THE UNIVERSITY OF CHICAGO

OPTIMIZING ERROR-BOUNDED LOSSY COMPRESSION FOR SCIENTIFIC DATA WITH DIVERSE CONSTRAINTS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY
YUANJIAN LIU

CHICAGO, ILLINOIS
MARCH 15TH, 2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I foremost thank my advisor Dr. Ian Foster and Dr. Kyle Chard for guiding me through the process of coming all the way from China to the United States to pursue computer science research and writing elements of this Master's thesis. My grad school is packed with support and understanding and I owe a big thank you to you both.

I thank Dr. Sheng Di for providing me lots of knowledge, ideas, and time to discuss every detail of the research topic. You helped me a lot in forming the thesis' main idea and writing skills that will certainly manifest in my future efforts.

I acknowledge the senior Ph.D. student Kai Zhao for providing easy-to-read and well-written C++ coding and explanation, which helps me a lot when developing algorithms used in this thesis.

I am also grateful for the current and former students Xi Jiang, Sian Jin, Xin Liang, Matt Baughman, Greg Pauloski, Maksim Levental, Tyler Skluzacek for their kind support when I was confused about things from academics to life.

Finally, I want to thank my parents and other family members for their strong support of my decision to go abroad and pursue academic achievement.

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 will not be used 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 particularly during compression, resulting in reduced overall compression ratios due to data that provide little or no value for post hoc analysis. This thesis addresses this issue by proposing an optimized solution based on the state-of-the-art SZ lossy compression framework. Specifically, the solution can automatically clean the irrelevant data, efficiently preserve different precisions for multiple value intervals, and allow users to set diverse precisions over both regular and irregular regions. Evaluations are performed on a supercomputer with up to 3,500 cores. Experiments with six real-world applications show that the 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. The experiments also demonstrate a very good scalability in compression performance compared with the I/O throughput of the parallel file system.

CHAPTER 1

INTRODUCTION

Modern scientific simulations can produce extraordinary volumes of data. For example, climate and weather simulations [Kay et al., 2015] can produce terabytes of data in a matter of seconds [Baker et al., 2014b], and the Hardware/Hybrid Accelerated Cosmology (HACC) simulation code [Habib et al., 2016] 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 [with Globus]), which may have lower bandwidth.

Error-bounded lossy compressors [Lindstrom, 2014, Di and Cappello, 2016, Cappello et al., 2019, Wu et al., 2019b, Tao et al., 2018, Jin et al., 2019, Tao et al., 2017a] are widely used to reduce scientific data volumes while meeting user requirements for data fidelity. For instance, the SZ [Di and Cappello, 2016, Tao et al., 2017b], ZFP [Lindstrom, 2014], and MGARD [Ainsworth et al., 2018] 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 [Sasaki et al., 2015b, Baker et al., 2014b, 2019]. Similarly, adopting a customized error-bounded lossy compressor has been shown to reduce the memory capacity required for general quantum circuit simulations [Wu et al., 2019b].

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 sciences, different values in a dataset commonly have different significance to post hoc analysis. Thus, users can set different pre-

cisions (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 [NYX simulation] 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 [81,83]. Therefore, in order to preserve the features (such as the count and location) of the dark matter halos, the 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) [Kay et al., 2015] 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 areas within or near U.S. boundaries.

This thesis proposes 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. In particular, the 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.

The key contributions are summarized as follows.

- This thesis proposes a constraint-based error-bounded lossy compression model, which is the first attempt, to the best of our knowledge. The user-specified constraints include

1. The SZ compression framework is adopted because it provides leading error-bounded lossy compression quality, as verified by several studies of different scientific datasets [Di and Cappello, 2016, Tao et al., 2017b, Underwood et al., 2020, Liang et al., 2018a].

(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.

- A series of optimization strategies are developed for preserving constraints efficiently. Specifically, the quantization stage in the SZ error-bounded lossy compression framework is redesigned.

- A comprehensive evaluation is performed using multiple real-world scientific datasets across different domains. Experiments show that the proposed solution can respect users' constraints, while maintaining a high compression ratio. Specifically, the 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 global error-bounded compressor. Experiments also demonstrate good scalability in compression time compared with the parallel file system's I/O cost.

The rest of the thesis is organized as follows. Chapter 2 introduces existing compression technologies and discuss related work. Chapter 3 discusses the five scientific constraints posed by scientists across different domains. Chapter 4 formulates the research problem based on the SZ error-bounded lossy compression model. Chapter 5 proposes a battery of efficient algorithms to preserve the user-required constraints and also optimize the compression quality and performance for different cases. Chapter 6 presents the evaluation results. Chapter 7 summarizes the findings and concludes with a vision of future work.

CHAPTER 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 [Zstd, Gzip, Burtscher and Ratanaworabhan, 2008, Blosc compressor, 2018] and lossy compression [Di and Cappello, 2016, Tao et al., 2017b, Lindstrom, 2014, Lindstrom and Isenburg, 2006, Zender, 2016]. The former introduces no data loss during compression, but it suffers from very low compression ratios (generally ∼2:1 [Son et al., 2014, Ratanaworabhan et al., 2006]). The latter can achieve very high compression ratios [Di and Cappello, 2016, Tao et al., 2017b, Lindstrom, 2014, Liang et al., 2018a], but potential data loss may distort analysis results. This work focuses on error bounded lossy compression as it can reduce the data size much more significantly while preserving enough precision for utilizing the compressed data. The following sections first introduce the fundamental compression framework and then describe existing technologies for each stage.

2.1 SZ Compression Model

The error-bounded lossy compression model implemented by SZ is illustrated in Figure 2.1. 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 [Di and Cappello, 2016, Tao et al., 2017b] or blockwise [Liang et al., 2018a]) to predict the data values. In a 1D dataset, the prediction method is simply a first-order Lorenzo predictor [Tao et al., 2017b], 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 [Liang et al., 2018a]. 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 [Zstd] by default in SZ).

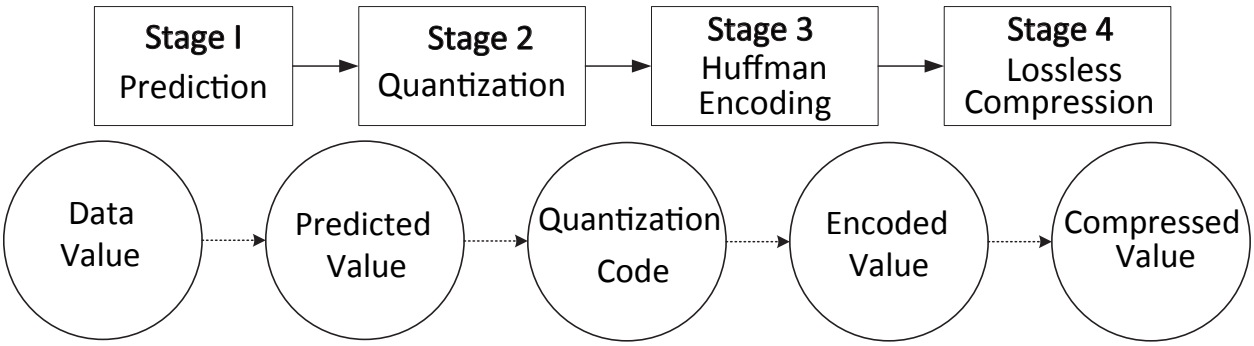


Figure 2.1: General procedure of prerequisite-preserving error-bounded lossy compression: Constraint (A) is handled before the prediction step; constraint (B) is handled primarily in both the prediction and quantization stage by replacing data points with Lorenzo-predicted values; constraints (C), (D), and (E) are addressed by designing a new quantization method.

2.2 Prediction Algorithms

2.2.1 Lorenzo Prediction

The easiest prediction method is called Lorenzo Prediction. According to Tao et al. [2017b], the value of (i_0, j_0) on the prediction surface can be expressed by the linear combination of the data values in the neighborhood, as shown in Figure 2.2. Because Tao et al. [2017b]'s evaluation results show that the first and second layer prediction yield the best results in most datasets, we only give the formula for the first two layers here.

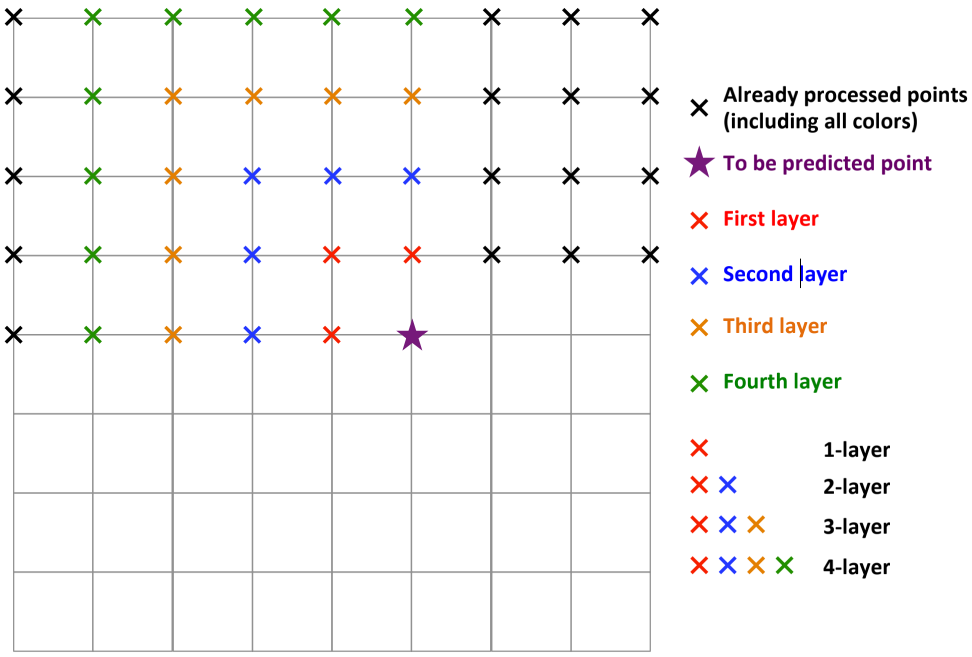


Figure 2.2: Example of Lorenzo Prediction in two-dimentional data: the point to be predicted can be predicted by its surrounding data points.

