An Efficient Transformation Scheme for Lossy Data Compression with Point-wise Relative Error Bound

Xin Liang,* Sheng Di,† Dingwen Tao,‡ Zizhong Chen,* and Franck Cappello†§
* University of California, Riverside, CA, USA
† Argonne National Laboratory, Lemont, IL, USA
‡ The University of Alabama, AL, USA
§ University of Illinois at Urbana-Champaign, IL, USA
xlian007@ucr.edu, sdi1@anl.gov, tao@cs.ua.edu,
chen@cs.ucr.edu, cappello@mcs.anl.gov

***Abstract*—Because of the ever-increasing execution scale of scientific applications, how to store the extremely large volume of data efficiently is becoming a serious issue. A significant reduction of the scientific data size can effectively mitigate the I/O burden and save considerable storage space. Since lossless compressors suffer from limited compression ratios, error-controlled lossy compressors have been studied for years. Existing error-controlled lossy compressors, however, focus mainly on absolute error bounds, which cannot meet users' diverse demands such as pointwise relative error bounds. Although some of the state-of-the-art lossy compressors support pointwise relative error bound, the compression ratios are generally low because of the limitation in their designs and possible spiky data changes in local data regions. In this work, we propose a specialized, efficient approach to perform compression based on the pointwise relative error bound with higher compression ratios than existing solutions provide. Our contribution is threefold. (1) We leverage a specialized transformation scheme that can transfer the pointwise relative-error-bounded compression problem to an absolute-error-bounded compression issue. We also analyze the practical properties of our transformation scheme both theoretically and experimentally. (2) We implement the logarithmic transform in two of the most popular absolute-error-bounded lossy compressors, SZ and ZFP. (3) We evaluate our solution using multiple real-world application data across different scientific domains on a supercomputer with up to 4,096 cores and 12 TB of data. Experiments show that our solution achieves over $1.38X$ dumping and $1.31X$ loading performance over the second-best lossy compressor, respectively.**

I. INTRODUCTION

Today's scientific research is in badly need of an efficient data compressor due to the extremely large volumes of data produced in high performance computing (HPC) simulations and applications. Hardware/Hybrid Accelerated Cosmology Code (HACC) [1], for example, is a well-known cosmology simulation code that simulates the motion and interaction of up to 3.5 trillion particles. leading to petabytes of data to store. Some climate research models, such as Community Earth Simulation Model (CESM), need to run large ensembles of simulations, generating hundreds of terabytes data in tens of seconds. On the other hand, the improvement in parallel file systems (PFS) cannot catch up with the increasing need of I/O. Some I/O systems [2], for example, suffer from 8 GB/s of parallel writing performance with 32 burst buffers [3]. Therefore, considerably reducing the size of scientific data is critical to the significant improvement of I/O performance for extreme-scale scientific simulations.

Lossless compressors ([4], [5], [6]) can be used to alleviate the heavy I/O burden without any distortion of the data. However, they suffer from limited compression ratios (usually no more than 2:1 [7]) because of the random mantissas in scientific floating data. Error-bounded lossy compressors have been considered better alternatives under this circumstance, because they can significantly reduce the data size while keeping the distortion of data based on users' error-controlling demand, guaranteeing an effective post-analysis using the decompressed data for scientific application users.

In comparison with the absolute error bound that has been widely used to control the data distortion by existing state-of-the-art lossy compressors [8], [9], pointwise relative error bound is significant for many scientific applications. Unlike absolute-error-bounded compression (i.e., the difference between each decompressed data value and its corresponding original value must be bounded within a constant number), pointwise relative-error-bounded compression means that the compression error of each data point should be bounded within a constant percentage of its data value. That is, the smaller the data value, the lower the absolute error bound to be applied on the data point. Obviously, the pointwise relative error bound can preserve more details in regions with small values. On the other hand, some application users demand pointwise relative error bound based on the physical meaning of the simulation. According to cosmologists (such as the users and developers of HACC and NYX), for example, the higher a particle's velocity is, the larger the compression error it can tolerate.

Pointwise relative error bound is much tougher to deal with than absolute error bound according to the principles of lossy compressions. SZ [8], [10], for example, predicts the value for each data point based on the consecutiveness of the dataset and control the compression errors by a linear-scaling quantization method. Since the bin size is fixed for each data point, the compression errors can be bounded within a constant value (i.e., absolute error bound). However, such a design cannot adapt to the demand of pointwise relative error bound, which requires diverse error controls on different data points. In order to enable SZ to support point-wise relative

2168-9253/18/$31.00 ©2018 IEEE
DOI 10.1109/CLUSTER.2018.00036

error bound, a blockwise strategy was proposed [12]: the entire data is split into multiple non-overlapped data blocks each adopting an absolute-error-bounded compression where the error bound is calculated by using the minimal value in the block. This strategy may significantly degrade the compression ratio, especially when the minimal values in some blocks are far smaller than other data values in those blocks, because the absolute error bounds would be too small to get a high compression ratio. The developers of ZFP [9] proposed an approximate compression mode (called *precision*) to achieve a similar effect with pointwise relative error bound. However, it cannot fully respect the pointwise relative error bound (to be presented later) because the data transformation and embedded encoding adopted in the compression are designed for the optimization of rate distortion, that is, peak signal-to-noise ratio (PSNR) vs. bit-rate (the number of bits used to represent one data point on average).

In this paper, we search for a specialized transformation scheme that can convert a pointwise relative-error-bounded compression problem to an absolute-error-bounded compression problem. Specifically, we want to find a mapping function that can transform the original dataset to another data domain, such that the pointwise relative error bound in the original data domain is equivalent to the absolute error bound in the transformed domain. Then, one can adopt any state-of-the-art lossy compressors (such as SZ and ZFP) supporting absolute error bounds to perform the compression based on the user's pointwise relative error bounding demand. The context of lossy compression has its specific constraints on the mapping function: continuity, bijection, dealing with floating point representation accuracy limits instead of real numbers. Considering all these constraints, the logarithmic mapping is a suitable mapping function for this problem. This solution coincides with previous works using logarithmic change to measure relative error in economics and ocean observation [13], [14], [15], [16]. However, while the natural base logarithm is a good choice in their research context, we show that for state-of-the-art lossy compressors the base 2 is the most efficient base. In addition, although Campbell et al. [16] have adopted the relationship between relative error and logarithmic absolute error for statistical error measurement, we further formalize this transformation problem according to those constraints and show that the logarithmic mapping is the unique possible mapping for lossy compression.

The contributions of this paper are as follows.

- We formalize the transformation problem between absolute error bound and relative error bound in the context of lossy data compression mathematically. We also solve this problem and find the unique mapping function, which is consistent with using logarithmic change for error measurement.
- We investigate the impact of the selection of different logarithmic bases on the compression quality for SZ and ZFP, respectively. We prove that various bases lead to the similar compression results theoretically.
- We propose an efficient pointwise relative-error-bounded lossy compression algorithm by combining the logarithmic data transform scheme and state-of-the-art absolute-error-bounded compressors. Specifically, we integrate the transformation scheme into both SZ and ZFP.
- We perform the evaluation using multiple datasets produced by real-world large-scale HPC applications across different scientific domains. Experiments show that our solution can significantly improve the compression ratio based on a particular pointwise relative error bound, compared with four other state-of-the-art lossy compressors with comparable compression and decompression rates. Parallel experiments with up to 4,096 cores show that both the overall data dumping performance (i.e., compression + data writing) and data loading performance (i.e., data reading + decompression) can be improved by 38% and 31%, respectively, compared with the second-best compressor.

