality or data fidelity on reconstructed data with the same or even higher compression ratios compared with the traditional state-of-the-art compressor SZ. Our experiments also demonstrate very good scalability in compression performance compared with the I/O throughput of the parallel file system.

Index Terms—Big Data, Error-Bounded Lossy Compression, Data Reduction, Large-Scale Scientific Simulation

1 Introduction

Modern scientific simulations can produce extraordinary volumes of data. For example, climate and weather simulations [1] can produce terabytes of data in a matter of seconds [2], and the Hardware/Hybrid Accelerated Cosmology (HACC) simulation code [3] can generate petabytes of data from a single run. The resulting data must be stored for subsequent use; however, the cost and availability of storage often lead to difficult decisions regarding future utility. Further, there is a growing need to transfer and share simulation data over wide area networks (e.g., via Globus [4]), which may have lower bandwidth.

Error-bounded lossy compressors [5], [6], [7], [8], [9], [10], [11] are widely used to reduce scientific data volumes while meeting user requirements for data fidelity. For instance, the SZ [6], [12], ZFP [5], and MGARD [13] compressors each allow users to request that the difference between original and reconstructed data be bounded by a specified absolute error bound (i.e., a threshold) when performing lossy compression. Climate research scientists have verified that the reconstructed data generated by error-bounded lossy compressors are acceptable for post hoc analysis [14], [2], [15]. Similarly, adopting a customized error-bounded lossy compressor has been shown to reduce the memory capacity required for general quantum circuit simulations [8].

Existing error-bounded lossy compressors have a significant limitation, however: none support preserving specific constraints, such as isolating irrelevant values, preserving value ranges, or preserving different precisions for different value intervals in the dataset or different regions in the space. For example, in environmental science, different values in a dataset commonly have different significance to post hoc analysis. Thus, users hope to set different precisions (or error bounds) based on various value intervals in the dataset. A typical example is tracing a hurricane's moving trajectory over the sea: only the data points whose water surface values are greater than a threshold (such as 1 meter) are interesting to environment scientists. The Nyx cosmological simulation [16] presents another good example as scientists performing post hoc analysis focus on a specific quantity of interest (e.g., dark matter halo cell information). According to the dark matter halo analysis algorithm, the construction of dark matter halos is determined primarily by the values of two fields (dark matter density and baryon density) in a specific value interval of [81,83]. Therefore, in order to preserve the features (such as the count and location) of the dark matter halos, the

* *Sheng Di (sdi@anl.gov) is the corresponding author.*

- *Yuanjian Liu and Kyle Chard are with the Department of Computer Science at the University of Chicago, Chicago, IL 60601, USA.*
- *Sheng Di, Cheng Wang, and Franck Cappello are with the Mathematics and Computer Science Division at Argonne National Laboratory, Lemont, IL 60439, USA.*
- *Ian Foster is with both Argonne National Laboratory and the University of Chicago, Chicago, IL 60601, USA.*
- *Kai Zhao is with the Department of Computer Science and Engineering at the University of California, Riverside, Riverside, CA 92521, USA.*
- *Sian Jin and Dingwen Tao are with the School of EECS at Washington State University, Pullman, WA 99164, USA.*

values in this interval should have higher precision than the values in other intervals. Some other datasets such as the Community Earth System Model (CESM) [1] map the geolocations into the data location, and it is important to choose certain regions for investigation. When studying the impact of weather in the United States, for example, scientists may care more (or only) about the areas within or near U.S. boundaries.

In this paper we propose a novel compression method based on the SZ error-bounded lossy compression framework,[1] which allows users to specify constraints, such as setting different error bounds in various value intervals or spatial regions, so that the reconstructed data can meet users' required quality better than traditional uniform error-bounded lossy compression can. In particular, our constraint-based compression model addresses irrelevant data in scientific datasets and effectively preserves the global value ranges, which are critical to obtaining a high compression quality in some cases.

We summarize the key contributions as follows.

- We propose a constraint-based error-bounded lossy compression model; to the best of our knowledge, this is the first attempt to develop such a model. The user-specified constraints include (A) isolating irrelevant values, (B) preserving global value range, (C) preserving multi-interval-based error bounds, (D) preserving multiregion-based error bounds, and (E) using a bitmap to mask complicated regions and apply different error bounds on each region. These constraints are critical to post hoc analysis of different applications in practice.
- We develop a series of optimization strategies for preserving constraints efficiently. Specifically, we redesign the quantization stage in the SZ error-bounded lossy compression framework.
- We perform a comprehensive evaluation using multiple real-world scientific datasets across different domains. Experiments show that our solution can respect users' constraints, while maintaining a high compression ratio. Specifically, our solution can obtain better visual quality or data fidelity in the lossy-reconstructed data for different applications, with the same compression ratios compared with the single error-bounded compressor. Our experiments also demonstrate a good scalability in compression time compared with the parallel file system's I/O cost.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we first introduce the SZ compression model and then discuss the five scientific constraints posed by scientists across different domains. In Section 4 we formulate the research problem based on the SZ error-bounded lossy compression model. In Section 5 we propose a battery of efficient algorithms to preserve the user-required constraints and also optimize the compression quality and performance for different cases. In Section 6 we present our evaluation results. In Section 7 we summarize our findings and conclude with a vision of future work.

1. We adopt the SZ compression framework because it provides leading error-bounded lossy compression quality, as verified by several studies of different scientific datasets [6], [12], [17], [18].

2 Related Work

Data compression is used widely in scientific research, for example to reduce data storage and transfer size and costs. Data compressors are typically split into two classes: lossless compression [19], [20], [21], [22] and lossy compression [6], [12], [5], [23], [24]. The former introduces no data loss during compression, but it suffers from very low compression ratios (generally 1.1-2 [25], [26]). The latter can achieve very high compression ratios (such as 100+) [6], [12], [5], [18], but potential data loss may distort analysis results.

To address the concern about data loss, researchers have studied error-bounded lossy compressors for scientific data, which can be split into two major categories – prediction-based compression model and transform-based compression model. SZ [6], [12], [18] is a typical prediction-based lossy compression model, which is composed of four key stages: data prediction, linear-scale quantization, Huffman encoding and lossless compression. ZFP [5] is a typical compressor designed based on the transform-based model, which includes four key steps: splitting dataset into fixed-size blocks, exponent alignment in each block, orthogonal data transform for each block; and embedded encoding for each block.

Existing error-bounded lossy compressors offer different types of error bounds to address diverse user demands. The most common error-bounding approach involves using an absolute error bound, which ensures that the pointwise difference between the original raw data and reconstructed data is confined within a constant threshold. Many compressors such as SZ [6], [12], [18], ZFP [5], [27], and MGARD [13] support absolute error bounds. Other error-bounding approaches have been explored to adapt to diverse user requirements. For instance, SZ supports pointwise relative error bounds [28], [29]; and Digit Rounding [30], Bit Grooming [24], zfp [5], and FPZIP [23] support a precision mode that allows users to specify the number of bits to be truncated in the end of the mantissa, in order to control the data distortion at different levels.

To satisfy user demands on a specific quality of interest, researchers have recently studied how to respect some specific metrics. For instance, Tao et al. [31] developed a formula that can link the target peak signal-to-noise ratio (PSNR) metric to an absolute error bound setting in SZ such that data can be compressed based on a user-specified PSNR metric. MGARD [13] supports various norm error metrics and linear quantities of interest in its multigrid compression method. However, none of the existing error-bounded lossy compressors allow users to set particular constraints in the error-bounded compression and hence impose a significant impediment on the practical use of such compressors. In fact, users often have diverse precision demands for various data value intervals or specific requirements on different spatial regions, which are determined by their sophisticated post hoc analysis purposes and quantities or features of interest.

3 Research Background

We describe the research background in this section, including SZ compression model and diverse constraints in scientific datasets.

3.1 SZ Compression Model

The error-bounded lossy compression model SZ is illustrated in Figure 1. As shown in the figure, the compression model is composed of four key stages: prediction, quantization, Huffman encoding, and lossless compression. Given a set of raw data, SZ scans the whole dataset (either pointwise [6], [12] or blockwise [18]) to predict the data values. In a 1D dataset, the prediction method is simply a first-order Lorenzo predictor [12], which uses only the preceding value to approximate the current data point. In a 2D or 3D dataset, SZ adopts a hybrid data prediction method combining the first-order Lorenzo predictor (using three nearby values in the 2D Lorenzo and 7 nearby values in the 3D Lorenzo) and a linear-regression-based predictor [18]. Such a hybrid predictor can significantly improve the data prediction accuracy, which in turn can substantially increase the compression ratio, especially when the error bound is relatively large. The second stage in SZ uses a linear quantization method to convert the distance between the predicted value and original value to an integer number (called the quantization code or quantization number) for each data point. A customized Huffman encoder is then applied to compress the integer quantization codes, followed by a lossless dictionary coding (using Zstd [19] by default in SZ).

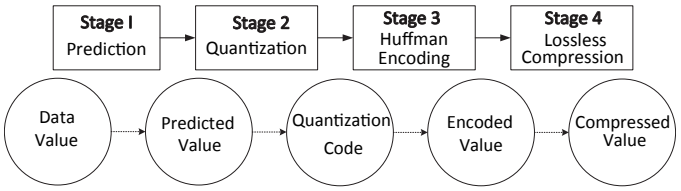


Fig. 1. General procedure of constraint preserving error bounded lossy compression: Constraint (A) is handled before the prediction step; constraint (B) is handled primarily in both the prediction and quantization stage by replacing data points with Lorenzo-predicted values; constraints (C), (D), and (E) are addressed by designing a new quantization method.

3.2 Diverse Constraints in Scientific Datasets

In this paper we propose a novel concept in the practical use of error-bounded lossy compression—preserving diverse constraints specified by users.

A constraint here is referred to as a particular condition that must be applied during the error bounded lossy compression. We describe five types of constraints that are commonly required in real-world science applications (see Table 1)

Irrelevant (or missing) data. Scienfitic datasets are often sparse, and missing data are often encoded in esoteric manners. Specifically, we observe that some datasets (particularly those generated by climate and weather simulations) often contain extremely large values (such as 1E35) that are far from the normal value range. These values are used to indicate "missing" values or background information (such as coastline locations). Those data points need to be recorded in the dataset for the purpose of post hoc analysis. However, these data affect data smoothness in space, which may substantially reduce data transform efficiency or prediction accuracy, significantly degrading the lossy compression quality.

Global value range. In some scientific datasets values outside a "normal" range may result in serious errors for post hoc analysis. For instance, the temperature of liquid water at one standard atmospheric pressure has a meaningful value range, which is $0°C \sim 100°C$. Any values outside this range would cause incorrect post hoc analysis. For the existing error-bounded lossy compressors, however, the reconstructed data could fall outside of the meaningful value range. For example, if the error bound is 5°C, some of the decompressed data values may reach up to 105°C or down to -5°C, which is undesirable for water temperature.

Interval-based error bound. In practice, post hoc analyses often focus on specific value intervals within the whole dataset. Thus, researchers may want to apply different error bounds (or precisions) based on value intervals. For instance, environmental scientists track the location of Hurricane Katrina [33] by calculating the height of the water surface (overly high water surface values indicate the location of the hurricane at that moment). Accordingly, the researchers care only about the data whose values are greater than a threshold, such as 1 meter, in the simulation. On the other hand, the decompressed data are supposed to be confined within the original interval. In the Hurricane Katrina simulation, for example, if the water surface threshold is set to 1 meter, all the reconstructed data points whose values are greater than 1 from the previously lower-than-1 raw values would be considered "false alarms," which is undesired by users.

Region-based error bound. Different regions in a scientific dataset may have different importance according to its physical meaning. For example, CESM [1] records the climate change globally, and its data indexes refer to geolocations. Researchers making use of a specific scientific dataset generally understand which spatial regions need to be studied. Thus, it is possible to set different error bounds across different regions of the datasets so that specific regions of interest can be kept at a high resolution to achieve necessary data fidelity, while other regions can be less precise to obtain a high compression ratio.

More complex error bounds. While the above constraints cover most real-world error bound requirements, some applications have more complex and fine-grained demands. For instance, geolocation-related datasets such as those in CESM [1] may have sophisticated contours around lands and oceans, and scientists may wish to have higher precision in land areas. In this case we allow users to mark a customized 2D or 3D area and use a bitmap array to specify different error bounds for every data point. We can also use the bitmap to automate some region selection for users based on the data patterns. By applying advanced bitmap generation algorithm, our solution can preserve customized diverse precisions for a dataset.