Let $V(i,j)$ be a function that returns the recontructed data of point (i,j). Let $f(i,j)$ be the prediction function for point (i,j). For the 1-layer lorenzo prediction, the formulae is

$$f_1(i,j) = V(i,j-1) + V(i-1,j) - V(i-1,j-1) \tag{2.1}$$

And for the 2-layer lorenzo prediction, the prediction formulae is

$$\begin{aligned} f_2(i,j) =& 2V(i-1,j) + 2V(i,j-1) - 4V(i-1,j-1) - V(i-2,j) - V(i,j-2) + \\ & 2V(i-2,j-1) + 2V(i-1,j-2) - V(i-2,j-2) \end{aligned} \tag{2.2}$$

Note that V does not return the original data but the reconstructed data, meaning the prediction of previous data points will affect the prediction result for the succeeding points.

2.2.2 Linear Regression

Liang et al. [2018a] proposed a linear regression method to predict data points, as illustrated in Figure 2.3. During the compression, each data block is approximated by a linear hyperplane with four coefficients ($f(x,y,z)$=$\beta_1 x + \beta_2 y + \beta_3 z + \beta_0$), and thus only four coefficient values need be stored to substitute all the data in that block. During the decompression, each data block would be recovered by the hyperplane reconstructed with the four regression coefficients. Without loss of generality, suppose the data block size is $n_3 \times n_2 \times n_1$, then its regression coefficients can be calculated as the following formula (d_{ijk} refers to the data values in the data block at the location $\{i,j,k\}$).

$$\begin{cases} \beta_1 = \frac{6}{n_1 n_2 n_3 (n_1+1)} (\frac{2V_x}{n_1-1} - V_0) \\ \beta_2 = \frac{6}{n_1 n_2 n_3 (n_2+1)} (\frac{2V_y}{n_2-1} - V_0) \\ \beta_3 = \frac{6}{n_1 n_2 n_3 (n_3+1)} (\frac{2V_z}{n_3-1} - V_0) \\ \beta_0 = \frac{V_0}{n_1 n_2 n_3} - (\frac{n_1-1}{2}\beta_1 + \frac{n_2-1}{2}\beta_2 + \frac{n_3-1}{2}\beta_3) \end{cases} \tag{2.3}$$

$$\text{where } V_0 = \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\sum_{k=0}^{n_3-1} d_{ijk}, \quad V_x = \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\sum_{k=0}^{n_3-1} i * d_{ijk},$$
$$V_y = \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\sum_{k=0}^{n_3-1} j * d_{ijk}, \quad V_z = \sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1}\sum_{k=0}^{n_3-1} k * d_{ijk}.$$

In this compression method, the compression ratio can be estimated based on the following formula. For instance, when the block size is 4×4×4, the compression ratio is 64/4=16; when the block size is 3×3×3, the compression ratio is 27/4=6.75.

$$CR = \frac{n_3 n_2 n_1}{4} \tag{2.4}$$

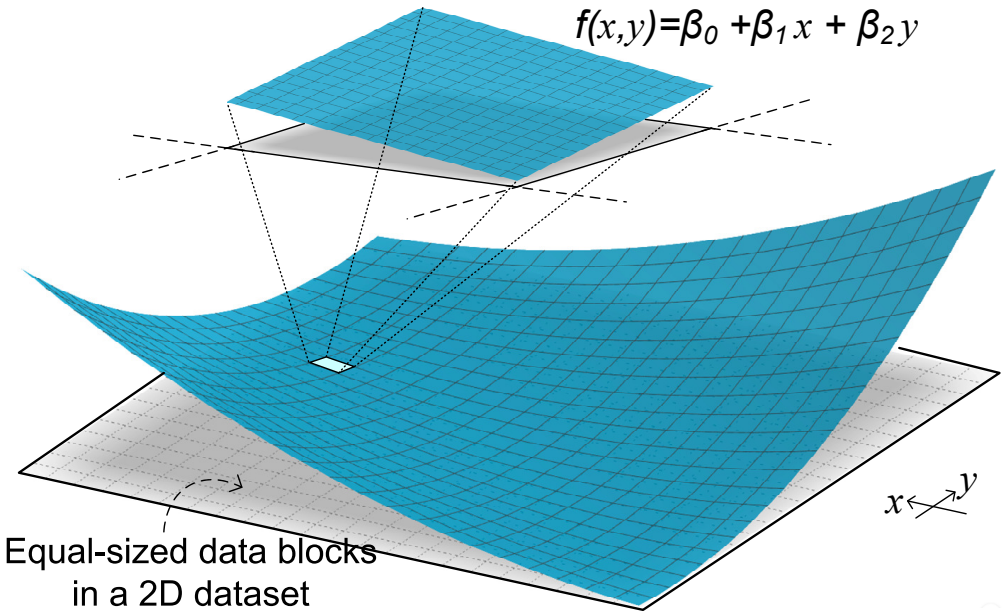


Figure 2.3: Illustration of linear regression model

2.2.3 Interpolation

Zhao et al. [2021] proposed a light weight cubic interpolation based prediction method for each unknown data point by only using its four surrounding data values. This method is claimed to outperform previous methods in terms of compression ratio (the prediction is more accurate) but have higher computational cost. The authors reduced the computational costs by precomputing a closed-form interpolation formula based on four specific neighbor data points (using the data points $i-3$, $i-1$, $i+1$ and $i+3$ to predict data point i as shown in Figure 2.4). There are two spline methods to do the prediction: (1) Linear spline; (2) Cubic spline. The closed form formula are equation 2.5 and 2.6 respectively.

$$p_i = \frac{1}{2}d_{i-1} + \frac{1}{2}d_{i+1} \tag{2.5}$$

$$p_i = -\frac{1}{16}d_{i-3} + \frac{9}{16}d_{i-1} + \frac{9}{16}d_{i+1} - \frac{1}{16}d_{i+3} \tag{2.6}$$

To predict data points in the whole dataset, the interpolation method follows a level-

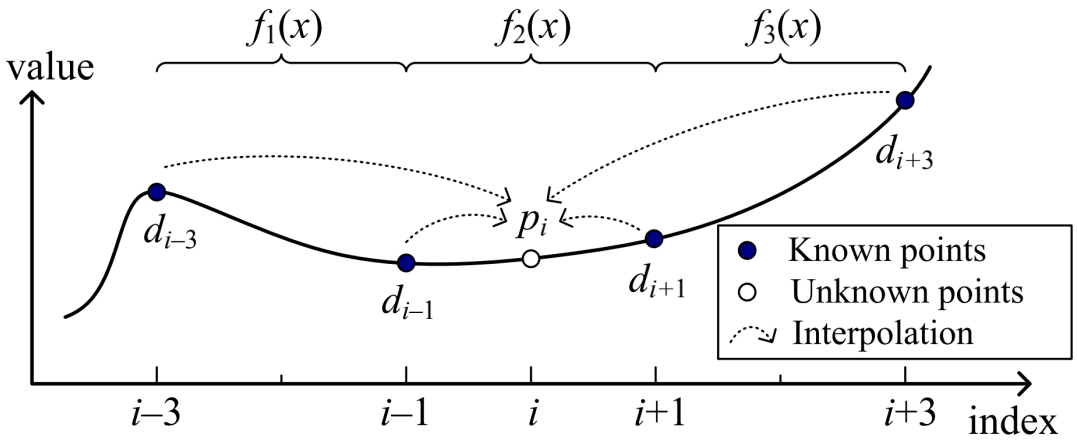


Figure 2.4: Illustration of Cubic Spline Interpolation

order pattern. Suppose the dataset has n elements in one dimension. The number of levels required to cover all elements in this dimension is $l = 1 + ceil(\log_2 n)$. In the example shown in Figure 2.5, at level 0, the algorithm uses 0 to predict d_1; at level 1, it uses d_1 to predict d_9. For now d_9 and d_1 are not necessarily 0 because there are quantizations to drag the predicted data back to an error bounded range. After recursively doing the interpolation through each level, all data points would be predicted properly.

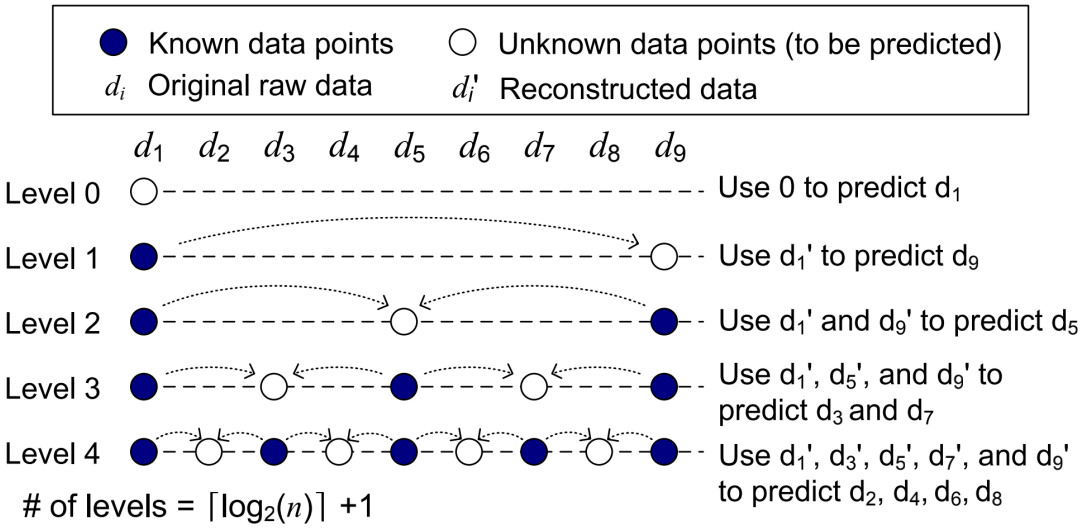


Figure 2.5: Illustration of Multilevel Linear Spline Interpolation

Such a multilevel interpolation can be applied on a multidimensional dataset. The idea

is that for each level, the interpolation will be done along every dimension. The details are presented in Zhao et al. [2021]'s work, and will be omitted here.