The reminder of this paper is organized as follows. In Section II, We discuss the related work. In Section III, we derive an efficient transformation scheme based on the logarithmic function for pointwise relative error bound and analyze its significant properties theoretically. In Section IV, we investigate the impact of the logarithmic base selection on the compression ratios, under our designed transformation scheme with SZ and ZFP, respectively. In Section V, we present how to support pointwise relative error bound by combining our data compression scheme and the existing absolute-error-bounded lossy compressors. In Section VI, we present the experimental results based on multiple real-world scientific HPC application data across different domains with up to 4,096 cores. in Section VII, we conclude our work and briefly look at future work .

II. Related Work

Many of the existing lossy compressors, such as Jpeg [17] and Jpeg2000 [18], cannot do the compression based on a user-defined error bound, which, however, is strictly required by many scientific applications. Error-controlled lossy compression is proposed in order to significantly reduce the size of data while keeping the distortion of the decompressed data to a tolerable range. However, such compressors are usually designed and optimized on maximizing the compression ratio given absolute error bounds or PSNR. In this section, we discuss the state-of-the-art lossy compressors in terms of pointwise relative error bound. We will present our transformation scheme in the next section and analyze the impact of our approach on the two lossy compressors thereafter.

Although many state-of-the-art lossy compressors provide users with pointwise relative error bounds, they are not designed as efficiently as absolute error bounds. ISABELA [19] suffers from large computational overhead on sorting as well as large storage overhead on the sorting index, yielding a low compression rate and ratio. The precision mode in ZFP [9] cannot strictly respect error for certain portions of data, and hence it usually does not meet the requirements of scientific analysis. The PWR mode in SZ [8], [12] has slightly higher

performance, but it cannot achieve a high compression ratio because it divides the whole dataset into small blocks and conservatively sets the pointwise relative error bound according to the data with the smallest value in each block. FPZIP [20] seems the best among existing compressors. However, it accepts only precision as a parameter, which makes it hard for users to select or tune this parameter given an error bound. Moreover, the compression ratio exhibits piecewise features over error bounds, which may not be able to maximize the compression ratio in a given error bound.

In this paper, we propose to use a specialized transformation scheme that can transform a state-of-the-art absolute-error-bounded lossy compressors to a state-of-the-art pointwise relative-error-bounded lossy compressor. This transformation scheme is generic and can work as a preprocessing stage and a postprocessing stage for any lossy compressor theoretically. In our experiments, it successfully turns two state-of-the-art lossy compressors, SZ and ZFP, representative of prediction-based compressors and transform-based compressors, to pointwise relative-error-bounded lossy compressors that are better than their own pointwise relative-error-bounded mode. Furthermore, since SZ is a top lossy compressor in terms of absolute error bound, this scheme turns it into a very good pointwise relative-error-bounded lossy compressor that exhibits much more efficiency than the existing poinywise relative-error-bounded lossy compressors.

III. An Efficient Transformation Scheme for Pointwise Relative Error Bound

In this section, we leverage a specialized data transformation scheme that can convert the pointwise relative-error-bounded lossy compression problem to an absolute-error-bounded lossy compression problem. That is, under our designed mapping scheme, any absolute-error-bounded lossy compressor can be enabled to support pointwise relative error bound such that more details can be preserved in the regions with small data values. In what follows, we first derive an efficient data mapping/transformation scheme and then prove that the mapping method is the unique solution to the data conversion between the absolute error bound and pointwise relative error bound. We also analyze how to adjust the absolute error bound for the lossy compression to strictly respect the pointwise relative error bound, considering the impact of the possible round-off errors raised during the mapping.

A. Theories of the Transformation Scheme

The research problem can be formulated as follows: find a bijective and continuous function (denoted by f) that can map the original dataset (denoted by x) to another domain $f(x)$ such that the pointwise relative error bound (denoted by b_r) can be mapped to an absolute error bound (denoted by b_a) using another mapping function g (i.e., $b_a = g(b_r)$). Note that the mapping function (g) used to convert the error bound is different from the mapping function (f) used to transform the data, which will be discussed later in more detail. Moreover, the mapping function f should be bijective because we need to map the data reconstructed by the absolute-error-bounded compressor back to the original data domain (i.e., x=$f^{-1}(f(x))$) during the decompression. It also needs to be continuous because otherwise the mapping function may affect the continuity of the original data, degrading the data prediction accuracy in turn. With such a continuous bijective mapping function, the original data can be mapped to another domain and then compressed by the corresponding absolute error bound b_a. In the decompression phase, the data will be mapped back to the original domain via the corresponding inverse function, and the pointwise relative error bound will automatically hold. We derive Theorem 1 in order to search for the most effective mapping function.

Theorem 1: Given a dataset whose data values are denoted by x, Equation (1) is a sufficient condition of transforming it to another data domain by a mapping function f, such that if the transformed data are compressed with an absolute error bound $g(b_r)$, the corresponding inverse mapping of the decompressed transformed data will always be bounded by the pointwise relative error bound b_r in the original domain.

$$\frac{f^{-1}(f(x)+g(b_r))-x}{x} = b_r \tag{1}$$

Proof: The pointwise relative-error-bounded compression regarding the mapping function can be formalized as

$$\frac{|f^{-1}(f(x)+\epsilon)-x|}{|x|} \leq b_r \tag{2}$$

where ϵ∈$[-g(b_r),\ g(b_r)]$ refers to compression error, f is the target forward-mapping function to be applied before the compression, and f^{-1} refers to the inverse function to be applied after the decompression.

Since f and f^{-1} are bijective and continuous, they must be strictly monotonic. Without loss of generality, we denote x as a positive value, and $f(x)$ is a monotonically increasing function (in fact, if $x < 0$, it can be mapped to $-x$ first and then we can derive the same result by the following derivation). Since f is a monotonic function and $x > 0$, we have $f^{-1}(f(x)+\epsilon)$ must be always in interval $[f^{-1}(f(x)-g(b_r)),\ f^{-1}(f(x)+g(b_r))]$.

In order to reach the maximum compression ratio, the absolute compression error ϵ and the pointwise relative compression error are expected to be equal to their bounds (i.e., $g(b_r)$ and b_r) at the same time. This leads to $\frac{f^{-1}(f(x)+g(b_r))-x}{x} = b_r$ and $\frac{f^{-1}(f(x)-g(b_r))-x}{x} = -b_r$. Although g is defined only in [0, +∞) because of the pointwise relative error bound $b_r \geq 0$, we can merge the two equations to Formula (1) if we introduce a definition $g(x) = -g(-x)(\forall x < 0)$ without loss of generality. ■

This theorem indicates that the pointwise relative error bound can be transformed to an absolute error bound as long as Equation (1) holds for the data-mapping function f and the error-bound-mapping function g. In what follows, we derive a theorem (Theorem (2)) to find the corresponding functions.

Before proposing the theorem, we recall a critical lemma based on the theory of functional equations.

Lemma 1: The exponential function is the only continuous and nonconstant function that satisfies $f(x+y) = f(x)f(y)$, where x and y are both real numbers.

Theorem 2: $f(x) = \log_{base} x + C$ is the unique mapping function that satisfies the Equation (1), where *base* > 1 and $C = f(1) \in R$. In this situation, the mapping function g of the absolute error bound is $b_a = g(b_r) = \log_{base}(1 + b_r)$.