4 Problem Formulation

In this section we formulate our diverse constraint error-bounded lossy compression problem.

Given a scientific dataset D composed of N floating-point values (either single precision or double precision), the

TABLE 1
Examples of user-required constraints applied to scientific simulation datasets

No.	User-Required Constraints	Examples	Science Domains
(A)	Isolating irrelevant value	Hurricane Isabel [32], Katrina [33]	Climate, Weather, etc.
(B)	Preserving global value range	CESM [1]	Climate, etc.
(C)	Preserving value-interval-based error bounds	Katrina [33], NYX	Weather, Cosmology, etc.
(D)	Preserving multiregion-based error bounds	CESM [1]	Weather, Seismic imaging, etc.
(E)	Preserving irregularly shaped regions	QMCPACK, Miranda, CESM [1]	Hydrodynamics, Weather, etc.

objective is to develop an error-bounded lossy compressor that can respect a set of user-defined constraints such as preserving global value range or preserving multiple error bounds based on value intervals or different regions in the dataset.

Three assessment metrics are considered. The first two are compression speed s_c and decompression speed s_d. They are usually measured in megabytes per second: in other words, the size (in MB) of the original dataset processed (either compressed or decompressed) per time unit. The third metric is compression ratio (denoted by ρ), which is defined as follows:

$$\rho = \frac{N \cdot sizeof(dataType)}{Size_{compression}}, \tag{1}$$

where $dataType$ can be either float or double and $Size_{compression}$ is the total size after compression.

Our goal then can be formulated as Formula (2).

$$\begin{aligned} &\text{Maximize } \rho \\ &subject\ to\ \ user\text{-}required\ \ constraint \end{aligned} \tag{2}$$

The *user-required constraint* refers to additional requirements applied to the lossy compression beyond the traditional error-bounding constraint. We formulate the five constraints listed in Table 1 as follows:

$$\text{CONSTRAINT (A): } Preserve\ and\ isolate\ d_i \notin [R_{min}, R_{max}] \tag{3}$$

$$\text{CONSTRAINT (B): } Preserve \begin{cases} \max(\widehat{d_i}) = high(r(D)) \\ \min(\widehat{d_i}) = low(r(D)) \end{cases} \tag{4}$$

$$\text{CONSTRAINT (C): } |d_i - \widehat{d_i}| \leq e(d_i) \tag{5}$$

$$\text{CONSTRAINT (D, E): } |d_i - \widehat{d_i}| \leq e(LOC(d_i)), \tag{6}$$

where $d_i \in D$ denotes the ith data point in the original dataset D, $\widehat{d_i}$ is its corresponding decompressed value, $low(r(D))$ and $high(r(D))$ are the boundaries of the dataset D's value range $r(D)$, $e(d_i)$ denotes the user-required error bound in terms of data point d_i's value (i.e., user-specified error bound in terms of the value interval that covers d_i), $LOC(d_i)$ refers to the spatial location of the data point d_i, and $e(LOC(d_i))$ denotes the user-specified error bound for the specific region covering $LOC(d_i)$. Constraints D and E have identical formulas: the key difference is that E allows irregular shapes, whereas D focuses on a regular shape defined by a rectangular box or cube. We summarize all the notation in Table Table 2.

We give an example to further illustrate how the research problem is formulated in our work. As described above, researchers using the Hurricane Katrina dataset to track the path of the hurricane are concerned only with water surface values above 1m. Based on Formula (2) and Formula (5), the target is to maximize the compression ratio while ensuring that the relatively higher values have lower error bound (e.g., if $d_i \geq 1$, then $e(d_i) = 0.01$; otherwise, $e(d_i)$=0.1). Another example is the Nyx cosmological simulation with a specific quantity of interest, namely, dark matter halo information. According to the Nyx analysis code [16], the dark matter halo cells are computed based on a threshold located in the interval of [80,85], which means that for any data point d_i in [80,85], their error bounds $e(d_i)$ should be lower than $e(d_j)$, where d_j refers to the data points that fall outside of the critical interval [80,85]. Such a multi-interval-based error bound setting can eliminate the distortion of halo cells calculated by the reconstructed data with the same compression ratios. The details will be presented in Section 6.

TABLE 2
Key Notation

Notation	Description
d_i	original data value at position i
p_i	predicted value of d_i
$\hat{d}_i$	reconstructed data value after decompression
$r(x)$	value interval of data point x (d_i)
$e(x)$	specified error bound based on a value interval (x=$r(d_i)$)
$e(LOC(x))$	specified error bound based on the location ($x = d_i$)
$l(x)$	length of some value range ($x = r(d_i)$)
$low(r(x))$	lower boundary of value interval $r(x)$
$high(r(x))$	higher boundary of value interval $r(x)$
q	quantization index (i.e., quantization code)
q_s	shifted quantization number

5 Error-Bounded Lossy Compression Framework with Diverse Constraints

We develop a constraint-based error-bounded lossy compression framework based on the SZ compression model [12]. In the following text we describe our design and how to optimize the performance and quality on this foundation.

5.1 Handling Irrelevant Data

In order to handle the irrelevant values correctly and efficiently, the first three stages in SZ (i.e., prediction, quantization, and Huffman encoding) all need to be modified. The details are as follows.

In stage 1 (data prediction), the key problem is to fill the missing values for the irrelevant data points such that the smoothness of the data will not be destroyed by irrelevant values. This strategy can maintain a high prediction accuracy at each data point throughout the whole dataset. To this end, we use the Lorenzo predicted values [12] to replace irrelevant values. More specifically, for a 1D dataset, the irrelevant data will be replaced by the values of their

preceding data points ($d_i \leftarrow d_{i-1}$); for a 2D dataset, $d_{i,j} \leftarrow d_{i,j-1} + d_{i-1,j} - d_{i-1,j-1}$; and for a 3D dataset, $d_{i,j,k} \leftarrow d_{i-1,j,k} + d_{i,j-1,k} + d_{i,j,k-1} - d_{i-1,j-1,k} - d_{i-1,j,k-1} - d_{i,j-1,k-1} + d_{i-1,j-1,k-1}$. Figure 2 illustrates how irrelevant values are modified in the prediction stage for a 2D dataset. As shown in the figure, the irrelevant value is 1E35. When encountering an irrelevant data point during compression, the values will be estimated based on the Lorenzo predictor: for example, 1.29 ← 1.25+1.27−1.23; 1.33 ← 1.31+1.29−1.27).

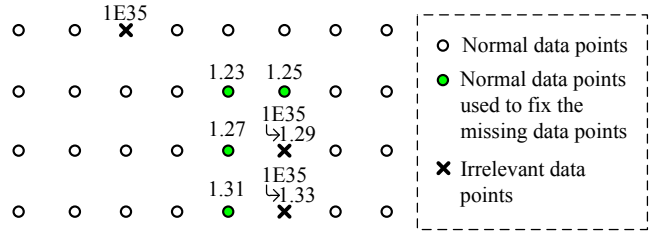


Fig. 2. Illustration of how irrelevant data values are cleared in a 2D dataset.

After modifying the "irrelevant" data points, we propose two strategies to preserve the irrelevant values during the second stage of the compression pipeline.

- **Strategy A:** Since the irrelevant value is often a single floating-point number (such as 1E35), we use a 1-bit array to mark whether this is an irrelevant value for each data point (1 indicates irrelevant value, and 0 indicates normal data).
- **Strategy B:** Use one quantization bin (such as bin #1) from the quantization range to mark whether the data point is an irrelevant value. Thus, there are three types of quantization bins in this case: (1) quantization bin #0 records the unpredicted data value as usual [12], (2) quantization bin #1 marks the irrelevant data, and (3) the remaining quantization bins are used to record the distance between the predicted value and original value.

Each of the two strategies has its own advantages and disadvantages. Strategy A has no impact on the distribution of quantization codes, so it can maintain high Huffman-encoding efficiency on the quantization codes; but it suffers from an overhead of storing the extra bit array. Strategy B does not have such an overhead; but it may affect the distribution of quantization codes to a certain extent, which will inevitably lower the effectiveness of compressing the quantization codes by Huffman encoder.

In the third stage (Huffman encoding), if the solution adopts strategy A, we compress the 1-bit array using Huffman encoding. This compression may significantly lower the overhead because irrelevant data points are generally a small portion of the whole dataset and therefore the 1-bit array is composed mainly of 0s (to be demonstrated in Section 6.1).

5.2 Preserving Global Value Range

The simplest, yet suitably efficient, strategy for preserving the global value range is to include the original value range information as metadata in the compressed data. During decompression, when a reconstructed data value outside the "original value range" is found, the algorithm will replace it with either the minimum value or maximum value of the value range. This strategy introduces little computation overhead in the compression stage because we need only to scan the dataset to find the maximum and minimum values, a process we refer to as "preprocessing" in our evaluation. During decompression, a small computation overhead (generally ∼10% in our experiments) may be introduced by this strategy, because the algorithm needs to check each data point to determine whether the reconstructed value falls outside of the original dataset's value range. If so, it would be substituted by either the maximum or minimum value.

5.3 Preserving Multi-Interval Error Bounds

We define an array of triplets, each containing the *low*, *high*, and *error bound*. Figure 3 illustrates our fundamental idea using a simplified diagram with relatively large error bounds. In this example the user specifies different error bounds for four value intervals: $[-100, 0)$, $[0, 14)$, $[14, 38)$, and $[38, 238)$; the error bounds are 10, 1, 3, and 50, respectively. We then apply different quantization bins (whose length is twice the error bound) in different intervals. As illustrated in the figure, each square denotes a quantization bin in its corresponding value interval. Our algorithm calculates the total number of varied-length quantization bins involved between the predicted value and the raw value during the compression and identifies the quantization bin based on the error bounds during the decompression.

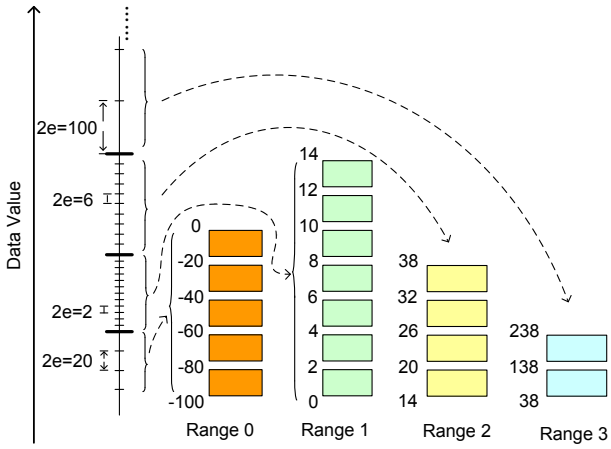


Fig. 3. Multi-interval error bound model. This example shows a few exaggerated error bounds in each range for simplicity of description: each rectangle represents twice the error bound in that range, and the ranges are tightly connected. The error bound will usually be smaller in practice, and each range may contain hundreds or thousands of error bounds.

Before describing our solution in detail, we review the notation. Let d_i denote the original data value at position i. Let p_i denote the predicted value for d_i. We use R to denote the radius of the quantization bin (for instance, if there are 65.536 quantization bins, the radius R is equal to 32 768). Let $\hat{d}_i$ denote the decompressed data value. Let $r(x)$ be a function that returns a value interval index based on a given data value x=d_i. Let $e(x)$ be a function that returns the user-specified error bound based on a given value interval (x=$r(d_i)$). Let $l(x)$ denote the length of some value interval

based on the interval index ($x = r(d_i)$). Let *low*($r(x)$) and *high*($r(x)$) denote the low boundary and high boundary of the value interval $r(x)$, respectively. Let q denote the quantization code, and let q_s denote the shifted quantization number.[2] We summarize the notation in Table 2 in order to help understand the following text.

As illustrated in Figure 3, we design a multi-interval quantization method that calculates the total number of quantization bin indices based on the varied-length quantization bins, followed by other compression techniques including Huffman encoding and dictionary encoding (Zstd). Algorithm 1 presents the pseudocode of the multi-interval quantization in the compression stage.