2.3 Quantization Algorithms

Previous work all simplified this part by using an integer quantizer with a constant error bound. This thesis focuses on the quantization algorithms to make improvement for compression performance. The following text briefly describe the existing error controlled quantization first proposed by Tao et al. [2017b].

The design of the quantization stage is shown in Figure 2.6. The first step is to use a prediction method to obtain a predicted value p. Tao et al. [2017b] call this value p the "first-phase predicted value". Then $2^m - 2$ values are expanded from the first-phase predicted value by sclaing the error bound linearly. And they are called the "second-phase predicted values". The quantization process is to add a certain integer (can be negative) times of double error bounds to p and make the specific second-phase predicted value within the error bound from the original value. Therefore, even if the prediction is not that accurate, the quantization can always pull the value back to the original data unless totally unpredictable.

2.4 Encoding Algorithms

Two encoding algorithms are adpoted in the SZ compression framework: (1) Huffman Encoding[Huffman, 1952]; (2) Run-length Encoding[Robinson and Cherry, 1967]. As they are classic algorithms, we will only give a very succinct introduction here.

Huffman Encoding is an Entropy-based lossless compression scheme which assigns each symbol in the data stream a unique prefix-free code. SZ will use this encoding to compress the quantization bins. Because the prediction is supposed to be relatively precise, there would be a large number of 0 or values near 0 in the quantization bins. The Huffman

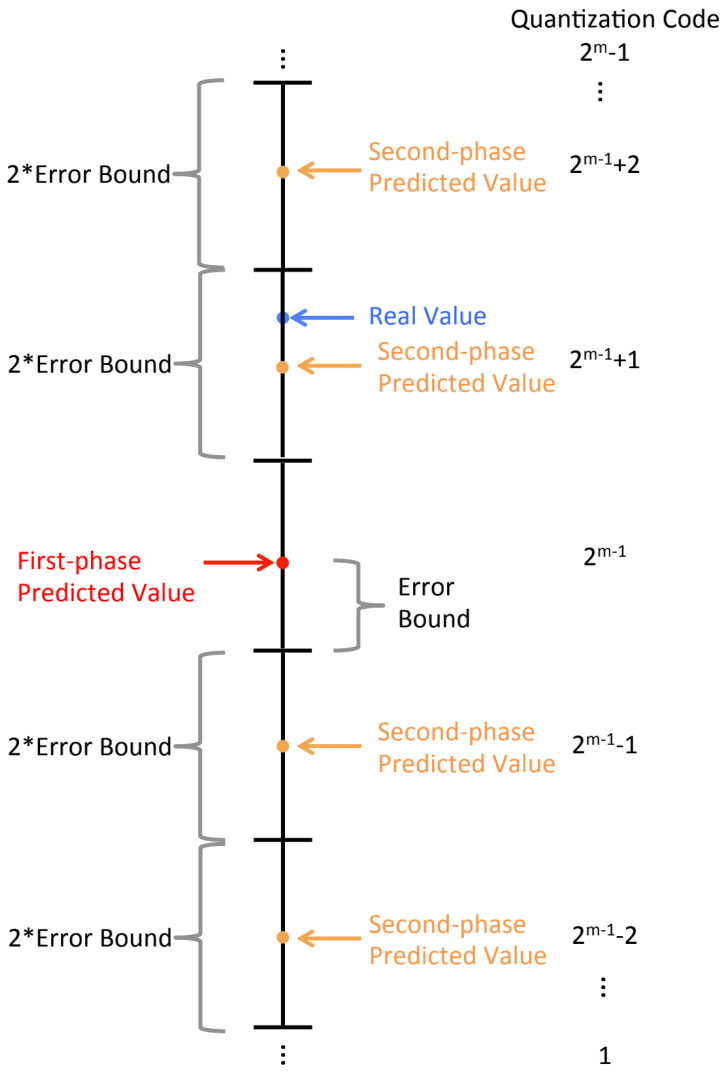


Figure 2.6: Illustration of The Basic Quantization Algorithm

Encoding can significantly reduces the quantization bins' size in SZ.

Run-length Encoding reduces size by eliminating the space taken by large number of repeating values. An illustrative example is to encode '111110000222222', the runlength encoding will be '5[1]4[0]6[2]', which records the number of repeating times for each consecutive pattern.

2.5 Lossless Compression Algorithms

Typical lossless compressors include Gzip[Gzip], LZ4[LZ4, 2018], FPC[Burtscher and Ratanaworabhan, 2008], Fpzip[Lindstrom and Isenburg, 2006]. Gzip[Gzip] is a generic compression tool that can compress any type of data stream, such as video stream and graph file. It integrates LZ4 and Huffman encoding to perform the compression. LZ4[LZ4, 2018] algorithm makes use of a sliding window to search the same repeated sequences of the data and replace them with references to only one single copy existing earlier in the data stream. FPC[Burtscher and Ratanaworabhan, 2008] is a lossless floating-point data array compressor, which analyzes the IEEE 753 binary representations and leverages finite context model. Fpzip[Lindstrom and Isenburg, 2006] was proposed to compress the HPC data by particularly focusing on the floating-point data compression. Fpzip can obtain higher compression ratio because of its more elaborate analysis on the HPC floating data, such as predictive coding of floating-point data.

In the SZ compression framework, ZSTD[Zstd] is used by default in the lossless compression stage because of its performance evaluated in [Tao et al., 2017b][Di and Cappello, 2016]. As the focus of this thesis is not on the lossless compression, ZSTD will be adopted in the lossless compression stage to compare fairly with previous SZ versions and demonstrate the improvement brought by new algorithms.

2.6 Addressing Diverse User Demands

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 [Di and Cappello, 2016, Tao et al., 2017b, Liang et al., 2018a], ZFP [Lindstrom,

2014, Diffenderfer et al., 2019], and MGARD [Ainsworth et al., 2018] 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 [Zhang et al., 2019]; and Digit Rounding [Delaunay et al., 2019], Bit Grooming [Zender, 2016], ZFP [Lindstrom, 2014], and FPZIP [Lindstrom and Isenburg, 2006] 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. Tao et al. [2018] 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 Ainsworth et al. [2018] 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 prerequisites (or constraints/conditions) 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.

CHAPTER 3

DIVERSE CONSTRAINTS IN SCIENTIFIC DATASETS

This chapter describes a novel concept in the practical use of error-bounded lossy compression—preserving diverse constraints specified by users.

A constraint is defined as a particular prerequisite that must be applied during the error-bounded lossy compression. There are five types of constraints that are commonly required in real-world science applications (see Table 3.1)

Table 3.1: Examples of user-required constraints applied to scientific simulation datasets

No.	User-Required Constraints	Science Domains
(A)	Isolating irrelevant value	Climate, Weather, etc.
(B)	Preserving global value range	Climate, etc.
(C)	Preserving value-interval-based error bounds	Weather, Cosmology, etc.
(D)	Preserving multiregion-based error bounds	Weather, Seismic imaging, etc.
(E)	Preserving irregularly shaped regions	Hydrodynamics, Weather, etc.

3.1 Irrelevant (or missing) data

Scienfitic datasets are often sparse, and missing data are often encoded in esoteric manners. Specifically, 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.

Although the irrelevant data contribute negatively to the compression ratio, they still carry important information such like coastline locations. Proper restoration of those irrelevant data in the decompression stage is required. This thesis will provide a series of

algorithms to handle the irrelevant data to improve the compression ratio while making sure they are restored after decompression.

3.2 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.

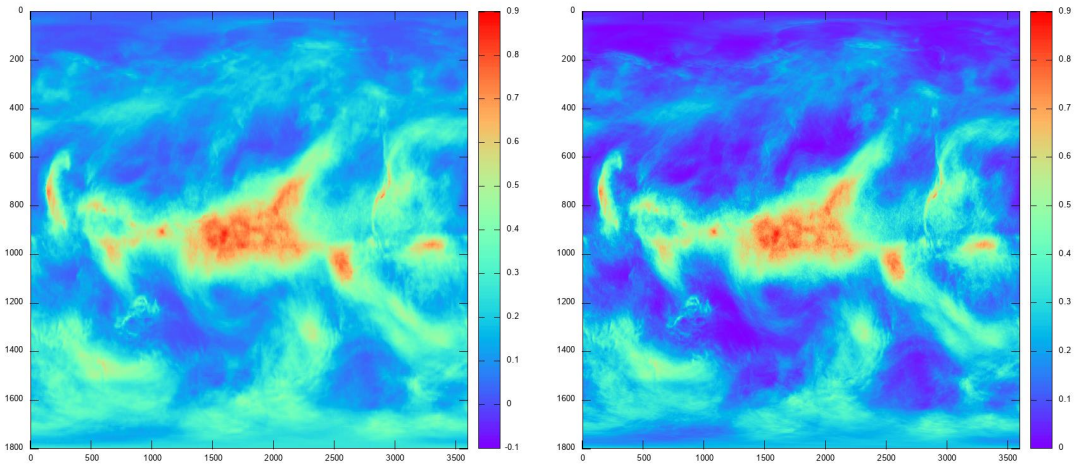


Figure 3.1: Without preserving the global range, the color mapping shifts, causing significant different visualization result compared to the original image.

Moreover, when analyzing data using visualization, often colors will be mapped according to the value range. If the image corresponding to the decompressed data has different color mapping compared to the original data, the visualization result will look very different, and may harm the post hoc analysis, as shown in Figure 3.1.

3.3 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 Scheffner et al. [1994] 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.

3.4 Region-based error bound

The data locations in scientific datasets always have a particular physical or chemical interpretation. For example, CESM [Kay et al., 2015] 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.

In particular, regions are required to be in regular rectangular shapes (for 2D data) or cubic boxes (for 3D data), so that a starting position vector and a length vector can sufficiently describe a region. For instance, if the starting position $p = (100, 100, 100)$ and

the length $l = (50, 50, 50)$, the region will be a cube with its upper left corner positioned at $(100, 100, 100)$ with a side length 50.