Proof: We rewrite Equation (1) and apply f to both sides

$$f(x) + g(b_r) = f((1 + b_r)x).$$

Let us set $x = 1$. Then the function g can be represented as $g(b_r) = f(1 + b_r) - f(1)$. Let $y = 1 + b_r$, and substitute g in the above equation. Then we have

$$f(x) + f(y) - f(1) = f(xy)$$

Let $h(x) = f(x) - f(1)$, then we can derive $h^{-1}(x) = f^{-1}(x + f(1))$. According to the above equation, we can get:

$$h(x)+h(y) = f(x)+f(y)-2f(1) = f(xy)-f(1) = h(xy).$$

Applying h^{-1} to the left-most and right-most sides of the above equation, we get the following equation:

$$h^{-1}(h(x) + h(y)) = xy = h^{-1}(h(x))h^{-1}(h(y)).$$

Let $x' = h(x)$ and $y' = h(y)$. The equation turns out to be $h^{-1}(x' + y') = h^{-1}(x')h^{-1}(y')$. Denoting $h^{-1}(1) = base$ leads to $h^{-1}(x) = base^x$ according to Lemma 1. Thus, $h(x) = \log_{base} x$. Let $f(1) = C$. We have $f(x) = h(x) + f(1) = \log_{base} x + C$. Accordingly, $b_a = g(b_r) = f(1 + b_r) - f(1) = \log_{base}(1 + b_r)$. ∎

Without loss of generality, C can be set to 0 because of two factors. On the one hand, C just adds a static shift to the mapped data, which means that it has no effect on the prediction accuracy of the Lorenzo predictor. On the other hand, more perturbations may be introduced to the result because of round-off errors if C is nonzero. Therefore we fix $C = 0$ in our implementation.

As previously mentioned, the logarithmic function is used for relative error measurement in some scientific domains such as economics and ocean observations. The theory presented in this paper, in addition, further indicates that the logarithmic function family is the unique mapping function for transformation between relative error and absolute error in the context of lossy data compression.

B. Round-off Error

Because of the inexact representation of floating-point arithmetic, we cannot produce exact calculation results. Thus, there is also round-off error while applying the mapping function f and f^{-1}. When this error is taken into consideration, the pointwise relative error bound may no longer be respected. In this subsection we analyze how to control this error.

Now that we have our mapping function $f(x) = \log_a x$ and its reverse $f^{-1}(x) = a^x$. To respect the point-wise error bound, we have the following lemma.

Lemma 2: The absolute error derived in Theorem 2 should be adjusted to $b'_a = log_a(1 + b_r) - \max_x |\log_a x|\varepsilon_0$ to respect the pointwise relative error bound considering round-off errors, where $\max_x |\log_a x|$ is the maximum absolute value of the mapped data $\log_a x$ and ε_0 is the round-off error introduced to $f(x)$ because of machine precision.

Proof: For any data point x, its decompressed value x_d will be

$$x_d = f^{-1}(f(x)(1+\varepsilon_0)+\epsilon) = a^{(1+\varepsilon_0)\log_a x+\epsilon} = xa^{\varepsilon_0 \log_a x+\epsilon}$$

where $\epsilon \in [-b'_a, b'_a]$ is the compression error of data in the transformed domain (i.e., logarithmic data domain). Similar to the analysis above, in order to respect the error bound, the absolute error bound b'_a should be set as follows: $a^{\varepsilon_0 \log_a x+b'_a} \leq 1+b_r$ for any data point x. Therefore, we have $b'_a = g'(b_r) = \log_a(1 + b_r) - \max_x |log_a x|\varepsilon_0$. ∎

We set ε_0 to machine epsilon in our implementation. This setting can already have all of the data points strictly bounded within the specified error bounds during the compression in our evaluation, as will be presented later.

IV. Impact of base selection

In this section, we investigate the impact of base selection for the logarithmic mapping solution in Theorem 2. Specifically, we prove that different logarithmic bases lead to similar compression results; therefore, simply changing the logarithmic base cannot improve the compression quality.

A. Impact of Base Selection on SZ

1) Review of SZ: To understand the impact of base selection on SZ, we need to review the principle of the SZ compressor. SZ [8] is a prediction-based lossy compressor. It consists of three stages during the compression. In the stage I, it predicts each data value in the dataset according to some deterministic prediction model. Then, it applies a linear-scaling quantization on the prediction errors such that all the floating-point values could be converted to a set of integer quantization codes. In stage II, SZ adopts a customized Huffman encoder constructed in terms of the quantization codes and performs the compression. Stage III applies GZIP to the encoded data to further improve the compression ratio. This step is optional depending on how hard the data is to be compressed and the error bound specified. During the decompression, SZ first decompresses the data by GZIP and the Huffman encoder and then recovers the data by the prediction model with the decoded prediction errors. For the data prediction, SZ adopts the Lorenzo predictor [21] with a limited number of neighbors[1] by default since more neighbors may cause degraded prediction accuracy because of the impact of the decompressed values. [2]

2) Logarithmic Base Analysis: We notice that the selection of different bases for the log mapping function does not affect clearly the compression quality of SZ. In absolute terms, we have the following lemma.

Lemma 3: For SZ lossy compressor with Lorenzo predictor and linear-scaling quantization, different bases will lead to the same prediction accuracy, if the arithmetic operations lead to the exact results.

[1]For the data prediction of SZ, there is 1 neighbor used per data point in 1D dataset, 3 neighbors per data point in a 2D dataset, and 7 neighbors per data point in a 3D dataset.

[2]Note that during the compression phase, the neighboring data used by the predictor have to be the decompressed values in order to avoid error propagation during the decompression phase.

Proof: Suppose a mapping function $f(x) = \log_a x$ is used to compress the same 1D dataset. Then the quantization index i should satisfy Equation (3), when predicting the next point x_1 by the previous data point x_0.

$$\log_a x_1 = \log_a x_0 + q \log_a (1 + b_r) \tag{3}$$

Thus, q can be solved as follows.

$$q = \frac{\log_a x_1 - \log_a x_0}{\log_a (1 + b_r)} = \frac{\log_a \frac{x_1}{x_0}}{\log_a (1 + b_r)} = \log_{1+b_r} \frac{x_1}{x_0}$$

Therefore, the quantization index is independent of the mapping base a if the arithmetic operations lead to exact results. This also holds for 2D and 3D Lorenzo predictors since the quantization index can be computed as follows.

$$q_{2D} = \frac{\log_a x_{01} + \log_a x_{10} - \log_a x_{00} - \log_a x_{11}}{\log_a (1+b_r)} = \log_{1+b_r} \frac{x_{01}x_{10}}{x_{00}x_{11}}$$

$$\begin{aligned} q_{3D} &= \frac{\log_a x_{001} + \log_a x_{010} + \log_a x_{100} + \log_a x_{111}}{\log_a (1+b_r)} \\ &\quad - \frac{\log_a x_{000} + \log_a x_{011} + \log_a x_{101} + \log_a x_{110}}{\log_a (1+b_r)} \\ &= \log_{1+b_r} \frac{x_{001}x_{010}x_{100}x_{111}}{x_{000}x_{011}x_{101}x_{110}} \end{aligned}$$

■

However, the floating-point arithmetic operations may cause nonexact results because of round-off errors, introducing certain deviation to the distribution of quantization index codes. Fortunately, we can derive a strict bound for the quantization index with different bases as follows.

Theorem 3: Given two coding integer values (q_0 and q_1) at ith quantization index bin based on two different bases, the difference in between is bounded by $|\log_{1+b_r}(1-b_r) - 1|$, $3|\log_{1+b_r}(1-b_r) - 1|$, $7|\log_{1+b_r}(1-b_r) - 1|$ for the 1D, 2D, and 3D Lorenzo predictions, respectively.