Algorithm 1 MULTI-INTERVAL QUANTIZATION IN COMPRESSION STAGE

Input: user-specified intervals and error bounds ε
Output: compressed data stream in form of bytes

1: **for** each data point d_i **do**
2: Use the composed prediction that combines Lorenzo predictor and linear regression predictor to obtain a prediction value p_i.
3: $I_p \leftarrow r(p_i)$. /*Obtain interval index of p_i*/
4: $I_d \leftarrow r(d_i)$. /*Obtain interval index of d_i*/
5: **if** $I_d == I_p$ **then**
6: $q \leftarrow round(\frac{(d_i - p_i)}{2e(I_d)})$./*Quantized distance between d_i & p_i.*/
7: **else if** $I_d > I_p$ **then**
8: $t = \sum_{i=I_p+1}^{I_d-1} \frac{l(i)}{2e(i)}$. /*Count bins for middle intervals.*/
9: $t_p = round(\frac{high(I_p) - p_i}{2e(I_p)})$. /*Get quantized distance for I_p.*/
10: $t_d = round(\frac{d_i - low(I_d)}{2e(I_d)})$. /*Get quantized distance for I_d.*/
11: $q = t + t_p + t_d$. /*Get the logic quantization code.*/
12: **else**
13: $t = \sum_{i=I_d+1}^{I_p-1} \frac{l(i)}{2e(i)}$. /*Count bins for middle intervals.*/
14: $t_p = round(\frac{high(I_d) - d}{2e(I_d)})$. /*Get quantized distance for I_d.*/
15: $t_d = round(\frac{p_i - low(I_p)}{2e(I_p)})$. /*Get quantized distance for I_p.*/
16: $q = t + t_p + t_d$. /*Get the logic quantization code.*/
17: **end if**
18: $q_s \leftarrow q + R$. /*Shift quantization code.*/
19: **end for**

For each data point, we must deal with three relationships between the original raw value d_i and its predicted value p_i: (1) $r(d_i) = r(p_i)$: they fall in the same interval; (2) $r(d_i) < r(p_i)$: the predicted data are in some range ahead of the original data; and (3) $r(d_i) > r(p_i)$: the predicted data are in some range before the original data.

Situation 1 (lines 5∼6): If the original raw value d_i and the predicted value p_i fall in the same interval ($i.e., r(d_i) = r(p_i)$), the quantization problem falls back to the traditional linear-scale quantization [12]. Specifically, we can use the following formulas to compute the logic quantization code and decompressed data.

$$q = round(\frac{d_i - p_i}{2e(r(d_i))}) \tag{7}$$

$$\hat{d}_i = p_i + 2e(r(d_i)) \cdot q \tag{8}$$

We use an example to illustrate how the linear-scale quantization works. Suppose the error bound (i.e., $e(r(d_i))$) is 20 and we have d_i=−74, p_i=−95. Then $d_i - p_i = 21$ and $q = round(21/40) = 1$. The decompressed value $\hat{d}_i$ is -75, whose distance to the raw value is less than the error bound.

2. Since the C array has no negative index in an array, the logic quantization bins [-R,R] need to be shifted to $[0, 2R]$ in our implementation.

Situations 2 and 3 (lines 7∼17): These correspond to the situation where the raw value d_i and its predicted value p_i fall in different value intervals (i.e., $r(d_i) \neq r(p_i)$). In the following text, we describe the situation with $r(d_i) > r(p_i)$ (i.e., lines 7∼11 shown in the algorithm); the other situation is similar.

The fundamental idea in handling this situation is to adjust the quantization policy to use various bin lengths or sizes in different value intervals. Specifically, we count the quantized distance (i.e., the number of quantization bins) from the predicted value to the original raw value. Whenever the counter crosses a different interval, we continue to add the quantization bins from the boundary of the new interval. As illustrated in Figure 3, suppose the predicted value is located at -10 and the original value is 100. Then the calculation of the quantization bins involves all the value intervals, and the quantization code is 1+7+4+1=13. The decompressed data would be (38+238)/2=138. Obviously, the decompressed data value is determined mainly by the last value interval and its quantization bin size. The formula for reconstructing the decompressed value is given below (we assume the raw data valuw is greater than the predicted value, without loss of generality):

$$q_t = round(\frac{d_i - low(r(d_i)) - e(r(d_i))}{2e(r(d_i))}) \tag{9}$$

$$\hat{d}_i = low(r(d_i)) + e(r(d_i)) + 2e(r(d_i)) \cdot q_t. \tag{10}$$

Now we describe the decompression in Algorithm 2. The algorithm proceeds by executing similar operations to the compression process but in reverse order to obtain the decompressed data from a predicted data value and the corresponding quantization bins. As shown in the pseudocode, we first calculate the number of quantization bins for each value interval (line 3∼5). We then decompress each data point based on the multi-interval quantization (lines 6∼34). If the raw data value is lower than the predicted value (i.e., $q_j < 0$), the code will scan all the involved value ranges downward (lines 10∼29). Lines 25∼29 refer to the situation where the predicted value and original raw data value fall in the same interval. Lines 13∼23 deal with the other situation where the two data values fall in different intervals.

Note that we need to deal with the edge situation carefully. For instance, when the original data are near the high or low bound of an interval, the quantization value in this final interval might be equal to $quantRange[i]$, causing the decompressed value to be in the next interval unexpectedly. In this case we shift the quantization by 1 in the compression stage to ensure that the decompressed data and original data are in the same interval.

5.4 Preserving Multiregion Error Bounds

Sometimes it is not apparent how to set different error bounds for different value intervals in a dataset; however, one usually knows which regions are likely to be interesting and thus require higher precision than others. For instance, in the CESM [1] dataset, the data indexes correspond to the geolocations, and some regions are more important than others for particular analyses (e.g., oceans, continents). Figure 4 illustrates our approach enabling users to mark interesting regions that we then use to apply a tighter error

Algorithm 2 MULTI-INTERVAL QUANTIZATION IN DECOMPRESSION

Input: compressed data stream
Output: decompressed data stream in the form of bytes

Read value intervals and error bounds in the header and initialize multi-interval quantizer.
Read the quantization bins and unpredictable data.
for each interval index I_i **do**
  $\hat{l}_i = \frac{l(I_i)}{2e(I_i)}$. /*Calculate # quantization bins for each interval*/
end for
for each decompressed data position j **do**
  Use the composed prediction that combines Lorenzo predictor and linear regression predictor to obtain a prediction value p_j.
  $q_j = q_s - R$. /*Get the logic quantization code q_j*/.
  $I_p \leftarrow r(p_j)$. /*Obtain range index of p_j*/
  if $q_j < 0$ **then**
    $\Delta \leftarrow p_j - low(I_p)$ /*Compute p_j's distance to the low boundary*/
    $\hat{\Delta} \leftarrow round(\Delta/2(e(I_p)))$ /*Compute quantized distance*/
    if $q_j + \hat{\Delta} < 0$ **then**
      for i from $I_p - 1$ to 1 **do**
        if $q_j + \hat{l}_i \geq 0$ **then**
          $\hat{d}_j \leftarrow high(i) - e(i) + (q_j + 1) \cdot (2 \cdot e(i))$. /*Get decompressed data*/
          if $\hat{d}_j < low(i)$ **then**
            $\hat{d}_j \leftarrow low(i) + e(i)$. /*Correct decompressed data*/
          end if
        else
          $q_j \leftarrow q_j + \hat{l}_i$. /*Add quantization length for further search*/
        end if
      end for
    else
      $\hat{d}_j \leftarrow p_j + q_j \cdot 2 \cdot (e(I_p))$. /*Compute decompressed value*/
      if $\hat{d}_j < low(I_p)$ **then**
        $\hat{d}_j \leftarrow low(I_p) + e(I_p)$. /*Perform correction to avoid undesired boundary-crossing*/
      end if
    end if
  else if $q_j == 0$ **then**
    $\hat{d}_j \leftarrow p_j$. /*The prediction is accurate, directly use the predicted value*/
  else if $q_j > 0$ **then**
    Calculate the decompressed data using similar methods. /*For brevity we do not include details here. It is similar to the case when $q_j < 0$, with just a few changes to the low and high bounds and some calculation differences.*/
  end if
end for

bound on each region according to the requirement and preknowledge of the data distribution.

To reduce the overhead in (de)compression time, we do not assign a region to each data point; instead, we consider each intrablock of data in the same region. To make the algorithm simpler, we adopt the intrablock of size 6×6×6 for 3D data, which is consistent with SZ's linear regression prediction block size [18]. The undesired side-effect of this method is that the user-customized region (a regular box) may cut through some intrablocks. Since the data have to be compressed/decompressed in the unit of blocks (e.g., 6×6×6 for 3D data), some storage overhead occurs at the edge of the customized region. We consider this storage overhead acceptable because the region of interest is relatively large in practice (at a scale of several thousands) while the block size is far smaller (such as 6×6×6). Keep in mind that the purpose of proposing this region-based algorithm is to reduce the compressed data size while preserving

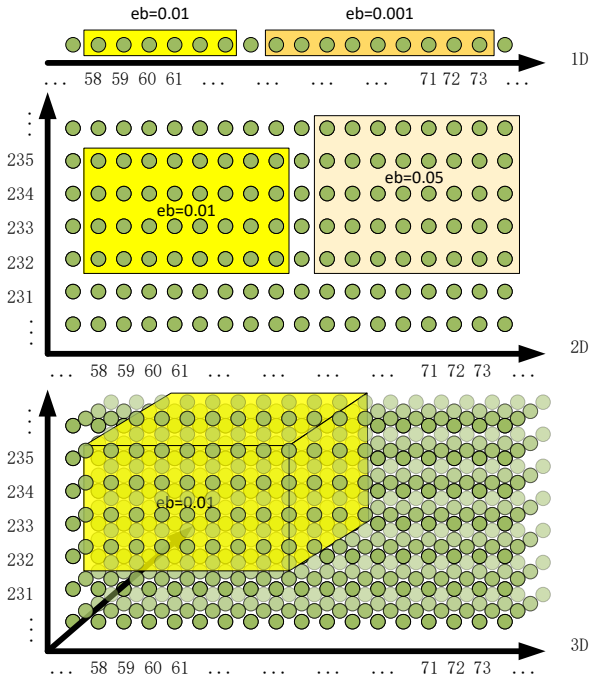


Fig. 4. Constraint(D) region selection for 1D, 2D, and 3D data: In 3D cases, each region can be specified with seven parameters: the starting positions (3 parameters), the length of each direction (3 parameters), and the error bound (1 parameter).

precision for post hoc analysis.

The whole process can be done in the quantization stage if the predictor is fixed, since the varied error bounds will take effect only when calculating the quantization code. However, when we compose the linear regression predictor and Lorenzo predictor together, the data sampling process will need a correct error bound to select an optimal predictor for the current block. The varying error bounds can cause the predictor selection to yield a bad result. This challenge exists in all kinds of blockwise compression where predictors may change according to the error bound for each block. We will describe the solution in detail in Section 5.6.

5.5 Preserving Irregular Regions by Bitmap

To satisfy complex, customized regions of error bounds (rather than just rectangles or cubes), we introduce a bitmap error bound array (as shown in Figure 5). It contains a set of integer values that indicate different data distortion levels, each of which corresponds to a specific error bound value. Such a method allows users to specify an error bound for each data point. However, it is not realistic to manually assign each data point an error bound, since there are usually millions of data points. Instead, users can use third-party software to mark a customized shape in a picture or apply computer vision techniques to obtain contours that distinguish regions (e.g., land and ocean). Such a customized-marking option is more accurate and flexible in practice especially in geolocation-related research (to be demonstrated later).