3.5 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 [Kay et al., 2015] may have sophisticated contours around lands and oceans, and scientists may wish to have higher precision in land areas. In this case the proposed solution allows users to mark a customized 2D or 3D area and use a bitmap array to specify different error bounds for every data point. It is also possible to use the bitmap to automate some region selection for users based on the data patterns. By applying advanced bitmap generation algorithm, the solution can preserve customized diverse precisions for a dataset.

Since the bitmap defines an error bound for each data point, the original size of a bitmap will be almost at the scale of original data size without proper compression. This thesis proposes an efficient method to compress the bitmap based on some assumptions so that the compression ratio for the bitmap can reach as high as 800, causing very limited overhead. The details will be discussed in the succeeding chapters.

CHAPTER 4

PROBLEM FORMULATION

This chapter formulates the diverse constraint error-bounded lossy compression problem. The compression work can be defined as an optimization problem, and the following text will clarify the optimization goal and the corresponding constraints respectively.

Given a scientific dataset D composed of N floating-point values (either single precision or double precision), the objective is to develop an error-bounded lossy compressor that can respect a set of user-defined prerequisites (i.e., conditions) such as preserving global value range or preserving multiple error bounds based on value intervals or different regions in the dataset.

4.1 Optimization Goal

Three assessment metrics are considered. The first two are compression speed s_c and decompression speed s_d, respectively. 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:

$$\rho = \frac{N \cdot sizeof(dataType)}{Size_{compression}}, \tag{4.1}$$

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

The optimization goal then can be formulated as Formula (4.2).

$$\begin{aligned} &\text{Maximize } \rho \\ &subject\,to \;\; user\text{-}required \;\; constraint \end{aligned} \tag{4.2}$$

4.2 User-required Constraints

The *user-required constraint* refers to additional requirements applied to the lossy compression beyond the traditional error-bounding constraint. Five constraints listed in Table 3.1 are formulated as follows:

$$\text{CONSTRAINT (A): } Preserve\ and\ isolate\ d_i \notin [R_{min}, R_{max}] \tag{4.3}$$

$$\text{CONSTRAINT (B): } Preserve \begin{cases} \max(\widehat{d_i}) = high(r(D)) \\ \min(\widehat{d_i}) = low(r(D)) \end{cases} \tag{4.4}$$

$$\text{CONSTRAINT (C): } |d_i - \widehat{d_i}| \leq e(d_i) \tag{4.5}$$

$$\text{CONSTRAINT (D, E): } |d_i - \widehat{d_i}| \leq e(LOC(d_i)), \tag{4.6}$$

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 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 4.1.

Here is an example to further illustrate how the research problem is formulated in this 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 (4.2) and Formula (4.5), the target is to maximize the compression ratio while ensuring 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

Table 4.1: 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

quantity of interest dark matter halo information. According to the Nyx analysis code [NYX simulation], 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 Chapter 6.

CHAPTER 5

ERROR-BOUNDED LOSSY COMPRESSION FRAMEWORK WITH DIVERSE CONSTRAINTS

This chapter designs a constraint-based error-bounded lossy compression framework in the SZ compression model Tao et al. [2017b]. All the aforementioned constraints will be addressed in the following text.

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, Lorenzo predicted values [Tao et al., 2017b] are adopted 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 5.1 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 $\leftarrow$ 1.25+1.27−1.23; 1.33 $\leftarrow$ 1.31+1.29−1.27).

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

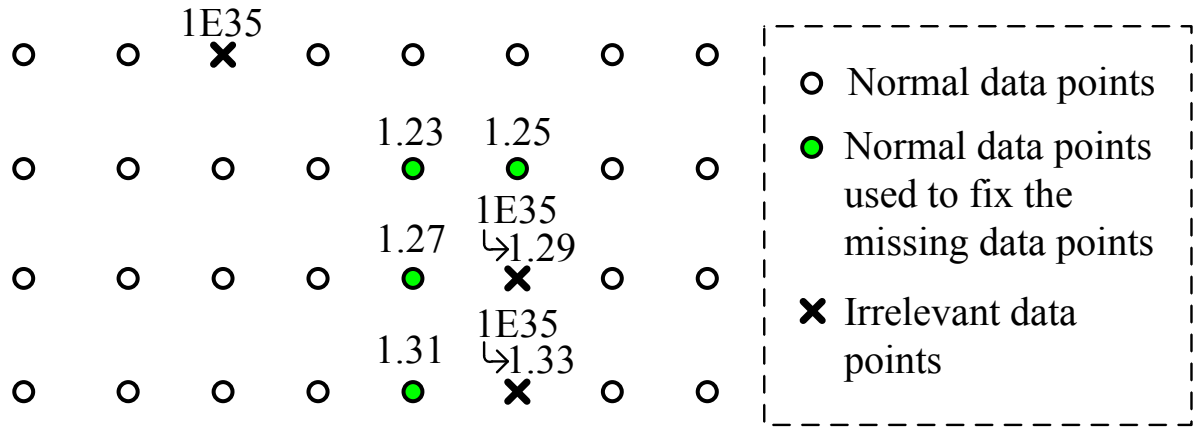


Figure 5.1: Illustration of how irrelevant data values are cleared in a 2D dataset.

- **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 Tao et al. [2017b], (2) quantization bin #1 is used to mark 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 $\sim$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 value-interval error bounds

To define a series of intervals with different error bounds, an array of triplets, each containing the *low*, *high*, and *error bound* was used. Figure 5.2 illustrates the 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. Different quantization bins (whose length is twice the error bound) are then applied in different intervals. As illustrated in the figure, each square denotes a quantization bin in its corresponding value interval. The 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.

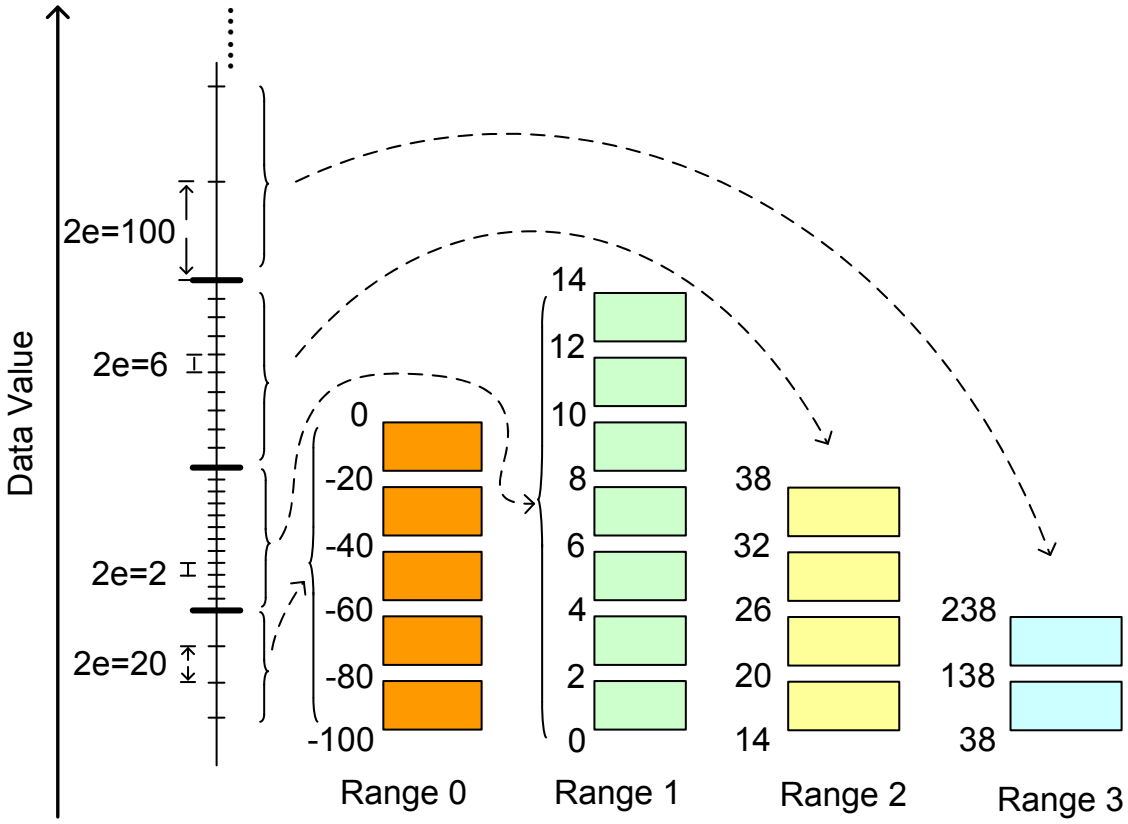


Figure 5.2: Multi-interval error-bound model. This example shows a few exaggerated error bounds in each range for simplicity of description: each rectangle represents twice the error bound in that range, and the ranges are tightly connected. The error bound will usually be smaller in practice, and each range may contain hundreds or thousands of error bounds.

Before describing the solution in detail, the notations will be reviewed. Let d_i denote the original data value at position i. Let p_i denote the predicted value for d_i. R is used to denote the radius of the quantization bin (for instance, if there are 65536 quantization bins, the radius R is equal to 32768). 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.[1] The notation are summarized in Table 4.1 in order to help understand the following text.

As illustrated in Figure 5.2, this thesis proposes a multi-interval quantization method that calculates the total number of quantization bin index 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

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

For each data point, three relationships between the original raw value d_i and its predicted

1. 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 the implementation.

value p_i must be dealth with: (1) $r(d_i) = r(p_i)$: they fall in the same interval; (2) $r(d_i) < r(p_i)$: the predicted data is in some range ahead of the original data; and (3) $r(d_i) > r(p_i)$: the predicted data is 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 Tao et al. [2017b]. Specifically, the following formulas can be used to compute the logic quantization code and decompressed data.

$$q = round(\frac{d_i - p_i}{2e(r(d_i))}) \tag{5.1}$$

$$\hat{d_i} = p_i + 2e(r(d_i)) \cdot q \tag{5.2}$$

The following example illustrates 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.