Proof: Let us start with the 1D case. Without loss of generality, assume that the decompressed data of two bases are x_{1d} and $x_{1d'}$. Then the difference of their quantization indices q_0 and q_1 can be derived according to Lemma 3.

$$q_0 - q_1 = \log_{1+b_r} \frac{x_1}{x_{0d}} - \log_{1+b_r} \frac{x_1}{x_{0d'}} = \log_{1+b_r} \frac{x_{0d'}}{x_{0d}}$$

Since $(1-b_r)x_0 \le x_{0d} \le (1+b_r)x_0$ and $(1-b_r)x_0 \le x_{0d'} \le (1+b_r)x_0$, we have

$$\log_{1+b_r} \frac{(1-b_r)x_0}{(1+b_r)x_0} \le q_0 - q_1 \le \log_{1+b_r} \frac{(1+b_r)x_0}{(1-b_r)x_0}.$$

It can be simplified to

$$\log_{1+b_r}(1-b_r) - 1 \le q_0 - q_1 \le 1 - \log_{1+b_r}(1-b_r).$$

As for the 2D and 3D Lorenzo predictions, the difference involves 3 and 7 multiplication of decompressed data. Thus they are bounded by

$$\log_{1+b_r}(\frac{1-b_r}{1+b_r})^3 \le q_0 - q_1 \le \log_{1+b_r}(\frac{1+b_r}{1-b_r})^3$$

$$\log_{1+b_r}(\frac{1-b_r}{1+b_r})^7 \le q_0 - q_1 \le \log_{1+b_r}(\frac{1+b_r}{1-b_r})^7,$$

which corresponds to the given bounds in the theorem. ■

B. Impact of Base Selection on ZFP

1) Review of ZFP: Similarly, we first go over how ZFP works as an error-bounded lossy compressor and the metrics to assess the effectiveness of its orthogonal transform. ZFP [9] adopts a largely different method to compress a floating-point dataset compared with SZ. Briefly, it involves two critical steps: an invertible blockwise transform on the input data and an embedded encoding for the transformed coefficients. Specifically, it divides the whole dataset into independent blocks, aligns the exponents, and turns the floating-point representations into fixed-point representations. Then, ZFP applies a data transform in each block. Finally, it performs an embedded encoding on the transformed data (or transformed coefficients). Such a design obtains an optimized rate distortion (i.e., PSNR vs. bit-rate), although it may not maximize the compression ratio given an error bound because it conservatively overestimates the errors in the maximum bit-plane computation for the purpose of strictly respecting the error bound.

The transform is the most critical part in ZFP compression, and it is the key factor of the final compression quality. Its effectiveness can be determined by two metrics, *decorrelation efficiency* and *coding gain*, as introduced in [9], [22]. Therefore, we analyze the impact of the logarithmic base selection according to the two metrics, defined as follows.

Definition 1: Considering that each entry in a block is a random variable, the decorrelation efficiency η and coding gain γ can be defined by

$$\eta = \frac{\sum_i \sigma_{ii}^2}{\sum_i \sum_j \sigma_{ij}^2} \quad \gamma = \frac{\sum_i \sigma_{ii}^2}{n(\prod_i \sigma_{ii}^2)^{\frac{1}{n}}}, \tag{4}$$

where σ_{ij} is the element in the ith row and jth column of the covariance matrix of transformed coefficients and n is the number of elements in a block.

2) Logarithmic Base Anaylsis: With these two metrics, we can analyze the impact of different logarithmic bases on the quality of ZFP compression. We have the following lemma.

Lemma 4: Decorrelation efficiency η and coding gain γ will be unchanged across different logarithmic bases.

Proof: For simplicity, we only discuss 1D cases. The multi-dimensional cases can be deducted accordingly. To make the analysis more general, we use A to denote any generic transform matrix.

Denote $X = (X_1, \ldots, X_n)^T$ as random variables of origin data in each block, $V = (V_1, \ldots, V_n)^T$ as the random variables of coefficients. The data variable will turn out to be $Y = \log_a X = \frac{\ln X}{\ln a}$ after the logarithmic transformation. Correspondingly, the coefficients variable will be $V = AY$. More specifically, let us denote $A = (A_1, \ldots, A_n)^T$, where A_i is the ith row vector of A. Then the covariance between any of two variables can be computed by

$$\begin{aligned} \sigma_{ij} &= cov(V_i, V_j) = E(V_i V_j) - E(V_i)E(V_j) \\ &= E(Y^T A_i^T A_j Y) - E(Y) A_i^T A_j^T E(Y) \\ &= E((\frac{\ln X}{\ln a})^T A_i^T A_j \frac{\ln X}{\ln a}) - E((\frac{\ln X}{\ln a})^T) A_i^T A_j E(\frac{\ln X}{\ln a}) \\ &= \frac{1}{(\ln a)^2}[E((\ln X)^T A_i^T A_j \ln X) - E(\ln X)^T A_i^T A_j E(\ln X)]. \end{aligned}$$

Thus, the logarithmic base a serves only as a multiplicand factor. During the computation of η and γ as shown in Equation (4), $\frac{1}{(\ln a)^2}$ can be extracted as a common factor of both numerator and denominator and thus canceled. Therefore, the base selection will not affect the decorrelation efficiency and coding gain of ZFP. ■

As such, we have proved that the ZFP transform will work equally well on different logarithmic bases. In fact, different logarithmic bases just apply different multiplicands to the transformed coefficients. Under these circumstances, the compression quality of ZFP will hardly change, as will be validated in Section VI.

V. Implementation

In this section, we discuss how to implement the pointwise relative error bound by combining the logarithmic data transform scheme and the existing state-of-the-art absolute-error-bounded lossy compressors.

The pseudo-code of the logarithmic-mapping-based lossy compression in terms of pointwise relative error bound is presented as Algorithm 1.

Algorithm 1 Logarithmic-mapping based lossy compression for point-wise relative error bound

Input: a dataset (denoted by D), user-specified point-wise relative error bound b_r
Output: compressed data stream in form of bytes

1: $b'_a = log_a(1+b_r) - \max_x |\log_a x|\varepsilon_0$; /*Calculate the absolute error bound in the transformed data domain*/
2: signs[$||D||$] = $\{0\}$, P = 1
3: **for** (each data point D_i in the dataset D) **do**
4: **if** (D_i == 0) **then**
5: d_i = $\log_2(\min) - 2b'_a$, where min refers to the minimum floating-point value; /*set d_i to be two error bounds less than the lower-bound exponent of the data value range*/
6: **else**
7: **if** ($D_i > 0$) **then**
8: Compute d_i = $\log_2(D_i)$; /*Perform the data mapping*/
9: **else**
10: Compute d_i = $\log_2(-D_i)$;
11: P = 0, sign[i] = 1;
12: **end if**
13: **end if**
14: **end for**
15: **if** (P==0) **then**
16: compress signs with gzip
17: **end if**
18: Perform the compression of the transformed dataset $\{d_i\}$ by a lossy compressor (such as SZ or ZFP) using the absolute error bound b'_a;
19: Output the compressed data stream in bytes;

In the algorithm, we first calculate the required absolute error bound (denoted by b'_a) based on the given pointwise relative error bound b_r, with a consideration of the possible round-off errors to be introduced during the mapping operation due to the machine epsilon (line 1), according to Lemma 2. Then, the algorithm performs the data transformation based on the logarithmic function (line 2∼8). If the original data point's value is equal to 0, we will map it to the lower-bound exponent of the floating-point data value range minus $2b'_a$. Specifically, the minimal positive values of a single-precision number and a double-precision number are 2^{-2^7}=2^{-127} and $2^{-2^{11}}$=2^{-1024}, respectively, because their IEEE 754 representations [11] adopt 1+7 bits and 1+11 bits to represent a signed exponent, respectively. In the decompression phase, the data values decompressed by the absolute-error-bounded compressor in the transformed domain will be back-transformed to 0, as long as their reconstructed values are lower than or equal to the minimal positive values minus b'_a. Such a design can ensure that almost all the zero-value data points will be decompressed to an exact zero number, unlike the SZ 1.4 that may reconstruct the zeros to be close-to-zero numbers approximately. After the data transformation step (line 3∼8), the algorithm will perform the absolute-error-bounded lossy compression over the transformed dataset, by leveraging an existing compressor such as SZ and ZFP (line 19).

VI. Evaluation

In this section, we compare our approaches with four state-of-the-art lossy compressors providing pointwise relative error bounds: ISABELA [19], SZ PW_REL mode (denoted SZ_PWR) [8], [12], ZFP precision mode (denoted ZFP_P) [9], and FPZIP [20]. We also demonstrate the effectiveness of pointwise relative error from the perspective of visual quality by comparing it with the absolute-error-bound mode in SZ (SZ_ABS). To distinguish from the existing SZ and ZFP compressors, we name our approaches SZ_T and ZFP_T for SZ and ZFP with our transformation scheme, respectively.