Although using bitmaps supports the most complex error bound settings—allowing each data point to have its own error bounds—cases rarely require many different error

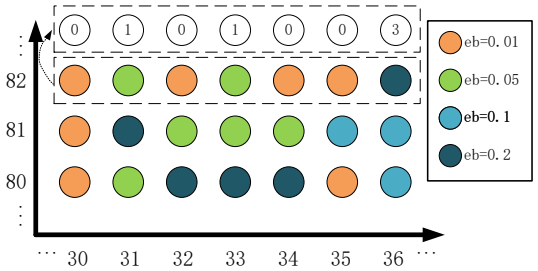


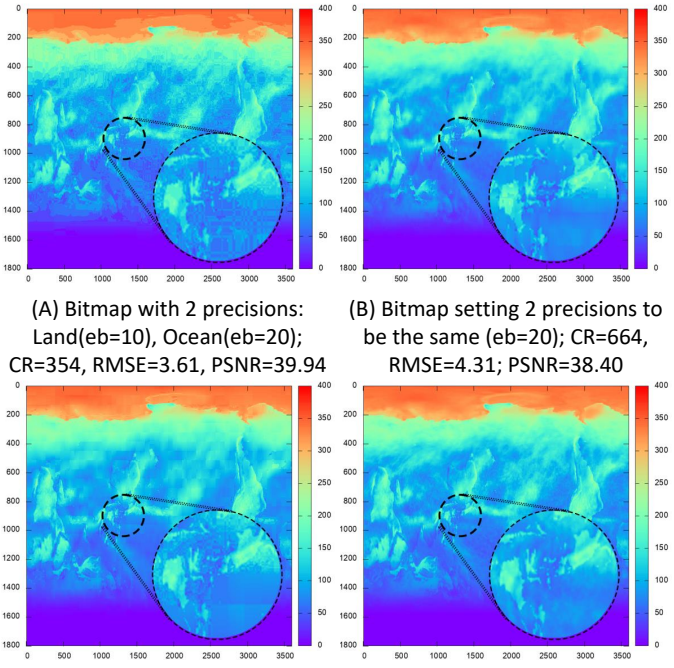
Fig. 5. Illustration of bitmap error bound setting: Use an index to represent the error bound for each data point, and use a separate array to store all possible error bounds.

bounds to coexist in one dataset in practice. Most requirements are limited to a few different error bounds in total, because of coherence of data in space; for example, "higher precision may be required near the hurricane center" or "land areas need higher precisions than ocean areas." Therefore, we use one byte to represent all different types of error bounds. That is, we use a byte array to store the index of error bound for each data point and apply Huffman coding and lossless compression to compress the bitmap array if needed. In the extreme case, the original single error bound would be equivalent to an all-zero bitmap, which would bring almost zero overhead after proper compression. The overhead of using a bitmap array will be presented in Section 6.

The bitmap solution solves a complicated error bound requirement (actually, all possible error bound requirements) and presents an opportunity for automated error bound selection, which may relieve scientists of having to configure advanced bitmap generation algorithms. This solution can also have additional global advantages compared with the region-based method when different error bounds are distributed evenly across the dataset. By setting a fixed proportion of data points with some certain error bounds, we can achieve higher compression ratio, lower root mean squared error, and comparable visual quality (the result will be presented in Section 6).

5.6 Artifact Removal in Multiprecision Compression

The above three multiprecision compression methods may cause undesired artifacts because of their blockwise design. As demonstrated in Figure 6, the two-precision setting ($eb_1 = 20$ at ocean and $eb_2 = 10$ at land) has worse visual quality in the land area (with prominent stripe-pattern artifacts) than does a uniform ($eb = 20$) setting. The root cause is due to the was SZ compresses the data. Specifically, SZ splits each dataset into many small blocks (e.g., 6×6×6 for 3D) and selects the better predictor between Lorenzo and linear regression based on the sampled data points. In general, the Lorenzo predictor may work well when the error bound is relatively low, however it is not as effective as linear regression when the error bound is high [18]. Therefore, the Lorenzo predictor would tend to be selected in each block at relatively low error bounds. Based on our observation, the artifacts shown in Figure 6 (A) are typical and are common to the Lorenzo predictor when the error bound is high. This can be verified in Figure 6 (C), in which the corresponding land area is using a linear



(A) Bitmap with 2 precisions: Land(eb=10), Ocean(eb=20); CR=354, RMSE=3.61, PSNR=39.94

(B) Bitmap setting 2 precisions to be the same (eb=20); CR=664, RMSE=4.31; PSNR=38.40

(C) Bitmap setting same with(A) but change block size to 64; CR=268, RMSE=5.29, PSNR=36.6

(D) The original uncompressed data

Fig. 6. Multiprecision compression problem: In (A), although the RMSE and PSNR behave normally, the continuity of the visualization seems to be broken compared with a lower-precision setting in (B). The block size for the 2D dataset is 32 in SZ3; if we change it to 64, we can obtain the visualization shown in (C).

regression predictor instead of Lorenzo predictor because of increased blocksize (from 32 to 64). Although increasing the blocksize can mitigate the artifact issue to a certain extent, the linear regression predictor may have an oversmooth visualization issue in the corresponding blocks when the error bound is overly large, which may cause undesired block pattern artifacts, as shown in Figures 6 (B) and (C). Moreover, the compression ratio is also degraded (compare Figure 6 (A) vs. (C)), which is undesired.

To overcome the artifact issue, we apply a new predictor—called interpolation—that works well in situations with high user-required error bounds. Specifically, instead of handling the data block by block, the interpolation-based method works level by level and handles every dimension in a unified pattern. This interpolation-based predictor may have much higher prediction accuracy than the linear regression predictor especially at high error bounds. Details about this interpolation-based compression method can be found in our prior work [34]. In this work we combine our multiprecision design for the linear quantization stage with the interpolation-based predictor, which can thus resolve the artifact issue. We evaluate this method in the following section.

5.7 Summary of Proposed Methods and Their Potential Use Cases

In this section, we proposed five constraints along with three multiprecision compression techniques. We will summarize their characteristics and potential use cases below.

TABLE 3
Summary of The Proposed Methods

Constraint	Features	Pros	Cons
Multi-interval	Allow different error bound settings for different value intervals	Obtain higher precision for interesting ranges without decreasing compression ratio much	When setting many ranges, the data near each range boundary may be distorted more, especially if the error bound is large
Multiregion	Allow different error bound settings for multiple regular regions	Require quite minimum metadata to represent each region	Cannot represent complicated boundary; Only rectangular regions can be represented
Irregular Region	Allow fully customizable error bound settings for each data point	Represent all kinds of regions, fully customizable	Require a bitmap that is of the same dimensions as the original data, extra space cost

Irrelevant data will almost always need to be cleaned with some method when existing. Global range should also often be respected because otherwise certain post-hoc analysis including color heatmap will render visually different results compared to the original data. Therefore, we consider these two constraints basic requirements and universally applicable for many datasets.

As shown in TABLE 3, the three multiprecision compression methods allow users to set different error bounds for different parts of data, but they have varied characteristics. In summary, the multi-interval method is suitable when value ranges have varied importance to users; the multi-region method targets at those scenarios where interesting data are in rectangular regions; the irregular region method is useful in geographical data with complicated boundaries.

6 Experimental Evaluation

In this section we use multiple real-world simulations to evaluate our multiprecision compression methods, and we compare the compression quality and performance with the global constant error-bounded lossy compressor SZ, which has been verified as one of the best error-bounded lossy compressors in most cases.

We evaluate our approaches on datasets generated by seven scientific applications: QMCPACK [35], RTM [36], Miranda [37], CESM [1], Nyx [16], Hurricane Isabel [32], and Hurricane Katrina [33], as presented in Table 4.

All time measurements are performed on the Argonne Bebop Machine, which is a HPC cluster managed by Laboratory Computing Resource Center (LCRC) at Argonne National Laboratory. It is equipped with 1200+ broadwall nodes (Intel Xeon E5-2696v4), each having 36 cores with a total of 128 GB DDR4 memory.

The two quantities we used to measure the data quality are RMSE and PSNR, and we will briefly introduce their meaning below. The root-mean-square error (RMSE) is a frequently used measure of the differences between values. We calculate the mean error between the decompressed data values and the original values to understand how much error the compression algorithm brings into the data. The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise. PSNR will not be severely affected by the data ranges, and we can have a universal understanding of how good the data is.

TABLE 4
Basic dataset information

Dataset	# Fields	Dimensions	Science
QMCPACK	1	33120×69×69	electronic structure of atoms, molecules, and solids
RTM	1	449×449×235	Electronic
Miranda	7	256×384×384	hydrodynamics code for large turbulence simulations
CESM	79	1800×3600	Climate
Nyx	6	512×512×512	Cosmology
Hurricane Isabel	13	100×500×500	Weather
Hurricane Katrina	1	162×417642	Weather

6.1 Preserving Irrelevant Data (constraint A) and Global Value Range (constraint B)

The Hurricane Isabel dataset contains irrelevant data values marked as 1E35, which is well outside the normal value range. Table 5 shows the value range for five of 13 fields in the dataset which contain irrelevant (or missing) data points. The reason for the missing values is that the data simulates an actual event (a hurricane) and, in the locations where there is ground, no meaningful wind speed or pressure is recorded. More information about the dataset is available on the website.[3]

TABLE 5
The 5 fields tested in the hurricane dataset

Field	Description	Value Range
P	Pressure (weight of atmosphere above a grid point)	-5471.8579/3225.4257
TC	Temperature (Celsius)	-83.00402/31.51576
U	X wind speed (positive means winds from west to east)	-79.47297/85.17703
V	Y wind speed (positive means winds from south to north)	-76.03391/82.95293
W	Z wind speed (positive means upward wind)	-9.06026/28.61434

Figure 7 shows the distribution of data points in the Hurricane Isabel dataset. Because the actual value of the irrelevant data is far too large to be put in the same figure with normal data, we use a made-up value that is outside the range of each field to represent the irrelevant value. We can see that every field contains a non-negligible amount of irrelevant data, although not as many as normal data points. While the amount of irrelevant data is small, such data may severely harm the overall compression ratio because they are mixed among normal data points, destroying the continuity of normal data. We verify this statement by sampling a random continuous portion of the temperature field, as shown in Figure 8.

Figure 8 clearly shows that irrelevant data are distributed among normal data, destroying the smoothness of the data space. Obviously, if we predict a normal data point

3. http://vis.computer.org/vis2004contest/data.html

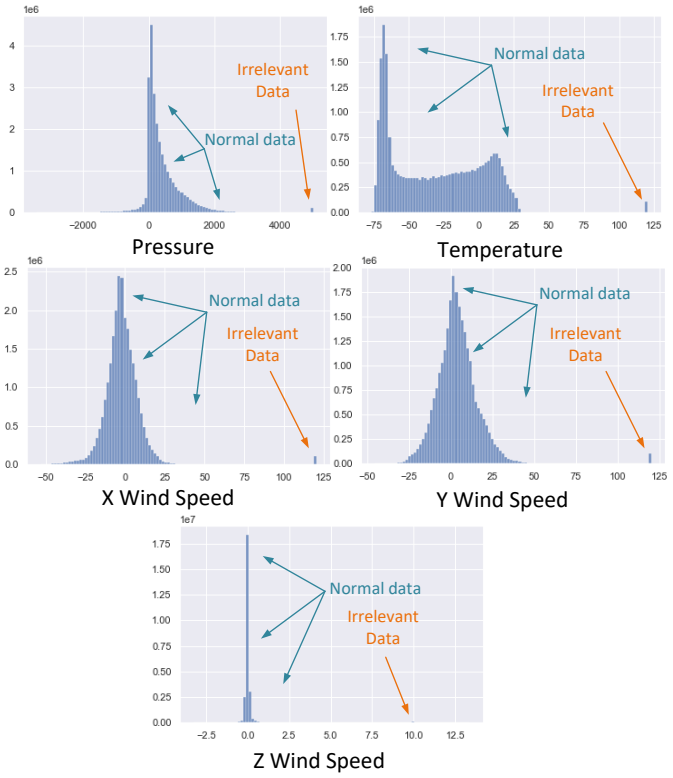


Fig. 7. Data distribution of the five fields in the hurricane dataset. The irrelevant data value is 1E35. To visualize it in the distribution figure, we modify the big value to a made-up outside value that is not in the normal data range.

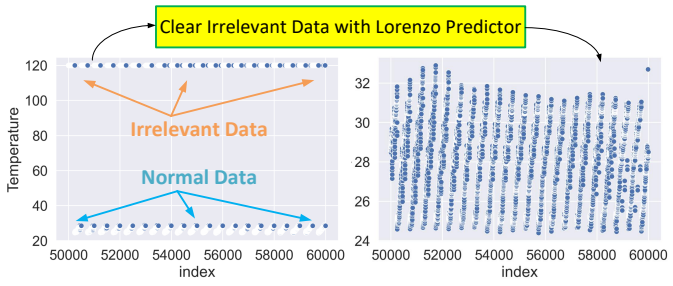


Fig. 8. Temperature data points with (left) and without (right) irrelevant data. We show only a sample of 10,000 points (between index 50,000 and 60,000 in the original dataset). We observe the data points in the given index range and can see that the irrelevant data is mixed among the normal data points, harming data continuity; after clearing them with the Lorenzo predictor, the separating effect disappears.

using the irrelevant data value, the prediction cannot be precise. As lossy compressor designers, we want to preserve irrelevant data values while mitigating their influence on the compression ratio. Note that even though they appear to be irrelevant for compression, they carry potentially useful information—in this case, they indicate ground locations.