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., $r(d_i) \neq r(p_i)$). The following text describes 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, the quantized distance (i.e., the number of quantization bins) is counted from the predicted value to the original raw value. Whenever the counter crosses a different interval, it continues to add the quantization bins from the boundary of the new interval. As illustrated in Figure 5.2, 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 by the last value interval and its quantization bin size. The formula for reconstructing the decompressed value is given below (assuming the raw data is greater than the predicted value, without loss of generality):

$$q_t = round(\frac{d_i - low(r(d_i)) - e(r(d_i))}{2e(r(d_i))}) \tag{5.3}$$

$$\hat{d_i} = low(r(d_i)) + e(r(d_i)) + 2e(r(d_i)) \cdot q_t. \tag{5.4}$$

The decompression is described in Algorithm 2. The algorithm proceeds by executing similar operations to the compression process but in the 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 the edge conditions need to be handled 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, the quantization is shifted by 1 in the compression stage to ensure 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

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

require higher precision than others. For instance, in the CESM [Kay et al., 2015] dataset, the data indexes correspond to the geolocations, and some regions are more important than others for particular analyses (e.g., oceans, continents). Figure 5.3 illustrates the approach enabling users to mark interesting regions that we then use to apply a tighter error bound on each region according to the requirement and preknowledge of the data distribution.

To reduce the overhead in (de)compression time, regions are not set to each data point; instead, each intrablock of data are considered in the same region. To make the algorithm simpler, the intrablock is of size 6×6×6 for 3D data, which is consistent with SZ's linear regression prediction block size Liang et al. [2018a]. 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. This storage overhead is 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 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 composing 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. The solution to this problem will be described in Chapter 5.6.

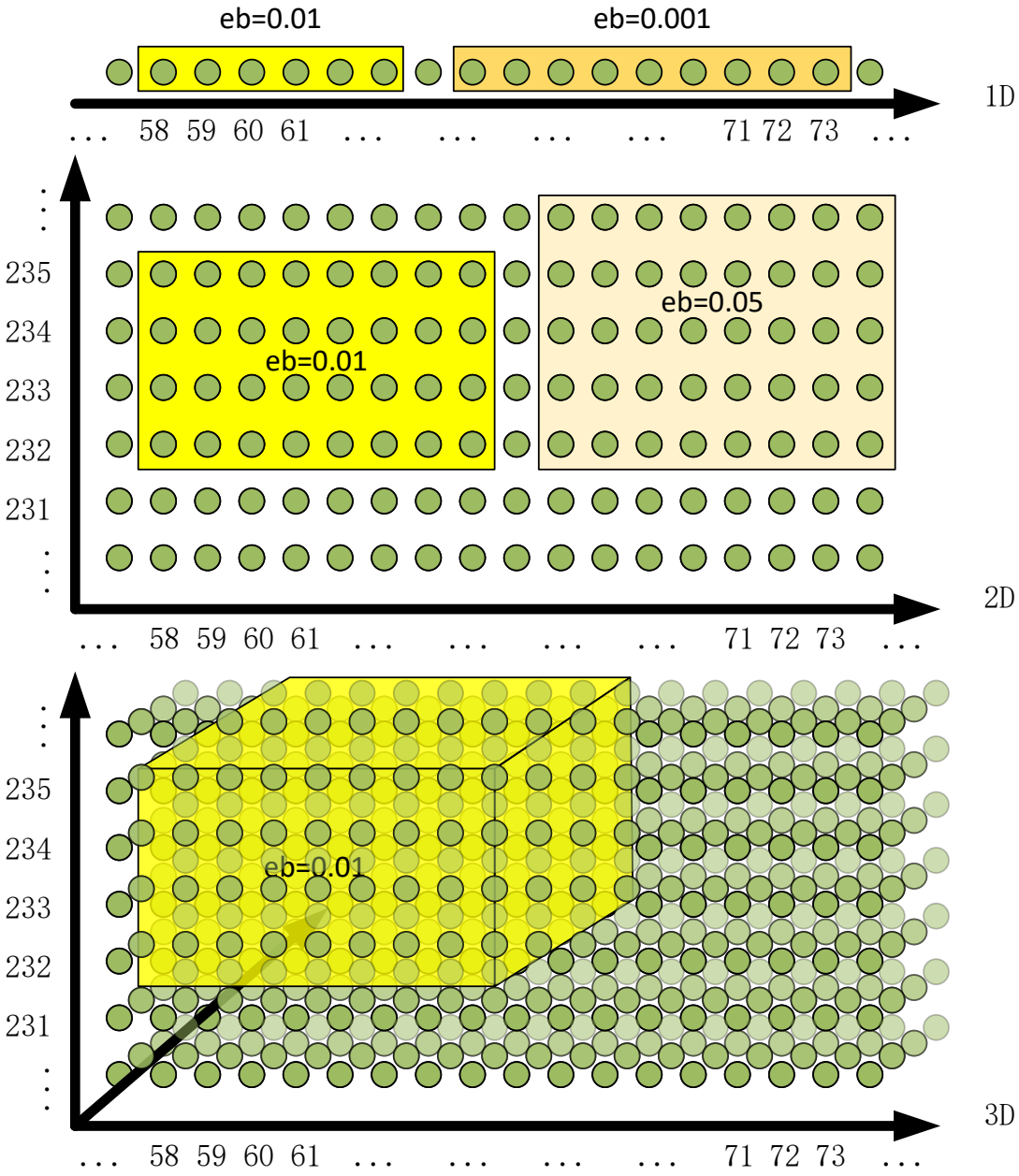


Figure 5.3: Constraint(D) region selection for 1D, 2D, and 3D data: In 3D cases, each region can be specified with seven parameters: the starting positions (3 parameters), the length of each direction (3 parameters), and the error bound (1 parameter).

5.5 Preserving irregular regions by bitmap

To satisfy complex, customized regions of error bounds (rather than just rectangles or cubes), this thesis introduces a bitmap error bound array (as shown in Figure 5.4). 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—there are rarely cases that require many different error 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." As such, one byte is sufficient to represent all different types of error bounds. In other words, a byte array can be adopted to store the index of error bound for each data point. After that, it is reasonable to apply Huffman coding and lossless compression to compress the bitmap array. 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 Chapter 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

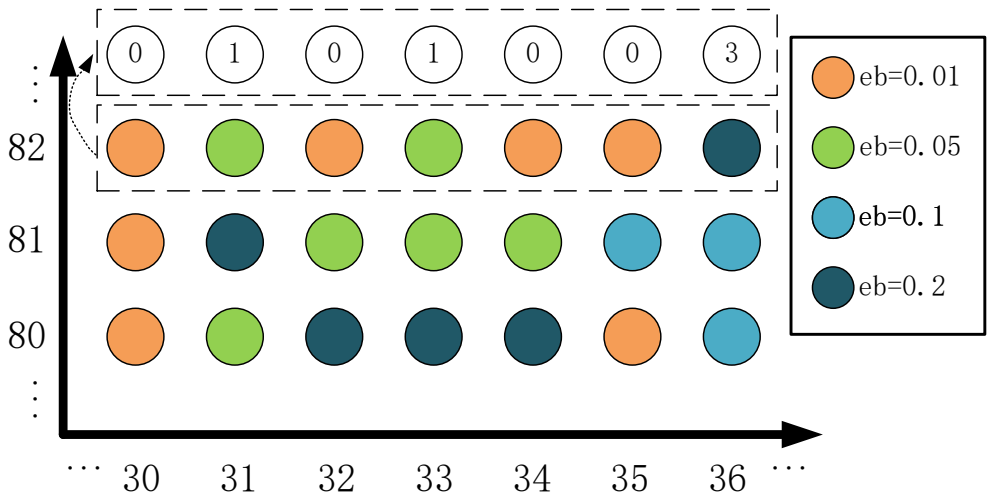


Figure 5.4: Illustration of bitmap error bound setting: Use an index to represent the error bound for each data point, and use a separate array to store all possible error bounds.

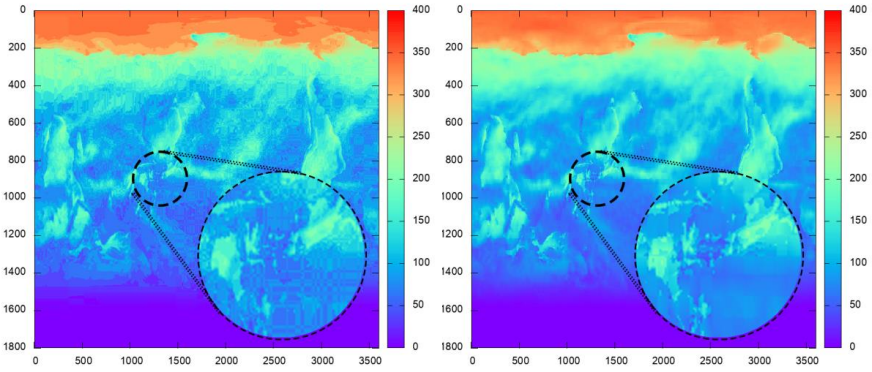
method when different error bounds are distributed evenly across the dataset. By setting a fixed proportion of data points with some certain error bounds, higher compression ratio, lower root mean squared error, and comparable visual quality can be achieved (the result will be presented in Chapter 6).

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 5.5, the two-precision setting ($eb_1 = 20$ at ocean and $eb_2 = 10$ at land) has a worse visual quality in the land area (with prominent stripe-pattern artifacts) than does a uniform ($eb = 20$) setting. The root cause is analyzed as follows. In fact, 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, whereas it is not as effective as linear regression when the error bound is high [Liang et al., 2018a]. Therefore, the Lorenzo predictor would tend to be selected in each block

at relatively low error bounds. Based on experimental observation, the artifacts shown in Figure 5.5 (A) are typical and are common to the Lorenzo predictor when the error bound is relatively high. This can be verified by Figure5.5 (C), in which the corresponding land area is using a linear regression predictor instead of Lorenzo 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 5.5 (B) and (C)). Moreover, the compression ratio is also degraded (compare Figure 5.5 (A) vs. (C)), which is undesired.