A. Experiment Setup

We conduct our experimental evaluations on a supercomputer [23] using up to 4,096 cores (i.e., 128 nodes, each with two Intel Xeon E5-2695 v4 processors and 128 GB of memory, and each processor with 16 cores). The storage system uses General Parallel File Systems (GPFS). These file systems are located on a raid array and served by multiple file servers. The I/O and storage systems are typical high-end supercomputer facilities. We use the file-per-process mode with POSIX I/O [24] on each process for reading/writing data in parallel. [1] The HPC application data are from multiple domains, namely, HACC cosmology simulation [1], CESM-ATM climate simulation [27], NYX cosmology simulation [28], and Hurricane ISABEL simulation [29]. Each application involves many simulation snapshots (or time steps). We assess only meaningful fields with relatively large data sizes (other fields have constant data or too small data sizes). Table I presents all the 101 fields across these simulations. The data sizes per snapshot are 3.1 GB, 1.9 GB, 1.2 GB, and 3 GB for the four applications, respectively.

TABLE I
Simulation fields used in the evaluation

Application	# Fields	Dimensions	Examples
HACC	3	280953867	velocity_x, velocity_y, velocity_z
CESM-ATM	79	1800×3600	CLDHGH, CLDLOW ···
NYX	6	512×512×512	dark_matter_density, temperature ···
Hurricane	13	100×500×500	CLOUDf48, Uf48 ···

[1] POSIX I/O performance is close to other parallel I/O performance such as MPI-IO [25] when thousands of files are written/read simultaneously on GPFS, as indicated by a recent study [26].

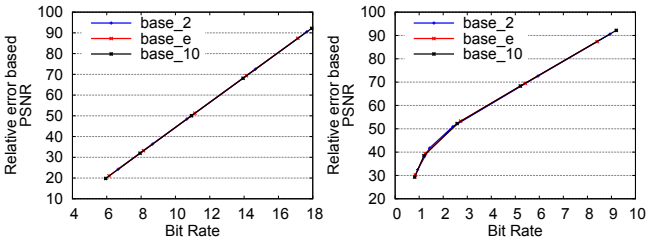
B. Impact of Base Selection

We choose two representative fields, dark_matter_density and velocity_x, in NYX to demonstrate the influence of different logarithmic bases on the final result. Dark_matter_density is a typical use case for pointwise relative error. A large majority (84%) of its data is distributed in [0, 1], and the rest is distributed in [1, 1.378E+4]. Simply applying the absolute value will result in huge distortion when users need to focus on the densest data in [0, 1]. Velocity_x, on the other hand, has usually large values with positive/negative signs indicating directions. Pointwise relative error is also needed when accurate directions of the 3D velocity are required for each point. We evaluate 3 most widely used logarithmic bases: 2, e and 10 on both SZ and ZFP on these two fields.

TABLE II
COMPRESSION RATIO OF DIFFERENT BASES FOR SZ_T ON 2 FIELDS IN NYX

fields	dark_matter_density			velocity_x		
log_bases	2	e	10	2	e	10
0.0001	2.033	2.036	2.036	4.235	4.202	4.254
0.001	2.724	2.725	2.585	7.647	7.509	7.482
0.01	3.842	3.843	3.847	13.047	13.115	13.131
0.1	6.298	6.249	6.307	20.788	18.171	20.079
0.2	7.619	7.595	7.562	22.623	23.090	24.635
0.3	8.529	8.427	8.541	29.696	28.799	29.361

Table II shows the compression ratio of SZ with six different pointwise relative error bounds ranging from 10^{-4} to 0.3. According to the table, the different logarithmic bases impact only 1% and 3% on the final compression ratio on average for the two fields, respectively. This variance is less when the pointwise relative error bound is small because of larger number of quantization intervals and tighter bound (Theorem 3) on the difference of quantization index at that time, resulting in a low ratio in frequency difference of the Huffman tree. When the pointwise relative error bound grows, the variance becomes slightly larger because of the smaller number of quantization intervals and the looser bound.



(a) dark_matter_density (b) velocity_x

Fig. 1. Rate distortion of different bases for ZFP_T on 2 fields in NYX

As mentioned, we prove that different bases do not affect the decorrelation efficiency and coding gain of ZFP. However, since ZFP aims at optimizing the rate distortion given an absolute error bound, it may not keep the same compression ratio because of the different maximum bit-plane computed in embedded coding. Thus, we show the point-wise relative error based rate distortion of ZFP_T in Figure 1, in which the PSNR is calculated based on point-wise relative errors with the value range being set to 1. From this figure, we can see that the different bases make little difference in terms of pointwise rate distortion, which is a result that is consistent with our analysis.

TABLE III
PERFORMANCE OVERHEAD OF DIFFERENT BASES ON 2 FIELDS IN NYX

fields	dark_matter_density			velocity_x		
log_bases	2	e	10	2	e	10
pre-processing time(s)	1.67	1.59	2.23	2.18	2.08	2.74
post-processing time(s)	1.73	2.30	7.11	2.04	2.52	7.35

Table III shows the overhead of preprocessing and postprocessing steps in our transformation scheme under difference bases. The overhead in preprocessing comes from two aspects: logarithmic mapping on the original dataset and lossless compression on the signs if the dataset is not always positive or negative. Correspondingly, the postprocessing step decompresses the signs if necessary and performs the inverse mapping. Field dark_matter_density has faster preprocessing and postprocessing time compared with velocity_x because its data are always positive and omit the sign compression. Base 10 performs badly during postprocessing because it does not have fast implementation in a standard C library such as base 2 (exp2) and base e (exp). Also it is not competitive on preprocessing, so we do not use it. Although base e is faster than base 2 while preprocessing, it is much slower during the postprocessing step. Thus we use logarithmic base 2 in our implementation, and we fix it for both SZ_T and ZFP_T in the rest part of evaluation.

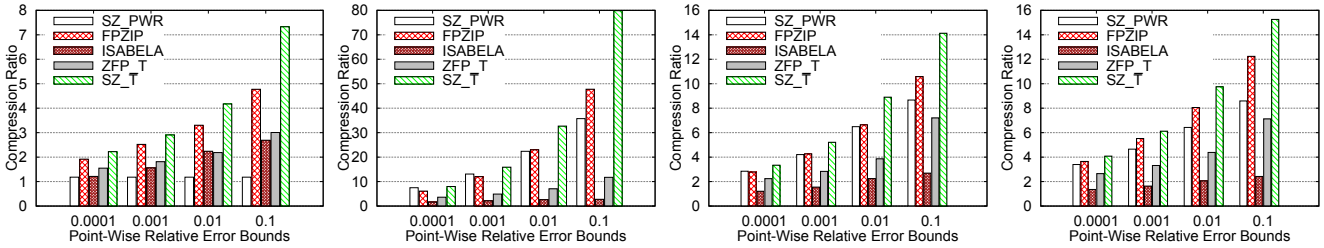
C. Strict Error Bound Test

In this subsection, we check the maximum pointwise relative errors of our approach and state-of-the-art approaches. As mentioned, ZFP overpreserves the absolute error bound and thus may not be competitive with other compressors given an absolute error bound. Therefore, we select SZ_T as our final solution to maximize the compression ratio, given the pointwise relative error bound. However, we still compare ZFP_T with the precision mode of ZFP to demonstrate that our transformation scheme can also be used to improved transform-based compressors such as ZFP.