In Figure 9, we investigate five different ways of handling irrelevant data. Time is measured on Bebop. The five strategies are: *Ignore* treats all irrelevant data as normal data; *Zero* replaces all irrelevant data by 0 for simplicity; *Clear* replaces all irrelevant data using the Lorenzo predictor based on their nearby values (our solution); *Quant* and *Bitmap* indicate the storage algorithm: Quant refers to using one additional quantization bin to mark irrelevant data, and

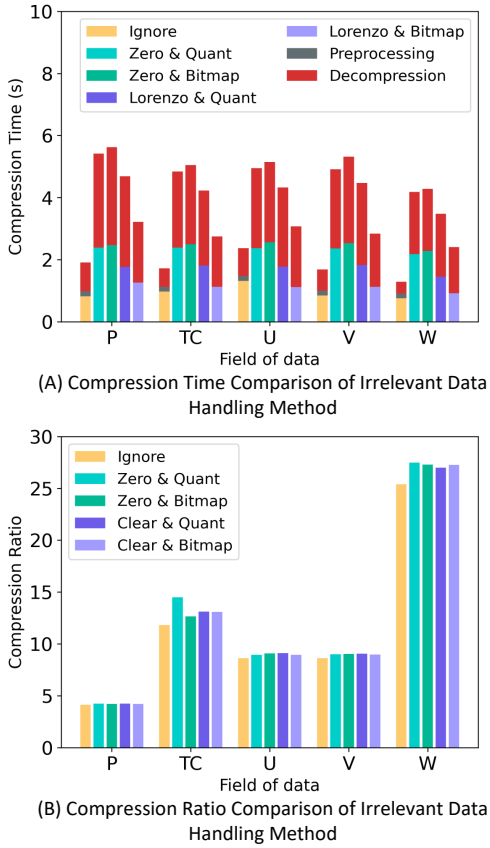


Fig. 9. Performance of irrelevant data-handling methods: all methods slightly improve the compression ratio with a cost of longer compression time and decompression time.

Bitmap indicates that we use a bit array containing 1 and 0 to indicate whether each data point is an irrelevant value or not. Figure 9(A) shows that handling the irrelevant data may double the compression and decompression time. The overhead is due primarily to additional traversing of the whole dataset to find, clean, and recover irrelevant data. Moreover, constructing additional Huffman trees for irrelevant data will add additional time to the compression and decompression. Figure 9(B) shows that handling irrelevant data is generally better than ignoring them; however, it is difficult to determine whether it is better to clear them with the Lorenzo predictor or simply convert them to 0. Moreover, the simple bitmap method and quantization method exhibit similar performance. The likely reason is that irrelevant data are only a very small portion of the entire data and thus the methods are unable to demonstrate a huge difference in terms of the overall compression ratio. We conclude that in this scenario the quantization strategy slightly outperforms use of a bitmap.

The global range constraint is the easiest one to deal with, which requires only a scan in the preprocessing stage to obtain the max and min value. After the decompression, an additional traverse will be sufficient to pull back those few points whose values are beyond the min or max value. The time overhead is nearly negligible, as indicated in the gray bar in Figure 9 (A).

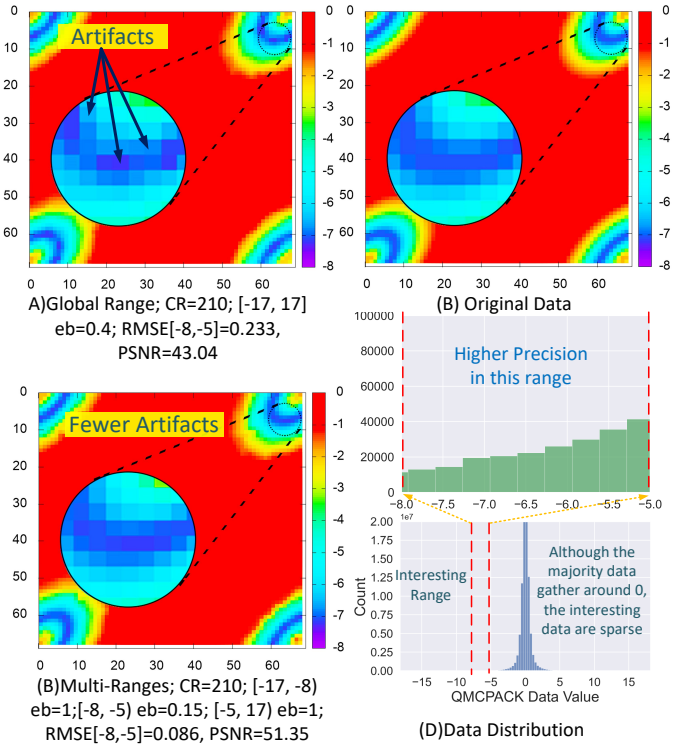


Fig. 10. QMCPACK data: (A) The basic method is setting one error bound for the global range. We can see obvious artifacts in the blue area. (B) Applying our multi-interval algorithm, we focus on the interesting range [-8,-5) and give it a tighter error bound 0.15 while leaving other ranges a higher error bound 1. We can see fewer artifacts, while the compression ratio is kept the same as the global range method. Compared with the original data shown in (C), we can see that the data in the interesting range have better visualization results.

6.2 Multi-interval Error-Bounded Compression (Constraint C) Based on Visual Quality

Figures 10 and 11 show the substantial advantage of our multi-interval error bound-based compression over the traditional constant error-bounded compression, using two datasets (QMCPACK and Miranda). Specifically, the multi-interval-based compression preserves higher visual quality for the value intervals of interest, while achieving the same or even higher overall compression ratios by lowering precision on insignificant value intervals. For instance, in the QMCPACK dataset, over 90% of the data points are located around 0, but they are smooth and easy to be predicted by neighboring data points; however, the data points with values in the interval of $[-8, -5]$ are the sparse interesting values that are harder to be predicted accurately. That is, they are more important to preserve the overall visual quality because the distortion of their values is easier to observe in the visualization image.

Our method grants a tighter error bound and thus a higher precision in the more important value intervals, while allowing more distortion in insignificant value ranges, such that the overall compression ratio is not degraded. Detailed evaluation results are shown in Table 6 and Table 7. Given similar compression ratios, our method can achieve lower RMSE and higher PSNR in the critical value interval.

TABLE 6
QMCPACK RMSE & PSNR Comparison

Method	Range	eb	RMSE	PSNR
Global Range CR=210	[-17, -8]	0.4	0.232	43.067
	[-8, -5]		**0.233**	**43.041**
	[-5, 17]		0.051	56.159
Multi-Intervals CR=210	[-17, -8]	1.0	0.538	35.747
	[-8, -5]	0.15	**0.086**	**51.623**
	[-5, 17]	1.0	0.089	51.354

TABLE 7
Miranda density RMSE & PSNR Comparison

Method	Range	eb	RMSE	PSNR
Global Range CR=206	[0.5, 1.4]	0.07	0.012	44.804
	[1.4, 2]		**0.036**	**34.801**
	[2, 3.5]		0.015	42.379
Multi-Intervals CR=207	[0.5, 1.4]	0.1	0.013	43.5813
	[1.4, 2]	0.05	**0.027**	**37.193**
	[2, 3.5]	0.1	0.018	40.682

6.3 Multi-Interval Error-Bounded Compression Based on Post Hoc Analysis in Nyx Cosmological Simulation

We now consider compression of the Nyx cosmological simulation with a specific quantity of interest (i.e., dark matter halo cell information). Dark matter halos play an important role in the formation and evolution of galaxies and consequently in cosmological simulations. Halos are overdensities in the dark matter distribution and can be identified by using different algorithms; in this instance, we use the friends-of-friends algorithm [38]. For the Nyx simulation, which is an Eulerian simulation instead of a Lagrangian simulation, the halo-finding algorithm uses density data to identify halos [39]. For decompressed data, some of the information can be distorted from the original, such as halo cells and halo mass.

Figure 12 demonstrates that setting different error bounds for different value intervals in Nyx simulation datasets can preserve the features of interest (i.e., halo cells in this example) better than global-range error-bounded compression can. The key reason is that according to the Nyx halo analysis code, the values in the range of [81,83] need to be extremely precise (the reason is related to the sophisticated physics, and we ignore the details here). For our compression task, we set three value ranges and assign a smaller error bound (0.1) to the data in the range of [81,83]. In this way the overall compression ratio will be higher with less distortion on the halo visualization result, as shown in Figure 12.

Table 8 shows the substantially higher precision of our multi-interval error-bounded compression over global-range error-bounded compression. We use RMSE of cell number differences of halos and RMSE of mass differences of halos in comparison with original data as two main metrics to evaluate the results. Specifically, when passed through the post hoc analysis, our multi-interval solution can lead to significantly lower RMSE for cell number and halo mass, compared with the original RMSE under the global-range error-bounded compression.

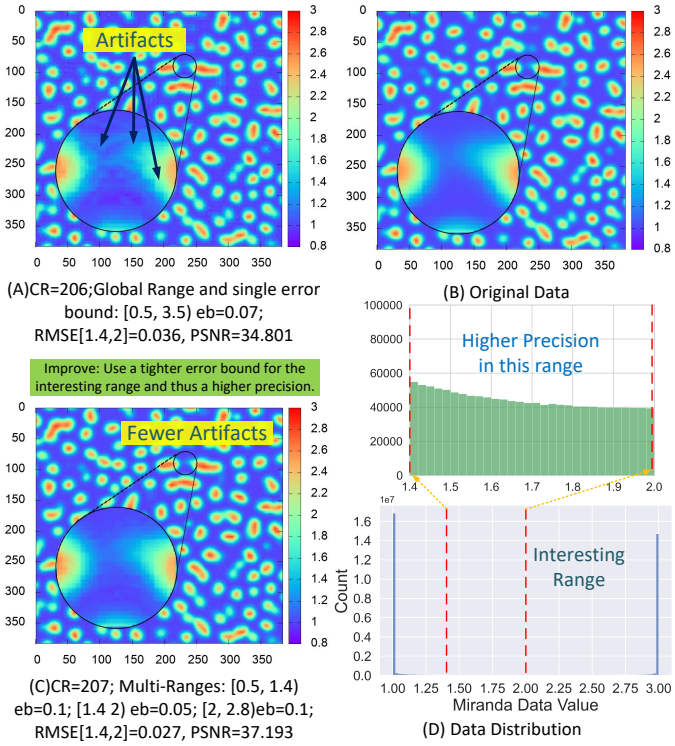


Fig. 11. Miranda density slice No. 120. Comparing B with A, we can see that not only can the multi-interval solution preserve a high precision at a value range of interest with high compression ratio, but it also prevents the blue regions from getting distorted.

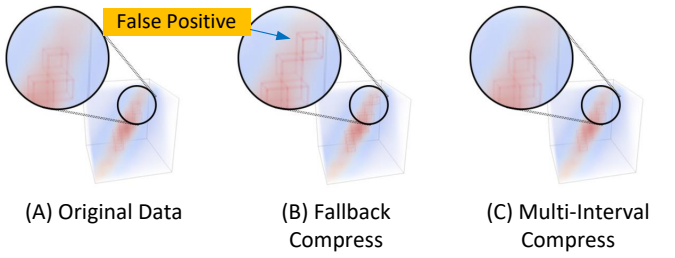


Fig. 12. Nyx halo cell visualization: The fallback method sets a global error bound to be 0.5, and the compression ratio is 75. Our solution (C) sets three ranges: [min, 81) with error bound 1, [81, 83) with error bound 0.01, and [83, max) with error bound 1, and the compression ratio is 78. In the visualization, our multi-interval solution (C) has cells almost identical to the result using the original data, while the fallback method (B) shows some distortion, and the cells' position and number are not identical to (A).

6.4 Multi-interval Error-Bounded Compression Using Hurricane Katrina Simulation

We now investigate the combination of our methods on the Hurricane Katrina dataset. The combined methods include handling irrelevant data, multi-interval error bound settings, and different predictor settings (Lorenzo/linear regression).