To overcome the artifact issue, a new predictor—called interpolation—is applied which 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 without the requirement of data blocks. This interpolation-based predictor may have much higher prediction accuracy than the linear regression predictor especially at high error bounds. The details about this interpolation-based compression method can be found in Zhao et al. [2021]'s work. This thesis combines the multiprecision design for the linear quantization stage with the interpolation-based predictor, which can thus resolve the artifact issue well. The evaluation of this method will be presented in the next chapter.



(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

Figure 5.5: 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).

CHAPTER 6

EXPERIMENTAL EVALUATION

This chapter evaluates all multiprecision compression methods designed above using multiple real-world simulations, and compares 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.

Evaluations are performed on datasets generated by seven scientific applications: QMCPACK QMCPack, RTM Migration, Miranda Miranda, CESM Kay et al. [2015], Nyx NYX simulation, Hurricane Isabel Hurricane ISABELA Simulation Datasets, and Hurricane Katrina Scheffner et al. [1994], as presented in Table 6.1.

Table 6.1: 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

All time measurements are performed on Argonne Bebop Machine. The machine's configuration is shown in Table 6.2.

Table 6.2: Machine Specifications

Partition	# Nodes	CPU	Cores/Node	Memory
bdwall	664	Intel Xeon E5-2695v4	36	128 GB DDR4

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

The Hurricane Isabel dataset is a good example to evaluate the preservation of irrelevant data because this dataset contains an irrelevant data value marked as 1E35, which is well outside the normal value range. As shown in Table 6.3, five out of 13 fields are chosen in the dataset because these fields all contain irrelevant (or missing) data points as well as specific value ranges. The reason why there exist missing values is that the data simulates an actual event (hurricane) and, in the locations where there is ground, no meaningful wind speed or pressure is recorded, thus causing missing values. More information about the dataset is available on the website.[1]

Table 6.3: 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 6.1 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

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

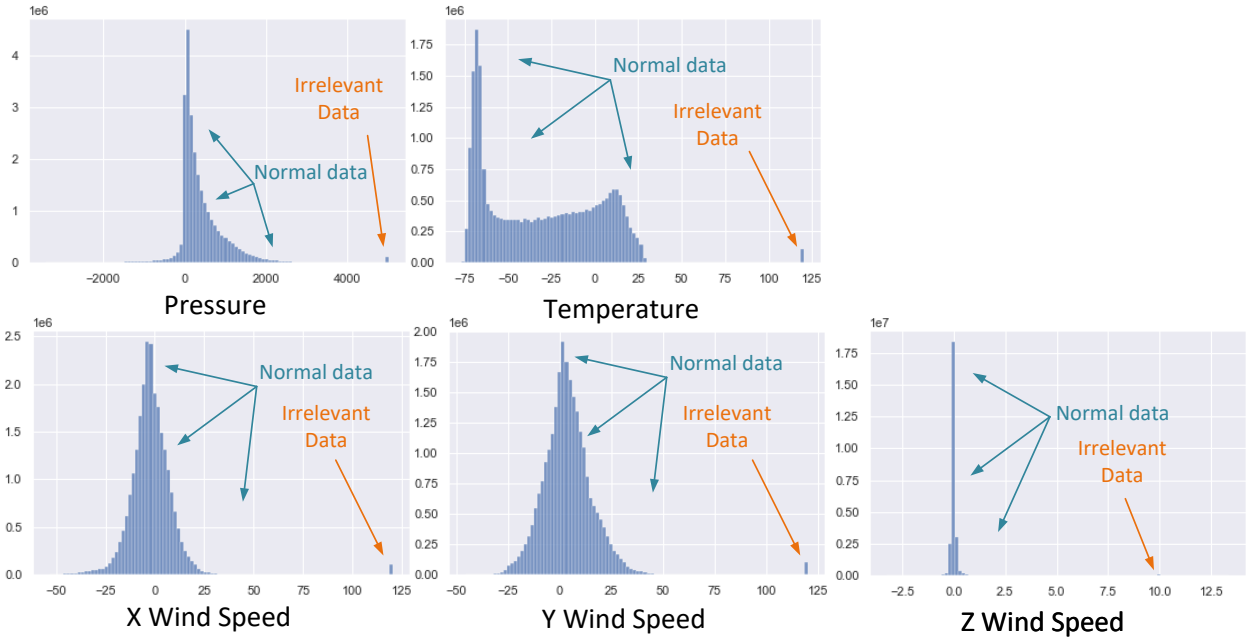


Figure 6.1: Data distribution of the five fields in the hurricane dataset. The irrelevant data value is 1E35. To visualize it in the distribution figure, the big value is modified to an outside value that is not in the normal data range.

normal data, a value that is outside the range of each field is used to represent the irrelevant value. The height of each bar means the total number of each value in the dataset. 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, they may severely harm the overall compression ratio because they are mixed among normal data points, destroying the continuity of normal data. This statement can be verified by sampling a random continuous portion of the temperature field, as shown in Figure 6.2.

Figure 6.2 clearly shows that the irrelevant data are distributed among the normal data, destroying the smoothness of the data space. Obviously, if predicting a normal data point using the irrelevant data value, the prediction cannot be precise. The goal of designing such a lossy compressor is to preserve the irrelevant data values while mitigating their influence on the compression ratio. Note that even though these data appear to be irrelevant for compression, they carry potentially useful information—in this case, they indicate the ground

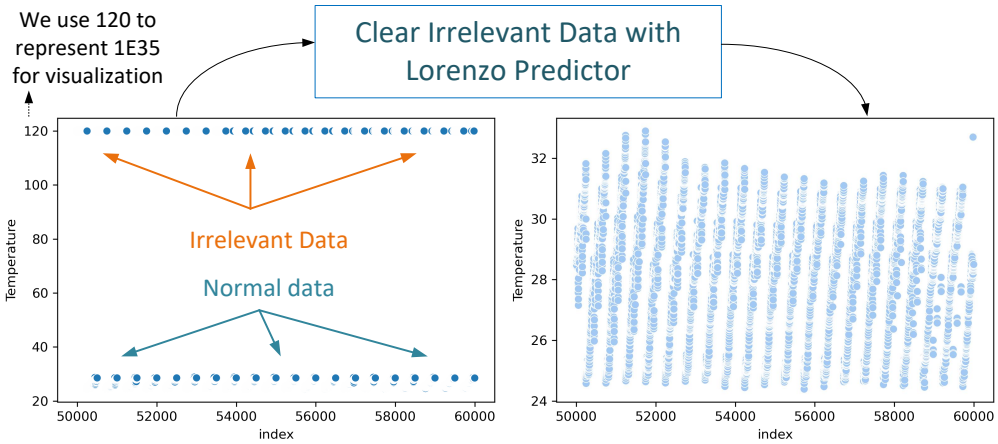


Figure 6.2: Temperature data points with (left) and without (right) irrelevant data. We show only a sample of 10,000 points (between index 50,000 and 60,000 in the original dataset). We observe the data points in the given index range and can see that the irrelevant data is mixed among the normal data points, harming data continuity; after clearing them with the Lorenzo predictor, the separating effect disappears.

locations.

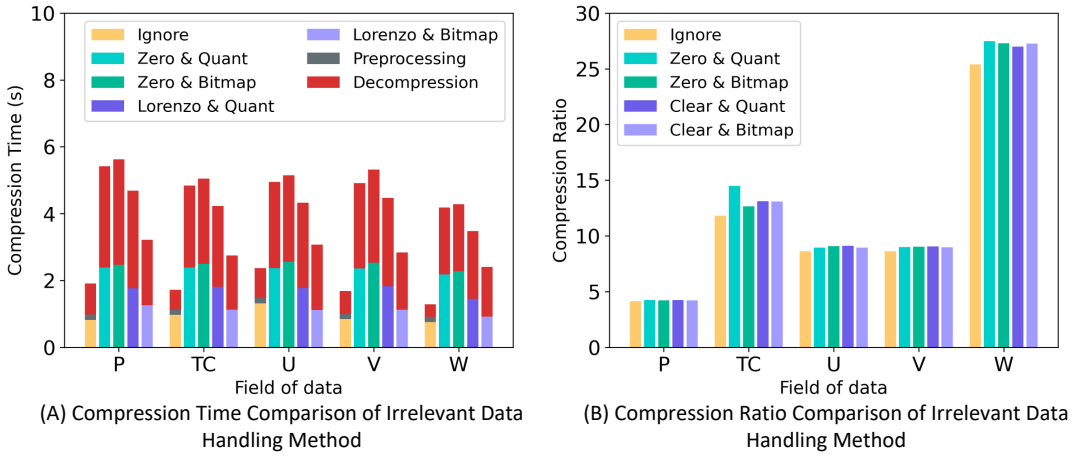


Figure 6.3: Performance of irrelevant data-handling methods: all methods slightly improve the compression ratio with a cost of longer compression time and decompression time.

In Figure 6.3, five different ways of handling irrelevant data are investigated. Time is measured on the Bebop machine[Bebop]. *Ignore* means treating all the irrelevant data as normal data. *Zero* replaces all the irrelevant data by 0 for simplicity, and *Clear* replaces all the 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 Bitmap indicates that we use a bit array containing 1 and 0 to indicate whether each data point is an irrelevant value or not. (A) shows that handling the irrelevant data may double the compression and decompression time. The overhead is mainly in the additional traversing of the whole dataset to find, clean, and recover the irrelevant data. Moreover, building up additional Huffman trees for irrelevant data will add additional time to the compression and decompression. (B) shows that handling irrelevant data is generally better than leaving them alone, but it is hard to determine whether clearing them with the Lorenzo predictor or simply zeroing them is better. Moreover, the simple bitmap method and quantization method exhibit similar performance. The likely reason is that the irrelevant data take only a very small portion so that the methods may never demonstrate a huge difference in terms of the overall compression ratio. The conclusion is in this scenario the quantization strategy slightly outperforms storing a bitmap.

The global range constraint is the easiest one to deal with, which requires only a traverse 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 almost ignorable, as indicated in the grey bar in Figure 6.3 (A).