The results of three most widely used pointwise error bounds (0.1, 0.01, 0,001) are shown in Table IV. Columns 4 and 9 (settings) indicate the parameters we choose for each compressor. Columns 5 and 10 show the percentage of decompressed data that is strictly bounded by the given error bound. The notation $\approx$ indicates that most of the data is bounded by the error bound, but there exists little data (usually much less than 0.01%) that exceeds the bound, likely because of round-off errors. The notation * in the table indicates that the compressor modifies original 0 in the data. The compressors without the notation * keep the original 0 as it is, such that the decompression has no loss on the values 0. The columns *Avg E* and *Max E* indicate the average and maximum pointwise relative errors, respectively. From this table, one can clearly see that only FPZIP and the compressors under our transformation scheme (SZ_T and ZFP_T) can strictly respect the given error bound and keep the original zeros as they are. Furthermore, the SZ_T compressor also yields the best

TABLE IV
POINTWISE RELATIVE ERROR BOUND ON 2 REPRESENTATIVE FIELDS IN NYX

			dark_matter_density					velocity_x				
pwr_eb	**type**	**name**	**settings**	**bounded**	**Avg E**	**Max E**	**CR**	**settings**	**bounded**	**Avg E**	**Max E**	**CR**
1E-3	prediction	ISABELA	1E-3	≈ 100%	4.6E-4	≈ 1E-3	1.35	1E-3	≈ 100%	4.7E-4	≈ 1E-3	1.71
		FPZIP	-p 19	100%	3.4E-4	9.8E-4	2.28	-p 19	100%	3.5E-4	9.8E-4	6.15
		SZ_PWR	-P 1E-3	≈ 100%*	1.2E-4	≈ 1E-3	1.87	-P 1E-3	≈ 100%	4.5E-4	≈ 1E-3	6.77
		SZ_T	-P 1E-3	100%	4.7E-4	9.9E-4	2.72	-P 1E-3	100%	4.9E-4	9.9E-4	7.58
	transform	ZFP_P	-p 26	99.93%*	5.7E-5	2.9E+4	1.5	-p 20	99.94%	2.3E-5	1.5E+2	3.4
		ZFP_T	-p 1E-3	100%	2.2E-5	2.1E-4	1.81	-p 1E-3	100%	2.3E-5	2.1E-4	3.58
1E-2	prediction	ISABELA	1E-2	≈ 100%	4E-3	≈ 1E-2	1.91	1E-2	≈ 100%	2.9E-3	≈ 1E-2	2.42
		FPZIP	-p 16	100%	2.7E-3	7.8E-3	2.89	-p 16	100%	2.8E-3	7.8E-3	10.79
		SZ_PWR	-P 1E-2	≈ 100%*	1.2E-3	≈ 1E-2	2.46	-P 1E-2	≈ 100%	4.5E-3	≈ 1E-2	11.08
		SZ_T	-P 1E-2	100%	4.8E-3	1E-2	3.85	-P 1E-2	100%	5E-3	1E-2	13.49
	transform	ZFP_P	-p 23	99.94%*	5.7E-4	2.5E+5	1.75	-p 16	99.91%	3.4E-4	7.7E+2	5.42
		ZFP_T	-p 1E-2	100%	1.8E-4	1.6E-3	2.18	-p 1E-2	100%	1.8E-4	1.6E-3	5.38
1E-1	prediction	ISABELA	1E-1	≈ 100%	1.8E-2	≈ 1E-1	2.52	1E-1	100%	7.6E-3	1E-1	2.75
		FPZIP	-p 13	100%	2.2E-2	5.9E-2	3.97	-p 13	100%	2.2E-2	5.9E-2	19.08
		SZ_PWR	-P 1E-1	≈ 100%*	1.2E-2	≈ 1E-1	3.37	-P 1E-1	100%	4.5E-2	1E-1	13.73
		SZ_T	-P 1E-1	100%	4.6E-2	1E-1	6.31	-P 1E-2	100%	4.8E-2	1E-1	22.07
	transform	ZFP_P	-p 19	99.91%*	5.7E-3	1.9E+5	2.23	-p 13	99.93%	2.6E-3	2E+5	12.1
		ZFP_T	-p 1E-1	100%	2.8E-3	2.6E-2	3.00	-p 1E-1	100%	2.1E-3	2.5E-2	13.3



(a) HACC (b) CESM-ATM (c) NYX (d) Hurricane

Fig. 2. Compression ratio on given point relative error bound

compression ratio on the two fields, demonstrating its high quality and good efficiency.

We also compare ZFP under our transformation scheme (denoted ZFP_T) with the -p option given by the original ZFP lossy compressor (denoted ZFP_P). Since ZFP_P does not strictly respect the error bound, we set the percentage threshold for bounded data in ZFP_P to 99.9%, in order to keep the same order of average error with ZFP_T. According to the table, ZFP_T outperforms ZFP_P in almost all aspects, demonstrating that our transformation scheme can really improve the compression quality for ZFP as well. Its compression ratio is not as high as those of other compressors because of the over-preserved error bound. If there exist some other transform-based compressors that may lead to a very high compression ratio given specific absolute error bounds, our transformation scheme can also turn them into outstanding compressors respecting pointwise relative error bound.

D. Compression Ratio & Compression/Decompression Rate

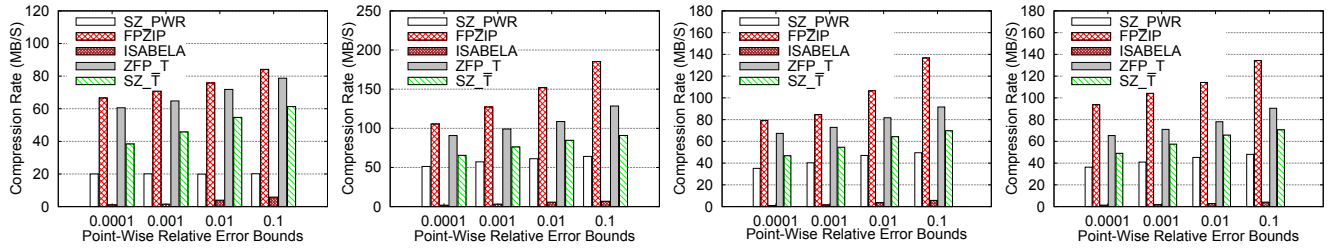
Here we showcase the compression performance (compression ratio and compression/decompression rate) of the above lossy compressors. However, tuning the parameter for ZFP_P for each field under each error bound is complicated because it does not respect the error bound. Also, according to two fields in NYX (Table IV), ZFP is not as competitive as FPZIP and SZ_PWR in terms of the compression ratio. Thus we do not test ZFP_P for overall performance from this section on.

The compression ratios of the lossy compressors on the four datasets are displayed in Figure 2. ISABELA usually cannot achieve a high compression ratio because of its high index overhead. SZ_PWR is competitive when the error bound is small, but its performance degrades for larger error bounds. Also, it is not good at sharply varying datasets such as HACC because of the group minimum design. FPZIP is good in most cases, but its performance suffers on 2D datasets, especially when the error bound is small. Our SZ_T almost outperforms all the other compressors by a certain scale under all the tested error bounds. However, ZFP_T does not exhibit a high compression ratio because it overpreserves the error bound.