Hurricane Katrina was one of the most devastating storms in the history of the United States because of its resulted significantly high storm surge (over 10 meters on the Mississippi coast) and high velocity. To model Katrina, the area was discretized into 417,642 nodes forming 826,866 unstructured meshes. The simulation was performed with a 1-second time step, from 18:00 UTC August 23 through 12:00 UTC August 30, 2005. The output hourly water elevation data downloaded from the ADCIRC website (adcirc.org) was used in this study, and the water elevation contour map with a 1-meter interval at four times—3:00 am and 17:00 pm UTC August 28 and 3:00 am UTC and 14:00 pm UTC August 29—was plotted for illustrative comparison.

TABLE 8
Comparison of Different Range Settings. Fallback sets only a global error bound (here 0.01 and 0.5). Multi-interval uses our multi-interval error-bounded compression with three error bounds ([min, 81)=1, [81, 83)=0.01, and [83,max)=1)

Method	RMSE of cell number	RMSE of halo mass
Fallback-0.01:	0.089	125.84
Fallback-0.5:	2.820	429.26
Multi-interval:	0.198	135.41

Katrina caused water elevation, and we wish to preserve more precisely the information about the elevation data that are above 1 meter (the multi-interval constraint). Moreover, some data points do not have meaningful values in this dataset and are represented by -99999 (irrelevant data). Therefore, we need to treat these values properly to mitigate their influence on the compression performance. By considering both irrelevant data and multi-interval error-bound constraints, the compression quality (as shown in Figure 13) can be improved significantly compared with the original compression quality under the state-of-the-art SZ 2.1; see Figure 13 (D) and (E) vs. (B) and (C).

6.5 Multiregion Error-Bounded Compression (constraint D) Based on Visual Quality

To demonstrate the power of the region-based compression method, we perform a post hoc analysis of three regions in the QMCPACK dataset: slices 200, 300, and 400. Since each slice will usually be observed in one analysis step, it is better to set a suitable error bound for each slice instead of using a uniform error bound. For example, an error bound of 0.001 might be suitable for a slice with the data value range [-0.5, 0.5] but would be too large for a slice with range [-0.0025, 0.0005]. In Figure 14, we can see significant distortion in the selected regions in (C) even though the error bound is generally small (0.01) for the whole dataset. Our solution improves the quality by applying tighter error bounds on the three regions/slices. The compression ratio may not drop clearly, because the "tight-error-bound regions" are small compared with the global dataset.

In addition to addressing some chosen slices with specific regions, the region-based compression algorithm can achieve the effect of "different precisions for different areas" in each slice. As shown in Figure 15, the left-bottom corner has much better visual quality than the other corners. With these two examples, we demonstrate the flexibility and locality of this region-based compression algorithm. In general, setting some small regions for some parts of the data that are of interest to the researchers will not influence the global compression quality. Moreover, researchers can set any number of regions in any parts of the dataset. Although it does not make sense to set hundreds of regions to select every possible interesting data points, the region-based algorithm offers the flexibility to accommodate complex requirements and demands.

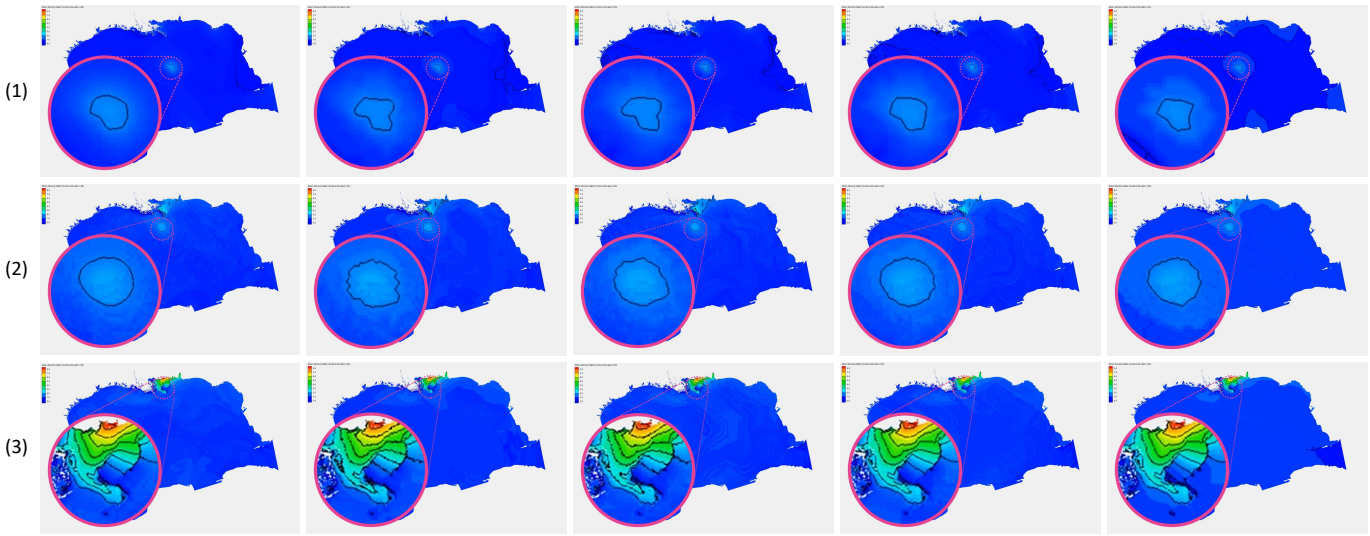
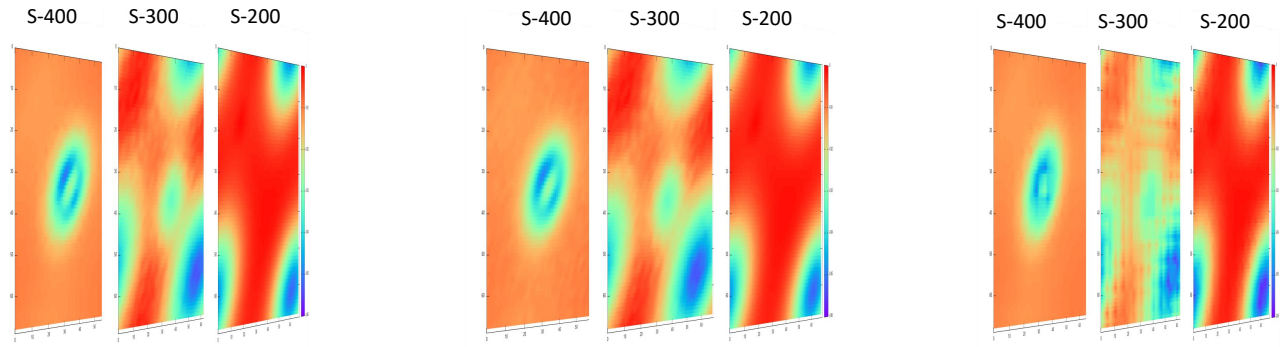


Fig. 13. Hurricane Katrina data: Each row is a frame of the Katrina simulation: (1) is frame 120, (2) is frame 130, and (3) is frame 141. Each column represents a different setting of ranges and error bounds. Most of the blue data points in the graphs are close to zero. By applying a global range with error bound to be 0.01 with our solution, the visualization is almost identical to the original data's, and therefore we use one column (A) to demonstrate the visualization result as a reference. The fallback version shown in (B) is to use the original 1D SZ compressor, which has only the Lorenzo predictor and does not handle the irrelevant data; thus it has the lowest compression ratio even with a higher error bound 0.1. "Composed" in (C) and (D) means we use a composed Lorenzo and linear regression predictor to predict values. "Lorenzo" in (E) means we use only the Lorenzo predictor with no linear regression. Comparing (B) and (C), our solution wins on the global range test by handling the irrelevant data and using the composed predictor (both Lorenzo and linear regression). Comparing (C) and (D), our multi-interval solution wins in both the compression ratio and visualization result. Comparing (D) and (E), we can further improve the compression ratio by using the Lorenzo predictor only and allowing some distortion in the deep blue area.



(A) Oursol (CR=54): multiple regions, create a small region-box for each significant region [190 0 0:20 69 69], [290 0 0:20 69 69], [390 0 0:20 69 69], and give each region a dedicated error bound.

(B) Original Data: the value ranges for the demonstrated regions are different, and each region requires a different precision to have a good visualization result.

(C) SZ3 All-0.01 (CR=27): the old method cannot take care of all regions. Even when giving a quite tight error bound 0.01, some regions will be hugely distorted.

Fig. 14. QMCPACK visual quality comparison: Each slice has 69×69 pixels. We select slice 200, 300, and 400 to observe the visual distortion because each has a different range: slice 200 has range [-0.06, 0], slice 300 has range [-0.0016, 0], and slice 400 has range [-0.0025, 0.0005].

The feature of being able to set "different precisions for different areas" is extremely useful in climate data. Scientists and policy makers from different nations may share the same global climate data while focusing on their own country's details. We use the CLDHGH field in the CESM dataset to exemplify this feature. Since the dataset has a tight value range and the neighboring values are smooth, it is hard to visualize the difference directly between the decompressed data and the original data in a small picture. We calculate the difference between each data point and visualize the difference instead. In Figure 16 (B), we can clearly see that the data inside the region (circled by a red rectangle) are much more precise than in the other areas since there are almost no artifacts in the difference image. In reaching the desired precision for the regions of interest, the region-based method clearly outperforms the traditional SZ compressor.

We also evaluate the (de)compression time overhead of both multi-interval and multiregion methods. The overhead of the multiregion method is proportional to the number of regions, since each block needs to check the region list to find which region it belongs to. In contrast, the overhead of the multi-interval method is highly related to the precision of the prediction. To make the performance measurement as

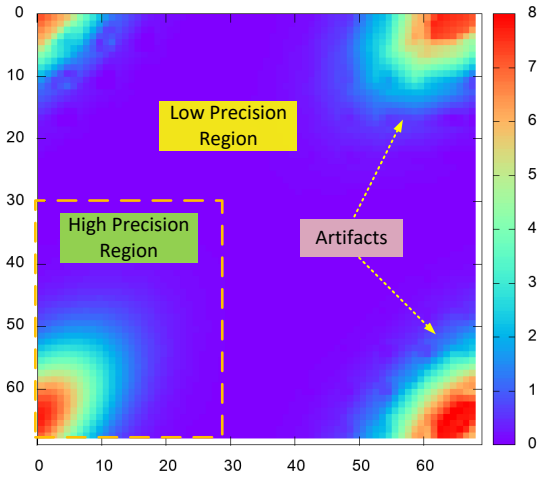
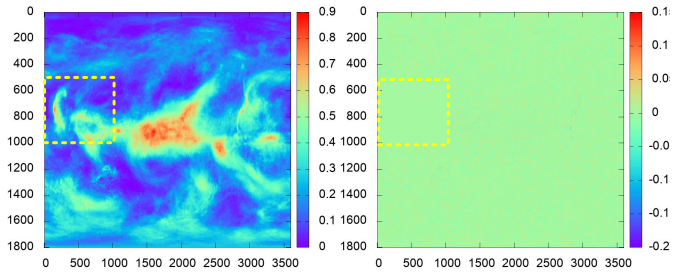


Fig. 15. QMCPACK Slice 450, value range [0, 8]: A higher precision 0.001 for data in the area where $x \in [0, 30]$ and $y \in [30, 69]$, while keeping the error bound of other areas 0.5; The compression ratio is 242, and the SZ3 method with global error bound equal to 0.5 has a compression ratio 243. The region almost does not harm the compression ratio at all.



(A) CLDHGH Data: The climate data map, the shape of which corresponds to the geolocations on earth

(B) The difference image using region-based compression

Fig. 16. CESM with a region: while keeping the compression ratio high (CR=316), we make the interesting region more precise (eb=0.01). The error bound for the remaining regions is 0.02 in this example. If the SZ3's global error bound is used to reach eb=0.01 for the desired area, the compression ratio is 57.

fair as possible, we use the same error bounds for all regions and value intervals on 6 datasets, and we set 5 different regions/intervals for each compression to guarantee that the overhead is observable.