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

Figures 6.4 and 6.5 show the substantial advantage of the 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

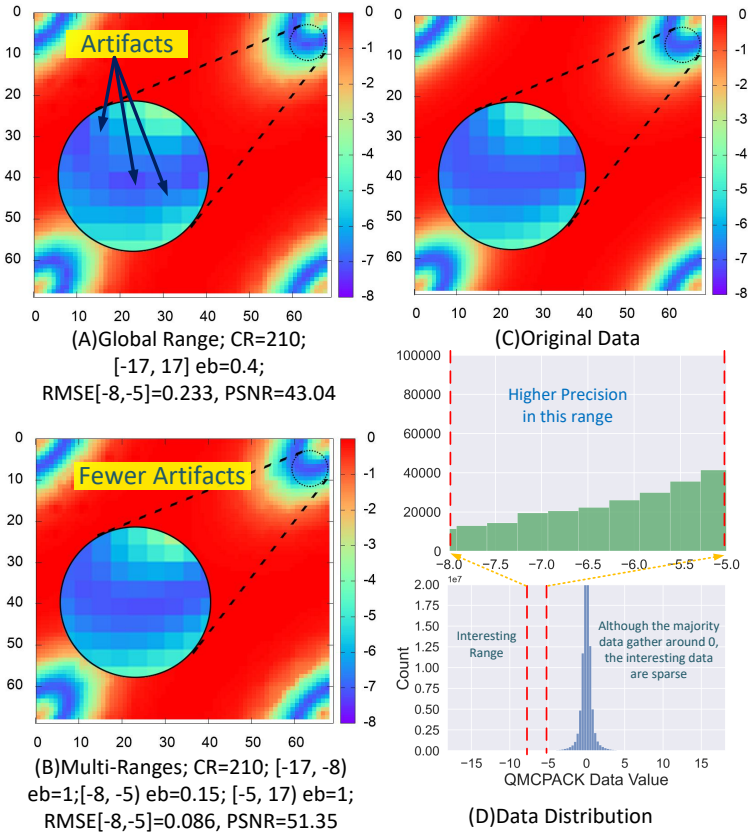


Figure 6.4: QMCPACK data: (A) The basic method is setting one error bound for the global range. Obvious artifacts are visible in the blue area. (B) Apply the multi-interval algorithm, set the interesting range [-8,-5) and give it a tighter error bound 0.15 while leaving other ranges a higher error bound 1. Fewer artifacts are observed, while the compression ratio is kept the same as the global range method. Compared with the original data shown in (C), the data in the interesting range have better visualization results.

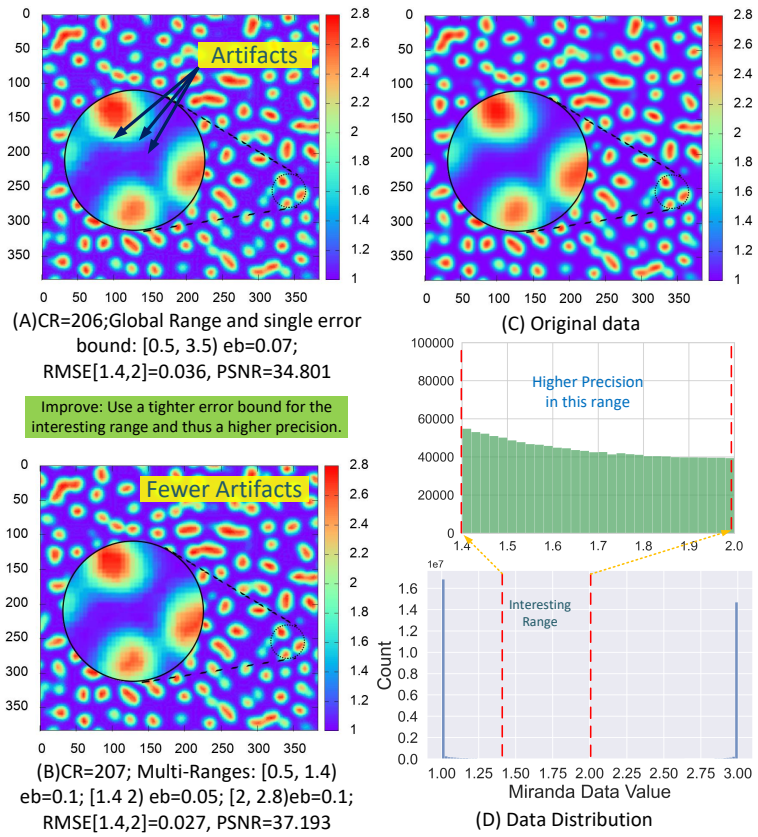


Figure 6.5: Miranda density slice No. 120. Comparing B with A, we can see that not only can the multi-interval solution preserve a high precision at a value range of interest with high compression ratio, but it also prevents the blue regions from getting distorted.

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.

The proposed 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 6.4 and Table 6.5. Given similar compression ratios, the proposed method can achieve lower RMSE and higher PSNR in the critical value interval.

Table 6.4: 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 6.5: 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

This section considers 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, the Friends-of-Friends algorithm Davis et al. [1985] is used. For the Nyx simulation, which is an Eulerian simulation instead of a Lagrangian simulation, the halo-finding algorithm uses density data to identify halos Friesen et al. [2016]. For decompressed data, some of the information can be distorted from the original, such as halo cells and halo mass.

Figure 6.6 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

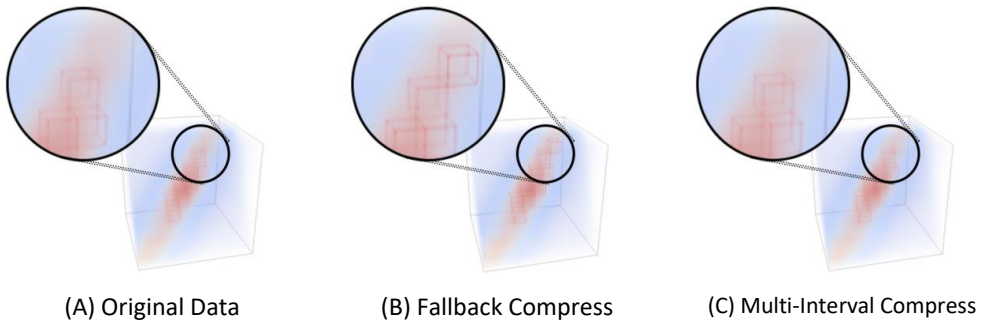


Figure 6.6: Nyx halo cell visualization: The fallback method sets a global error bound to be 0.5, and the compression ratio is 75. The proposed solution (C) sets three ranges: [min, 81) with error bound 1, [81, 83) with error bound 0.01, and [83, max) with error bound 1, and the compression ratio is 78. In the visualization, our solution (C) has cells almost identical to the original data's result, while the fallback method (B) shows some distortion and the cells' position and number are not identical to (A).

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 the compression task, three value ranges are set: a smaller error bound (0.1) is assigned to the data in the range of [81,83]. In this way the overall compression ratio will get higher with less distortion on the halo visualization result, as shown in Figure 6.6.

Table 6.6 shows the substantially higher precision of the proposed multi-interval error-bounded compression over global-range error-bounded compression. RMSE (root mean square error) of cell number differences of halos and RMSE of mass differences of halos are used in comparison with original data as two main metrics to evaluate the results. Specifically, when passed through the post hoc analysis, the multi-interval solution can lead to much lower RMSE of cell number and RMSE of halo mass compared with the original RMSE under the global-range error-bounded compression.

Table 6.6: Comparison of Different Range Settings. Fallback sets only a global error bound (here 0.01 and 0.5), and the multi-interval solution uses the proposed 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

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

This section investigates 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 (LorenzolLinear regression). A performance boost after applying the proposed algorithms will be shown, and it is also a good example to see that the multi-interval constraint indeed exists in the real-world scientific simulations.

Hurricane Katrina was one of the most devastating storms and the deadliest hurricane 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 study 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.

Katrina caused water elevation, and the elevation above 1 meter is considered to be significant; thus the hope is to preserve the information of the elevation data that are above 1 meter more precisely (the multi-interval constraint). Moreover, some data points do not have meaningful values in this dataset and are represented by -99999 (irrelevant data). Therefore, proper treatment to these values is needed to mitigate their influence on the compression per-

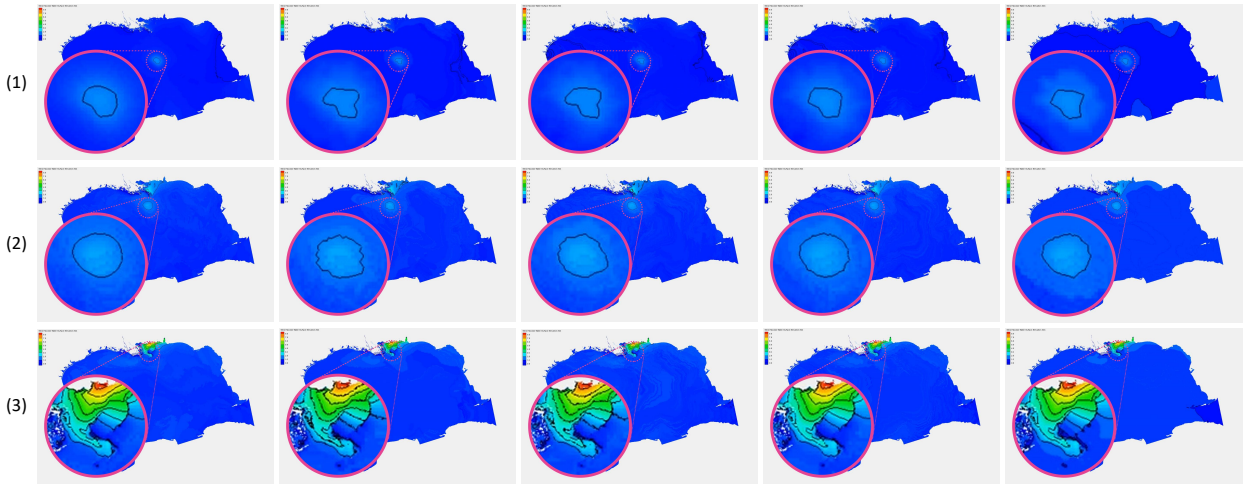
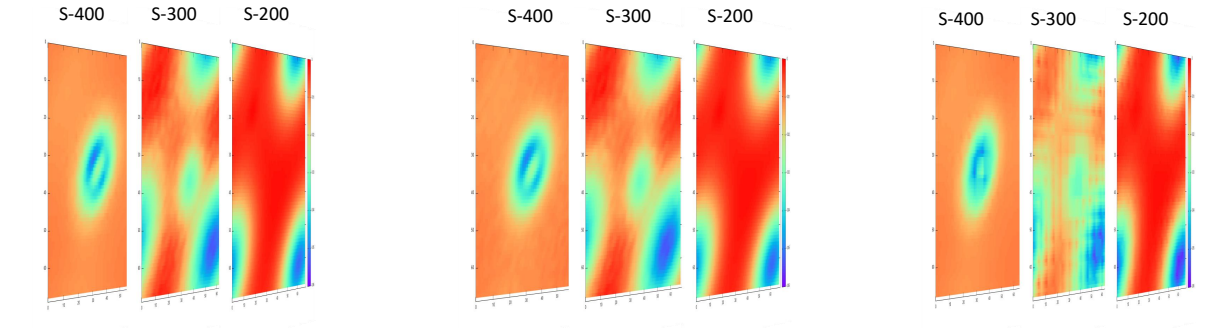