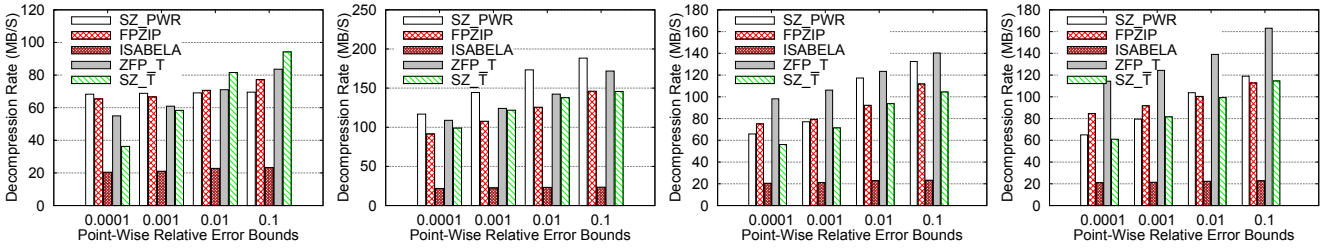
We also evaluate the compression/decompression rate of these lossy compressors. The results are shown in Figure 3. According to this figure, FPZIP leads all the other compressors in all the datasets in terms of compression speed. ZFP_T usually gets the second place because ZFP is faster than SZ. SZ_T is better than SZ_PWR since it does not compute complicated block information. ISABELA is slow because of the high sorting overhead. Regarding decompression, all the compressors except ISABELA exhibit comparable performance. SZ_PWR shows considerable improvement compared with its compression rate because it saves the block information and does not compute it during decompression.

E. Visualization for Multiprecision and Angle Skews

Besides compression performance, compression quality, such as the multiprecision visualization, is also important. Unlike absolute error bound, which requires universal restriction on each data point, pointwise relative error bound provides value-based restrictions that are usually different for different

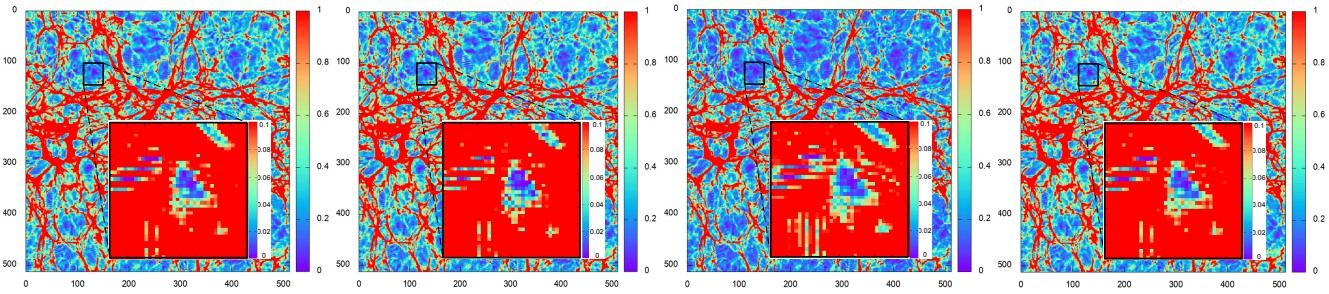


(a) HACC Compression Rate (b) CESM-ATM Compression Rate (c) NYX Compression Rate (d) Hurricane Compression Rate



(e) HACC Decompression Rate (f) CESM-ATM Decompression Rate (g) NYX Compression Rate (h) Hurricane Decompression Rate

Fig. 3. Compression/decompression rate on given point relative error bound



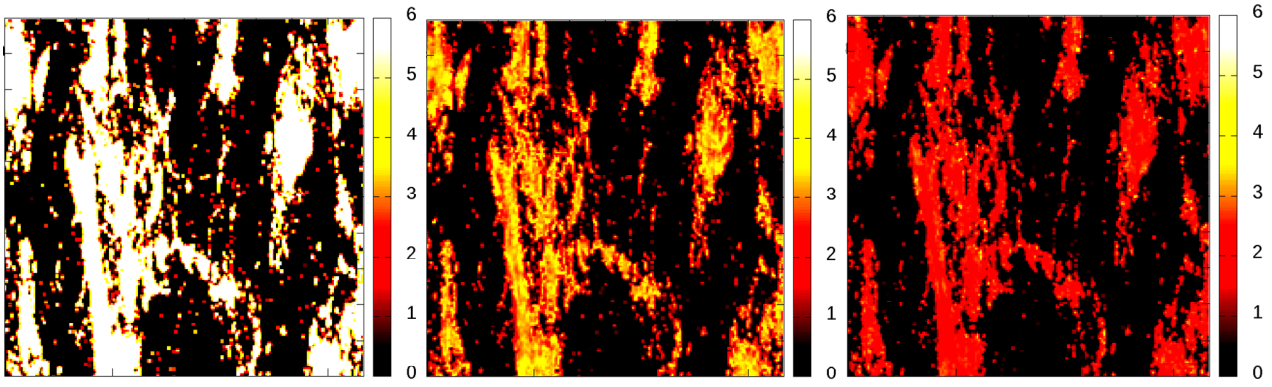
(a) Original data (b) SZ_ABS (c) FPZIP (d) SZ_T

Fig. 4. Multiprecision distortion of Slice 100 in dark_matter_density (NYX, $512 \times 512 \times 512$) when the compression ratio is 7. The original data is shown in range [0, 1], and enlarged windows are observed with a higher precision [0, 0.1]. Compared with SZ_ABS, FPZIP clearly keeps the features in blue parts (e.g., data in the center). However, it exploits more local loss and adds certain noise to regions between the blue and red parts (i.e., data in the top right and bottom left parts) since its max pointwise relative error is 0.5, which is much larger than that of SZ_T (0.15).

data points. Under this requirement, smaller value will have a smaller error bound on this data point and vice versa. Respecting this error bound is very effective when all the data points are all equally important regardless of their data value; otherwise, the small data value will be easily distorted by the universal restriction. In this section, we analyze the quality of pointwise error bound by visualizing the decompressed data generated by difference compressors.

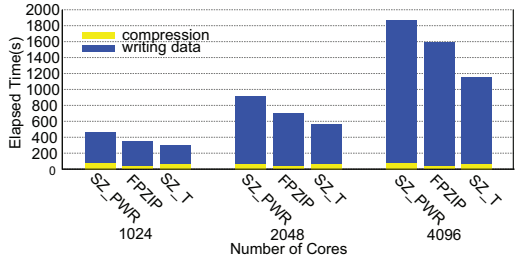
Figure 4 shows the multiprecision visualization results of original data, SZ_ABS, FPZIP, and SZ_T on the 100th slice in dark_matter_density fields of the NYX dataset when the compression ratio is set to 7. The absolute-error-bound mode of SZ is used as a comparison to demonstrate the advantages of pointwise relative error bound. Only FPZIP and SZ_T are selected because the other compressors cannot achieve such a compression ratio when the point-wise relative error bound is set to less than 100%. The original images show the visualization of a data range [0, 1] (value greater than 1 is shown as 1). The zoomed-in windows show a more precise range of [0, 0.1]. According to this figure, the absolute error bound does lead to certain distortion. The blue region in the center is distorted a lot because the universal restriction on each point is 0.055, which is large for data in range [0, 0.1]. On the other hand, FPZIP keeps the rough features in the center because it uses pointwise relative error bound, which has tighter restrictions on those blue regions. However, since it has to relax its pointwise error bound to 0.5 to reach such a compression ratio, it loses certain information for data in range (0.1, 1), which leads to the artifacts in the bottom left and top right. As a contrast, SZ_T needs only a pointwise relative error bound of 0.15 to get the same compression ratio. Thus its compressed data is distorted very little.

We also compare the skewed angles between original data and decompressed data for the velocity fields in HACC. A particle's skewed angle is defined as the angle between its original velocity and its reconstructed velocity in 3D space. It is calculated by $\theta = \arccos \frac{\vec{v}\vec{v_d}}{||\vec{v}||||\vec{v_d}||}$, where $\vec{v}$ is the original velocity in the 3D space and $\vec{v_d}$ is the reconstructed one. Because data points in HACC are scattered in the 3D space, we divide the whole space into $200 \times 200 \times 200$ blocks and compute the average skewed angle values in each block. We present the result at slice 100 in Figure 5. The brighter a region

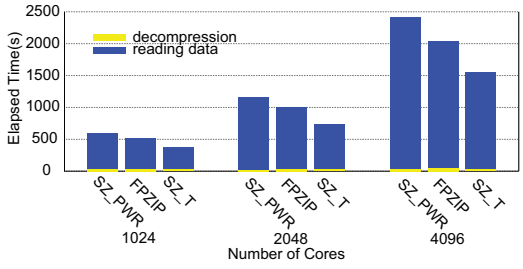


(a) SZ_ABS (b) FPZIP (c) SZ_T

Fig. 5. Angle skews of different compressors on HACC datasets when compression ratio is 8. The absolute-error-bounded compressor leads to large angle skews because the universal error bound (15 in this case) may greatly affect a small value. SZ_T has better performance because it has a stricter pointwise error bound (0.145) than that of FPZIP (0.334) under the given compression ratio.