The compression tasks are performed on the Bebop bdwall partition with a single node, and we record the average of 10 runs for each compression configuration. As shown in Figure 17 and Table 9, the compression time overheads of both the multi-interval method and region-based method are not very high. The region-based method has slightly smaller overhead compared with the multi-interval method. The main reason is that our region-based method does not follow a point-to-point evaluation; instead, we stipulate each intrablock of the same region, cutting down considerable unnecessary computation. The same approach cannot be applied to the multi-interval method because we cannot assume neighboring points to be in the same value interval: actually, they are likely to be in two different value intervals specified by the user. To summarize, both methods lead to a certain compression time overhead, while the overheads are confined within an acceptable range.

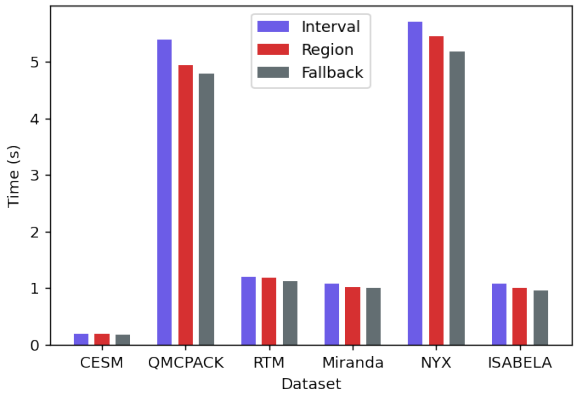


Fig. 17. Comparison of compression time: The reference point is the *Fallback* version, which means using a uniform error bound for all data points. The overhead of the region-based method is slightly lower than that of the multi-interval method.

TABLE 9
Compression Time and Overhead of Interval/Region/Fallback Methods

Method	CESM	QMC	RTM	MIRAN	NYX	ISAB
Interval(s)	0.20	5.39	1.20	1.08	5.70	1.08
Region(s)	0.19	4.94	1.18	1.03	5.46	1.01
Fallback(s)	0.18	4.80	1.12	1.00	5.18	0.96
Interval%	8.9%	12.3%	7.1%	6.8%	10.0%	13.0%
Region%	3.3%	3.0%	5.4%	1.9%	5.4%	5.7%

6.6 Bitmap-Specified Error Bound Compression (Constraint E)

A bitmap defines the most concrete error bound information since it specifies an error bound for each data point. The overhead of storing a bitmap is non-negligible if not properly compressed. In the following, we evaluate two methods for storing the bitmap-specified error bounds: (1) the bitmap array is background information that is stored separately by users as metadata (e.g., the world map); and (2) the bitmap needs to be stored with compressed data so it must also be compressed.

6.6.1 Situation 1

We consider the CESM dataset as an example to evaluate the first bitmap method. Our bitmap solution can help users specify different precisions with fine granularity on irregular regions, in contrast with the other regular-region-based multierror-bounded compression method.

In the CESM dataset, we retrieve the bitmap array by using the LANDFRAC field, because it is a good match for separating the land and ocean area in a world map (as shown in Figure 18 (F)). Applying LANDFRAC as the bitmap, we test four different compression settings (described in Table 10) on the other five data fields, as shown in Table 11. In Table 10 we can see that the bitmap solution sacrifices precision in the red area and can obtain a higher compression ratio. The overall PSNR will decrease when enlarging the error bound for red areas, but the compression quality for the interesting areas (here, the blue areas are considered interesting areas) remains the same—P_0 almost does not decrease, while P_1 decreases because of a larger error bound set in the corresponding area.

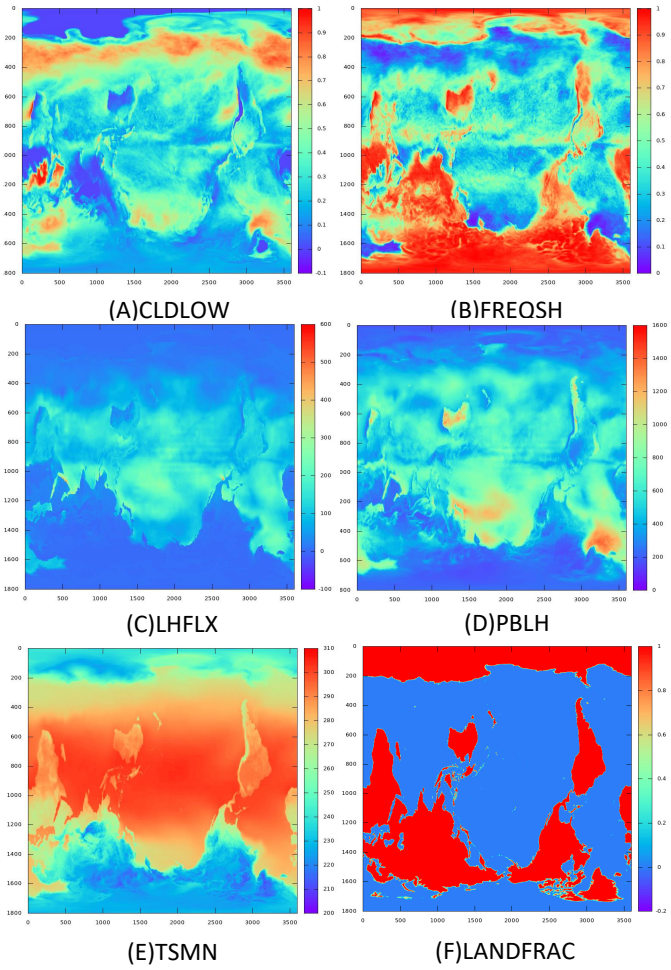


Fig. 18. Six fields in CESM: the visualization indicates that bitmap-separated precisions may be suitable to compress these fields.

TABLE 10
Compression Setting Definition

Setting	Description
A	SZ2.1 [18]: Lorenzo & Linear Regression Predictor with one global error bound
B	Use SZ2.1's predictor, but adopt two error bounds set by a bitmap array
C	Interpolation-based compression with one uniform error bound [34]
D	Our developed region-based error-bounded compressor with two error bounds set by a bitmap

Table 11 demonstrates that our region-based multierror-bounded compression method significantly outperforms all other solutions in compression quality. The reason is twofold. (1) Our developed bitmap method can be used to fine-tune the precisions for different irregular regions, which can preserve the quality for regions of interest more effectively while reaching a high compression ratio. This can be verified by comparing the settings C and D in the table. (2) As we discussed in Section 5.6, the interpolation predictor is much more effective than the linear regression predictor used by SZ2.1. This can be verified by comparing settings A and C in the table. The artifact issue described in Section 5.6 no longer exists when applying the interpolation predictor, based on our experiment. We do not show a visualization image here because of space limits. In fact, its visualization for the interpolation method is almost indistinguishable from Figure 6 (D).

TABLE 11
Impact of compression settings on compression ratio (CR) and PSNR for the six CESM fields of Fig 18: P_0/P_1 are the PSNR in the bitmap separated blue/red area, respectively; CR' is the compression ratio that takes the bitmap into account

Data Field	Setting	CR	CR'	PSNR	P_0	P_1
CLDLOW min=-0.1 max=1	A: eb=0.01	21	-	44.94	46.74	49.59
	B: eb=0.01, 0.1	30	29.0	29.71	46.74	29.73
	C: eb=0.01	138	-	47.14	49.23	51.26
	D: eb=0.01, 0.1	224	176.6	32.31	49.22	32.34
FREQSH min=0 max=1	A: eb=0.01	16	-	44.73	46.76	48.97
	B: eb=0.01, 0.1	22	21.4	28.67	46.76	28.67
	C: eb=0.01	88	-	46.79	48.83	50.99
	D: eb= 0.01, 0.1	126	109.5	32.10	48.83	32.13
LHFLX min=-100 max=600	A: eb=1	30	-	60.27	62.28	64.55
	B: eb=1, 10	48	45.4	49.36	62.28	49.55
	C: eb=1	106	-	62.41	64.58	66.40
	D: eb= 1, 10	216	171.6	47.81	64.63	47.84
PBLH min=0 max=1600	A: eb=5	37	-	53.04	55.20	57.07
	B: eb=5, 15	45	42.7	47.72	55.20	48.55
	C: eb=5	107	-	55.03	57.24	58.99
	D: eb= 5, 15	169	140.5	49.23	57.26	49.93
TSMN min=200 max=310	A: eb=1	66	-	44.78	47.04	48.64
	B: eb=1, 10	191	155.4	36.19	47.04	36.51
	C: eb=1	292	-	47.14	49.41	50.99
	D: eb= 1, 10	812	411.5	31.64	49.24	31.66

6.6.2 *Situation 2*

In the second situation where the bitmap array needs to be stored together with the compressed data, we compress the bitmap array by integer-based Huffman encoding [6] and Zstd [19]. Specifically, the input data is the integer bitmap array with the same number of elements as the original dataset.

Table 11 shows the compression ratio of our region-based multierror-bounded lossy compression method (denoted as CR') after embedding the bitmap into the compressed data. Since uniform error-bounded compression does not need to store the bitmap array, this column shows only the compression ratios for settings B and D. We observe that CR' is close to CR (i.e., the compression ratio without storing the bitmap array) in most cases. The reason is that the bitmap array is fairly easy to compress with high ratios (reach ~800 in this example) because of the limited number of error bounds. In fact, there are typically few error bounds in practice because of the limited number of value intervals of interest or regions of interest in general. Accordingly, the error level values would likely exhibit repeated patterns in the bitmap array, especially for the consecutive data points in space, leading to a very high compression ratio.

6.7 Compression Time and Scalability

To evaluate the compression time and scalability, we run a series of tests in parallel on thousands of CPU cores on the Argonne LCRC Bebop supercomputer [40].

We use the QMCPACK dataset for these experiments. According to the visualization results we obtained from the preceding section, we observe that no data distortion can be viewed by the naked eye as long as a relatively low error bound of 0.15 is used. However, considering the potential

impact of the lossy compression on the user's analysis, we set a very low error bound (1E-5) for the range of interest: [-8,-5). Preserving this condition, we perform the experiments on the Bebop supercomputer with different numbers of cores (each core has 600 MB of raw data to compress). The results of BDW partitions are shown in Figure 19.

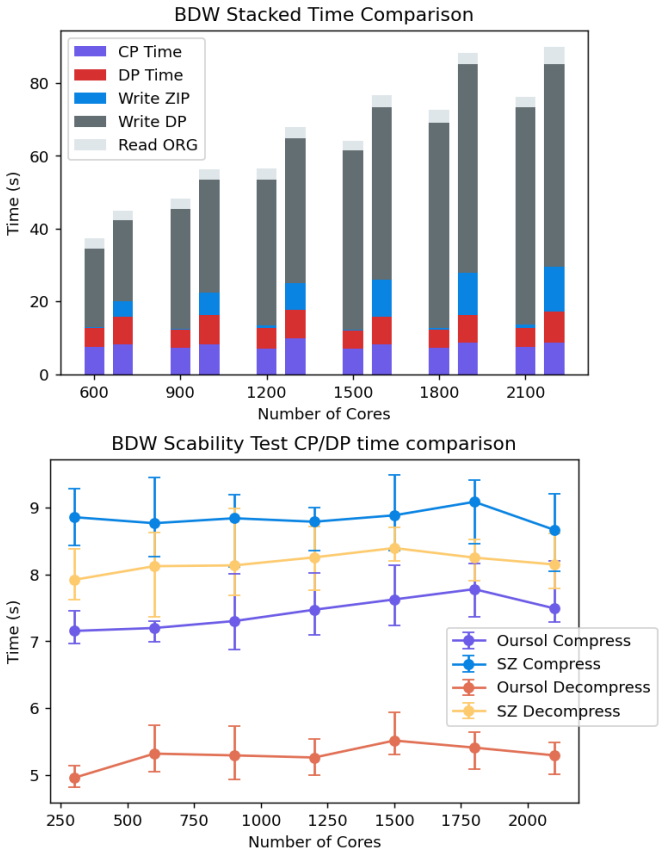


Fig. 19. BDW partition: for each pair of bars, the left side is multi-interval solution's result, and the right side is SZ's result. CP/DP Time are compression/decompression time respectively. Write ZIP/Write DP are the I/O time to write the compressed/decompressed file respectively. Read ORG is the time to read the original file.

TABLE 12
Time cost for each stage of SZ running with 2100 Cores; The variance is calculated based on the 5 runs.