(a) Data Dumping Time



(b) Data-loading Time

Fig. 6. Dumping and loading performance of NYX in parallel execution

is, the larger distortion it has, meaning a worse result. The figure shows that the absolute-error-bounded compressor has larger skewed angles (usually > 6), while pointwise relative-error-bounded compressors have much smaller skewed angles (around 4 for FPZIP and 2 for SZ_T). SZ_T is better than FPZIP because it has a stricter error bound (0.145 versus 0.334) at this compression ratio.

F. Parallel Evaluation

We still use the NYX dataset to demonstrate the data-dumping and data-loading performance in parallel execution. Since ISABELA and ZFP_T have a much lower compression ratio (ISABELA also has a relatively lower compression rate), we do not involve them in the parallel execution. For the rest of the compressors (SZ_PWR, FPZIP, SZ_T), we fix the pointwise error bound to 0.01 for all six fields in NYX datasets. We evaluate the three compressors on three scales (1,024 cores $\sim$ 4,096 cores). Each rank in the evaluation needs to process 3 GB of data, which corresponds to a total of 3 TB $\sim$ 12 TB of data. We plot the breakdown of data-dumping time (compression and writing) and data loading time (reading and decompression) in Figure 6. As a comparison, the original data needs about 0.7 $\sim$ 2.8 hours and 1 $\sim$ 4 hours for dumping and loading, respectively. From this figure, we can observe that our transformation scheme is able to achieve $1.62X$, $1.38X$ dumping performance and $1.55X$, $1.31X$ loading performance over SZ_PWR and FPZIP on 4k cores, respectively, thanks to the higher compression ratio and acceptable compression/decompression rates. Also, we tend to have more advantages when the scale continues to increase.

VII. Conclusion

In this paper, we formulate the bidirectional mapping problem between relative error bound and absolute error bound in order to use the existing lossy compressors in their best compression mode (absolute error mode). Under the specific constraints of lossy compression, we conclude that the logarithm in base 2 is the most effective transform from relative error bound to absolute error bound. We also prove the uniqueness of the log transform to solve our initial mapping problem. We develop a new lossy compressor by combining the existing state-of-the-art compressors and the logarithmic mapping scheme. Experimental results based on multiple real-world datasets across different scientific domains show that our solution can significantly improve the compression ratios (up to 60%) under the demand of pointwise relative error bound. Our solution achieves over $1.38X$ dumping and $1.31X$ loading performance over the second-best lossy compressor, respectively, in parallel experiments with 4,096 cores.

Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nations exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant No. 1619253. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

[1] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka, et al., "Simulating sky surveys on state-of-the-art supercomputing architectures," in *New Astronomy*, 42:4965, 2016.

[2] Lustre file system. [Online]. Available at: http://lustre.org/.

[3] A. Ovsyannikov, M. Romanus, B. V. Straalen, G. H. Weber and D. Trebotich, "Scientific Workflows at DataWarp-Speed: Accelerated Data-Intensive Science Using NERSC's Burst Buffer," in *1st Joint International Workshop on Parallel Data Storage and data Intensive Scalable Computing Systems (PDSW-DISCS)*, Salt Lake City, UT, 2016, pages 1-6.

[4] Gzip compression. [Online]. Available at http://www.gzip.org.

[5] M. Burtscher and P. Ratanaworabhan, "High Throughput Compression of Double-Precision Floating-Point Data," In *Data Compression Conference (DCC'07)*, pp. 293–302, 2007.

[6] BlosC compressor. [Online]. Available at http://blosc.org

[7] P. Lindstrom, "Error Distributions of Lossy Floating-Point Compressors," in *Joint Statistical Meetings*, 2017.

[8] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization," In *IEEE International Parallel and Distributed Processing Symposium IPDPS2017*, Orlando, Florida, USA, May 29-June 2, pp. 1129–1139, 2017.

[9] P. Lindstrom. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Trans. on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[10] S. Di and F. Cappello, "Fast Error-bounded Lossy HPC Data Compression with SZ," In *Proceedings of IEEE 30th International Parallel and Distributed Processing Symposium (IPDPS16)*, pp. 730–739, 2016.

[11] W. Kahan, "Ieee standard 754 for binary floating-point arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, no. 94720-1776, p. 11, 1996.

[12] S. Di, D. Tao, F. Cappello. "An Efficient Approach to Lossy Compression with Pointwise Relative Error Bound," in *DRBSD-2*, 2017.

[13] Theil, H. "Economics and information theory," (No. 04; HB74. M3, T4.), 1967.

[14] Trnqvist, L., Vartia, P. and Vartia, Y.O., "How should relative changes be measured?," The American Statistician, 39(1), pp.43-46, 1985.

[15] Wetherell, C., "The Log Percent (L%): An Absolute Measure of Relative Change," Historical Methods: A Journal of Quantitative and Interdisciplinary History, 19(1), pp.25-26, 1986.

[16] Campbell, J.W. and OReilly, J.E., "Metrics for Quantifying the Uncertainty in a Chlorophyll Algorithm: Explicit equations and examples using the OC4. v4 algorithm and NOMAD data", in Ocean color bio-optical algorithm mini workshop. Vol (Vol. 4, pp. 1-15), 2006.

[17] G.K. Wallace, "The JPEG still picture compression standard," in *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pages xviii-xxxiv, 1992.

[18] D. Taubman and M. Marcellin, "JPEG2000 image compression fundamentals, standards and practice: image compression fundamentals, standards and practice," in *Springer Science & Business Media*, vol. 642, 2012.

[19] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Ku, C. Chang, S. Klasky, R. Latham, R. B. Ross, and N. F. Samatova, "ISABELA for effective in situ compression of scientific data," in *Concurrency and Computation: Practice and Experience*, 25(4):524540, 2013.

[20] P. Lindstrom and M. Isenburg, "Fast and Efficient Compression of Floating-Point Data," in *IEEE Trans. on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[21] L. Ibarria, P. Linderstrom, J. Rossignac, and A. Szymczak, "Outofcore compression and decompression of large ndimensional scalar fields," in *Computer Graphs Forums*, vol. 22, no. 3, pages 343–348, 2003.

[22] R.J.Clarke, "Transform coding of images," AcademicPress, Inc., 1985.

[23] Bebop supercomputer. [Online]. Available at https://www.lcrc.anl.gov/systems/resources/bebop.

[24] B. Welch, "POSIX IO extensions for HPC," Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST05), 2005.

[25] R. Thakur, W. Gropp, E. Lusk, "On implementing MPI-IO portably and with high performance," Proceedings of the sixth workshop on I/O in parallel and distributed systems, pages 23–32, 1999.

[26] A. Turner, "Parallel I/O Performance," [Online]. Available at: https://www.archer.ac.uk/training/virtual/2017-02-08-Parallel-IO/2017_02_ParallelIO_ARCHERWebinar.pdf.

[27] CESM-ATM climate model. [Online]. Available at: http://www.cesm.ucar.edu/models/.

[28] NYX simulation. [Online]. Available at: https://amrex-astro.github.io/Nyx/.

[29] Hurrican ISABEL simulation data. [Online]. Available at http://vis.computer.org/vis2004contest/data.html