RUN	1	2	3	4	5	Variance
CP Time	9.11	8.67	8.3	9.21	8.05	0.25
DP Time	8.42	8.61	8.11	7.82	7.79	0.13
Write ZIP	12.17	112.75	30.29	30.11	16.93	1697.60
Write DP	55.77	53.84	176.56	175.79	162.33	4122.01
Total Time	95.61	229.77	304.32	304.56	231.38	7274.09

TABLE 13
Time cost for each stage of multi-interval algorithm running with 2100 Cores; The variance is calculated based on the 5 runs.

RUN	1	2	3	4	5	Variance
CP Time	7.38	7.3	7.29	7.29	8.2	0.16
DP Time	5.49	5.16	5.01	5.3	5.49	0.04
Write ZIP	39.78	45.46	46.05	1.49	1.43	542.87
Write DP	192.09	115.54	115.76	81.56	88.38	1925.85
Total Time	256.77	181.45	181.97	107.15	109.32	3850.00

As shown in TABLE 12 and TABLE 13, the (de)compression time is very stable, but the I/O time varies a lot for different runs. The reason for a large variance in I/O is that the Bebop machine is a shared system, and the disk I/O time will be influenced by other users' tasks.

In Figure 19, we see that the write time takes an increasing portion of the total time as we increase the number of cores. Obviously, the I/O cost scales worse than our lossy compression/decompression performance, especially because of the limited number of I/O nodes used by the system.

Based on our results, we observe that the (de)compression time does not increase with the number of cores, which shows that both our algorithm and SZ have very good scalability. The key reason for good scaling is that the lossy compression adopted in practice follows an embarrassing parallel mode: no communication exists among the execution ranks/cores. The key reason our algorithm has lower compression/decompression time than SZ is that our model allows for higher error bounds for noninteresting ranges, which can lead to higher compression ratios.

7 Conclusion and Future Work

In this paper we propose multiple novel error-bounded lossy compression methods that allow preserving various user-defined constraints; to the best of our knowledge, this is the first such lossy compression to allow these constraints. Based on our evaluation using real-world simulations, we report the following key findings.

- Multi-interval/region error-bound-based compression can significantly improve the visual quality for users with the same or even higher compression ratios.
- In the Nyx cosmology simulation, the multivalue-interval error-bounded lossy compression can preserve the halo cells perfectly with a high compression ratio up to 78, while the uniform error-bounded compression suffers significant distortion of cells.
- In the Hurricane Katrina simulation, multi-interval error-bounded compression can improve the compression ratio from 37 (based on SZ) to 80 (improved by 116%), even with higher data fidelity in maintaining the shape of hurricane.
- Evaluation for the bitmap-based solution shows that the cost to satisfying a customized complex region requirement is acceptable and our solution can possibly be generalized to suit all kinds of fine-grained error bound settings.
- Experiments on the Argonne Bebop [40] supercomputer with up to 2,100 cores show that our multi-precision lossy compressors have a very good scalability.

In the future we will explore new data fidelity requirements used by more applications in practice.

Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collabora-

tive 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 based upon work supported by the U.S. Department of Energy, Office of Science, by the DOE's Advanced Scientific Computing Research office, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709, OAC-2003624/2042084 and CSSI-2104023/2104024. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

-

REFERENCES

[1] J. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. Arblaster, S. Bates, G. Danabasoglu, J. Edwards *et al.*, "The community earth system model (CESM), large ensemble project: A community resource for studying climate change in the presence of internal climate variability," *Bulletin of the American Meteorological Society*, vol. 96, no. 8, pp. 1333–1349, 2015.

[2] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, "A methodology for evaluating the impact of data compression on climate simulation data," in *HPDC'14*, 2014, pp. 203–214.

[3] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, V. Vishwanath, T. Peterka, J. Insley *et al.*, "HACC: extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

[4] D. T. with Globus, https://www.globus.org/data-transfer, online.

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

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

[7] F. Cappello, S. Di, and et al., "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.

[8] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. NY, USA: Association for Computing Machinery, 2019.

[9] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Improving performance of iterative methods by lossy checkponting," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 52–65.

[10] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "DeepSZ: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '19. New York, NY, USA: ACM, 2019, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/3307681.3326608

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

[12] ——, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS)*. IEEE, 2017, pp. 1129–1139.

[13] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the univariate case," *Computing and Visualization in Science*, vol. 19, no. 5, pp. 65–76, Dec 2018.

[14] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2015, pp. 914–922.

[15] A. H. Baker, D. M. Hammerling, and T. L. Turton, "Evaluating image quality measures to assess the impact of lossy data compression applied to climate simulation data," *Computer Graphics Forum*, vol. 38, no. 3, pp. 517–528, 2019.

[16] NYX simulation, https://amrex-astro.github.io/Nyx, online.

[17] R. Underwood, S. Di, J. C. Calhoun, and F. Cappello, "FRaZ: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data," https://arxiv.org/abs/2001.06139, 2020, online.

[18] 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.

[19] Zstd, https://github.com/facebook/zstd/releases, online.

[20] Gzip, https://www.gzip.org/, online.

[21] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2008.

[22] Blosc compressor, http://blosc.org/, 2018, online.

[23] 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.

[24] C. S. Zender, "Bit grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF operators (NCO, v4.4.8+)," *Geoscientific Model Development*, vol. 9, no. 9, pp. 3199–3211, 2016.

[25] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "Data compression for the exascale computing era – survey," *Supercomputing Frontiers and Innovations*, vol. 1, no. 2, pp. 76–88, 2014.

[26] P. Ratanaworabhan, J. Ke, and M. Burtscher, "Fast lossless compression of scientific floating-point data," in *Data Compression Conference (DCC'06)*, IEEE. New York, NY, USA: IEEE, 2006, pp. 133–142.

[27] J. Diffenderfer, A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom, "Error analysis of ZFP compression for floating-point data," *SIAM Journal on Scientific Computing*, 02 2019.

[28] J. Zhang, X. Zhuo, A. Moon, H. Liu, and S. W. Son, "Efficient encoding and reconstruction of HPC datasets for checkpoint/restart," in *Proceedings of the 35th International Conference on Massive Storage Systems and Technology (IEEE MSST19)*, 2019.

[29] S. Di, D. Tao, X. Liang, and F. Cappello, "Sz tutorial hands-on guide," https://www.mcs.anl.gov/~shdi/download/sz-hands-on.pdf, 2018, online.

[30] X. Delaunay, A. Courtois, and F. Gouillon, "Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netcdf-4 or hdf5 files," *Geoscientific Model Development*, vol. 12, no. 9, pp. 4099–4113, 2019.

[31] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Fixed-PSNR lossy compression for scientific data," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 314–318.

[32] Hurricane ISABELA Simulation Datasets, http://vis.computer.org/vis2004contest/data.html.

[33] "Katrina simulation," https://adcirc.org/home/documentation/example-problems/katrina-run-2015-nws-20-example/, online.

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

[35] QMCPack, https://qmcpack.org/, online.

[36] R. T. Migration, http://www.seismiccity.com/RTM.html, online.

[37] Miranda, https://wci.llnl.gov/simulation/computer-codes/miranda.

[38] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. White, "The evolution of large-scale structure in a universe dominated by cold dark matter," *The Astrophysical Journal*, vol. 292, pp. 371–394, 1985.

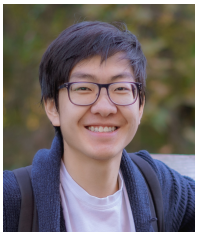
[39] B. Friesen, A. Almgren, Z. Lukić, G. Weber, D. Morozov, V. Beckner, and M. Day, "In situ and in-transit analysis of cosmological

simulations," *Computational Astrophysics and Cosmology*, vol. 3, no. 1, pp. 1–18, 2016.
[40] Bebop, https://www.lcrc.anl.gov/systems/resources/bebop, online.

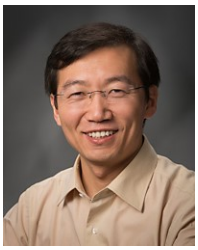
Yuanjian Liu is a Ph.D. student at the University of Chicago. His research interests include autonomous laboratories, computer vision, high-performance computing, and the utilization of scientific data. He is also working on the availability of knowledge and believes that online education can have both higher efficiency and broader influence. Email: yuanjian@uchicago.edu.

Sheng Di (Senior Member, IEEE) received his master's degree from Huazhong University of Science and Technology in 2007 and Ph.D. degree from the University of Hong Kong in 2011. He is currently a computer scientist at Argonne National Laboratory. His research interests involve resilience on high-performance computing (such as silent data corruption, optimization checkpoint model, and in situ data compression) and broad research topics on cloud computing. He is working on multiple HPC projects, such as detection of silent data corruption, characterization of failures and faults for HPC systems, and optimization of multilevel checkpoint models. He is the recipient of a DOE 2021 Early Career Research Program award. Email: sdi@anl.gov.

Kai Zhao received his bachelor's degree from Peking University in 2014 and will receive his Ph.D. degree from the University of California, Riverside, in 2022. He is a long-term intern at Argonne National Laboratory. His research interests include high-performance computing, scientific data management and reduction, and resilient machine learning. Email: kzhao016@ucr.edu.



Sian Jin is a Ph.D. candidate in computer science at Washington State University. He received his B.S. in physics from Beijing Normal University in 2018. His research interests include high-performance computing, scientific data analysis and management, data reduction and lossy compression, and large-scale machine learning. Email: sian.jin@wsu.edu.



Cheng Wang Cheng Wang is an environmental system engineer in the Environmental Science Division at Argonne National Laboratory. He received his Ph.D. in civil engineering from the University of Central Florida in 2009. His specialty areas are in the field of water resources engineering involved in hydrology, reactive biogeochemistry and transport, and other related disciplines. Specific examples of his expertise include numerical watershed modeling (e.g., WASH123D model); surface water hydrodynamic and sediment and reactive-chemical transport modeling; subsurface water hydrodynamic and reactive-chemical transport modeling; and coupled fluid flow and water quality modeling in surface and subsurface water systems. Email: wangcheng@anl.gov.

Kyle Chard is a research assistant professor in the Department of Computer Science at the University of Chicago. He also holds a joint appointment at Argonne National Laboratory. He received his Ph.D. in computer science from Victoria University of Wellington, New Zealand, in 2011. He is a member of the ACM and IEEE. He co-leads the Globus Labs research group, which focuses on a broad range of research problems in data-intensive computing and research data management. Email: chard@uchicago.edu.

Dingwen Tao is an assistant professor of computer science at Washington State University. He received his Ph.D. in computer science from the University of California, Riverside, in 2018 and B.S. in mathematics from the University of Science and Technology of China in 2013. His research interests include high-performance computing, big data analytics, scientific data management, fault tolerance and resilience, and large-scale machine learning. He is the recipient of a 2021 R&D 100 Award, 2020 IEEE-CS TCHPC Early Career Researchers Award for Excellence in HPC, 2020 NSF CRII Award, and 2017 UCR Dissertation Year Program Award. Email: dingwen.tao@wsu.edu

Franck Cappello is the director of the Joint-Laboratory on Extreme Scale Computing gathering six of the leading high-performance computing institutions in the world: Argonne National Laboratory, National Center for Scientific Applications, Inria, Barcelona Supercomputing Center, Julich Supercomputing Center, and Riken AICS. He is a senior computer scientist at Argonne National Laboratory and an adjunct associate professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He is an expert in resilience and fault tolerance for scientific computing and data analytics. Recently he started investigating lossy compression for scientific datasets to respond to the pressing needs of scientist performing large-scale simulations and experiments. His contribution to this domain is one of the best lossy compressors for scientific datasets respecting user-set error bounds. He is a member of the editorial board of the *IEEE Transactions on Parallel and Distributed Computing* and of the *ACM HPDC* and *IEEE CCGRID* steering committees. He is a fellow of the IEEE. Email: cappello@mcs.anl.gov.

Ian Foster is an Argonne Distinguished Fellow, senior scientist, and director of the Data Science and Learning division at Argonne National Laboratory and a professor in the Department of Computer Science at the University of Chicago. He develops tools and techniques that allow people to use high-performance computing technologies to do qualitatively new things. His work involves investigations of parallel and distributed languages, algorithms, and communication, as well as applications. He is particularly interested in using high-performance networking to incorporate remote compute and information resources into local computational environments. Email: foster@cs.uchicago.edu.