Figure 6.7: Hurricane Katrina data: Each row is a frame of the Katrina simulation: (1) is frame 120, (2) is frame 130, and (3) is frame 141. Each column represents a different setting of ranges and error bounds. Most of the blue data points in the graphs are close to zero. By applying a global range with error bound to be 0.01 with our solution, the visualization is almost identical to the original data's, and therefore one column (A) is used 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 using a composed Lorenzo and linear regression predictor to predict values. "Lorenzo" in (E) means using only the Lorenzo predictor with no linear regression. Comparing (B) and (C), the proposed 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), the multi-interval solution wins in both the compression ratio and visualization result. Comparing (D) and (E), a further improvement on the compression ratio can be made by using the Lorenzo predictor only and allowing some distortion in the deep blue area.

formance. By considering both irrelevant data and multi-interval error-bound prerequisites, the compression quality (as shown in Figure 6.7) can be improved significantly compared with the original compression quality under the state-of-the-art SZ 2.1; see Figure 6.7 (D) and (E) vs. (B) and (C).

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



(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.

Figure 6.8: 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].

To demonstrate the unique power of region-based compression method, post hoc analysis on three regions is performed 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 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 a range [-0.0025, 0.0005]. In Figure 6.8, significant distortion can be observed in the selected regions in (C) even though the error bound is generally small (0.01) for the whole dataset. The proposed 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 to the global dataset.

In addition to taking care of 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 6.9, the left-bottom corner has much better visual quality than

the rest has. These two examples 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.

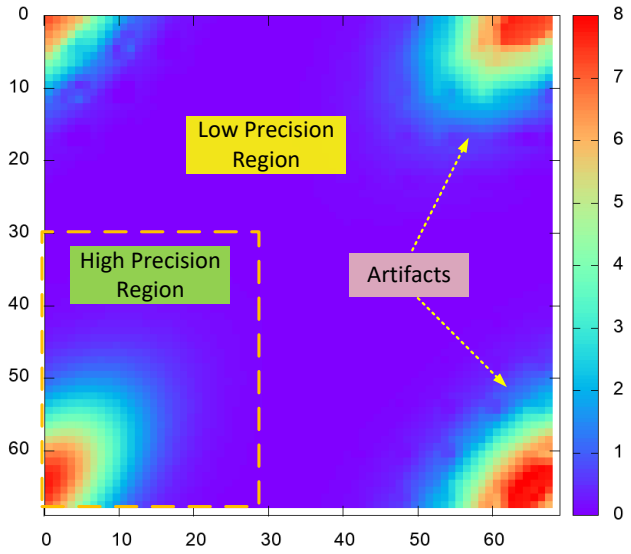


Figure 6.9: QMCPACK Slice 450, value range [0, 8]: A higher precision 0.001 for data in the area where $x \in [0, 30]$ and $y \in [30, 69]$, while keeping the error bound of other areas 0.5; The compression ratio is 242, and the SZ3 method with global error bound equal to 0.5 has a compression ratio 243. The region almost does not harm the compression ratio at all.

The feature "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. The CLDHGH field in the CESM dataset is used to exemplify this feature. Since the dataset has a rather tight value range and the neighboring values are quite smooth, it is hard to visualize the difference directly between the decompressed data and the original data in a small picture. The difference between each data point is calculated and then visualized instead. In Figure 6.10 (B), it is obvious that

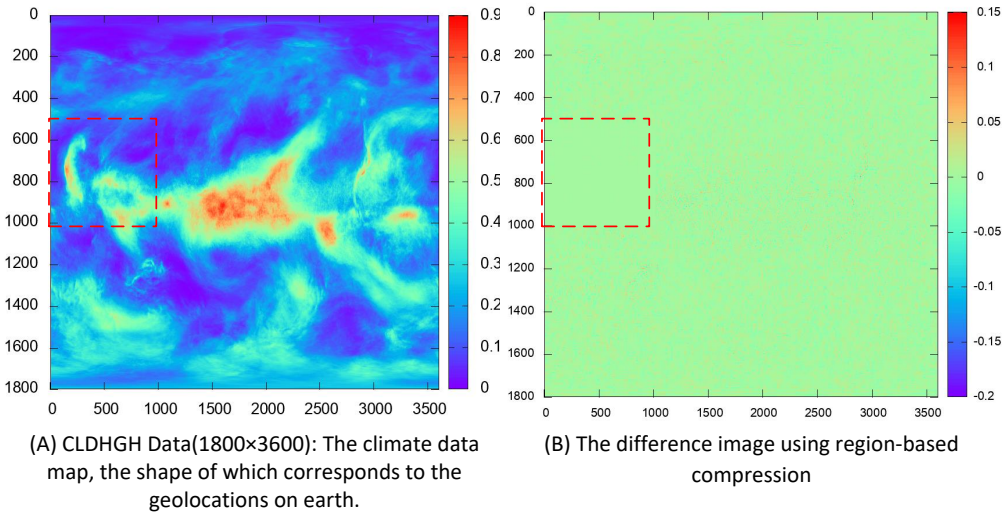


Figure 6.10: CESM with a region: while keeping the compression ratio high (CR=316), the interesting region is more precise (eb=0.01). The error bound for the rest regions is 0.02 in this example. If using the SZ3's global error bound, to reach eb=0.01 for the desired area, the compression ratio is 57.

the data inside the region (circled by a red rectangle) are much more precise than the other areas as there are almost no artifacts in the difference image. To reach the desired precision for the regions of interest, the region-based method clearly outperforms the traditional SZ compressor.

The (de)compression time overhead of both multi-interval and multi-region methods is also evaluated. The overhead of multi-region method is proportional to the number of regions, as each block needs to check the region list to find which region it belongs to, while the overhead of multi-interval method is highly related to the precision of prediction. To make the performance measurement as fair as possible, the same error bounds are used for all regions and value-intervals on six datasets. Five different regions/intervals are set for each compression to guarantee the overhead is observable.

The compression tasks are performed on Bebop bdwall partition with a single node, and the average of 10 runnings for each compression configuration is recorded. As shown in Figure 6.11 and Table 6.7, the compression time overheads of both multi-interval method and region-based method are not very high. The region-based method has slightly smaller

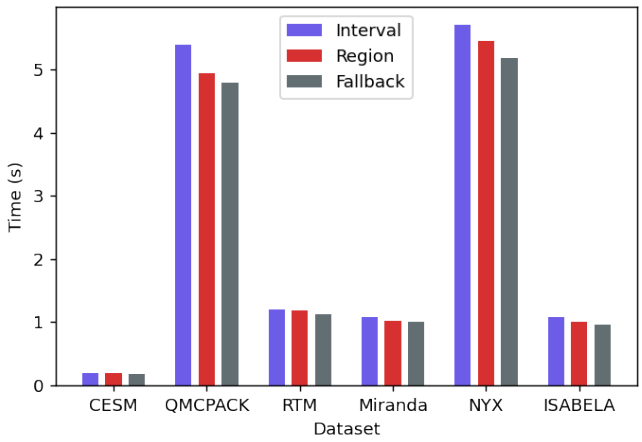


Figure 6.11: Compression time comparison: The reference point is the *Fallback* version, which means using a uniform error bound for all data points. The overhead of region-based method is slightly lower than the multi-interval method.

overhead compared to the multi-interval method. The main reason is that the proposed region-based method does not follow a point-to-point evaluation, instead, each intra block is stipulated to be of the same region, cutting down lots of unnecessary computation. The same approach cannot be applied to the multi-interval method because the assumption that neighboring points are in the same value interval does not hold, and actually, they are very likely to be in two different value intervals specified by the user. To conclude, both methods lead to a certain compression time overhead, while the overheads are confined within an acceptable range.

Table 6.7: 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)

Without loss of generality, users are assumed to be capable of marking the regions of interest by a third-party GUI toolkit, generating a bitmap array, which can thus allow users to specify fine-grained error bounds for irregular regions of interest. A bitmap defines the most concrete error bound information as it specifies an error bound for each data point. The overhead of storing a bitmap is non-negligible if not properly compressed. The following text presents the evaluation results about the bitmap solution in two situations: (1) the bitmap array is a background information which is stored separately by users as a metadata (e.g., the world map); (2) the bitmap needs to be stored with compressed data so it needs to be compressed as well.

Situation 1

The CESM dataset is a good example to evaluate this bitmap method. By setting different error bounds to ocean or land areas, the users can skip the details of uninteresting areas (either land or ocean) to achieve enough precision for only the regions of interest, with improved compression ratios. The proposed bitmap solution can help users specify different precisions in a fine granularity on irregular regions, by contrast with the other regular-region based multi-error-bounded compression method.

In the CESM dataset, the bitmap array is retrieved 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 6.12 (F)). Applying LANDFRAC as the bitmap, four different compression settings (described in Table 6.8) are tested on the other five data fields, as shown in Table 6.9. It can be observed that the bitmap solution sacrifices the 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) remain the same — P_0 almost does not decrease while P_1

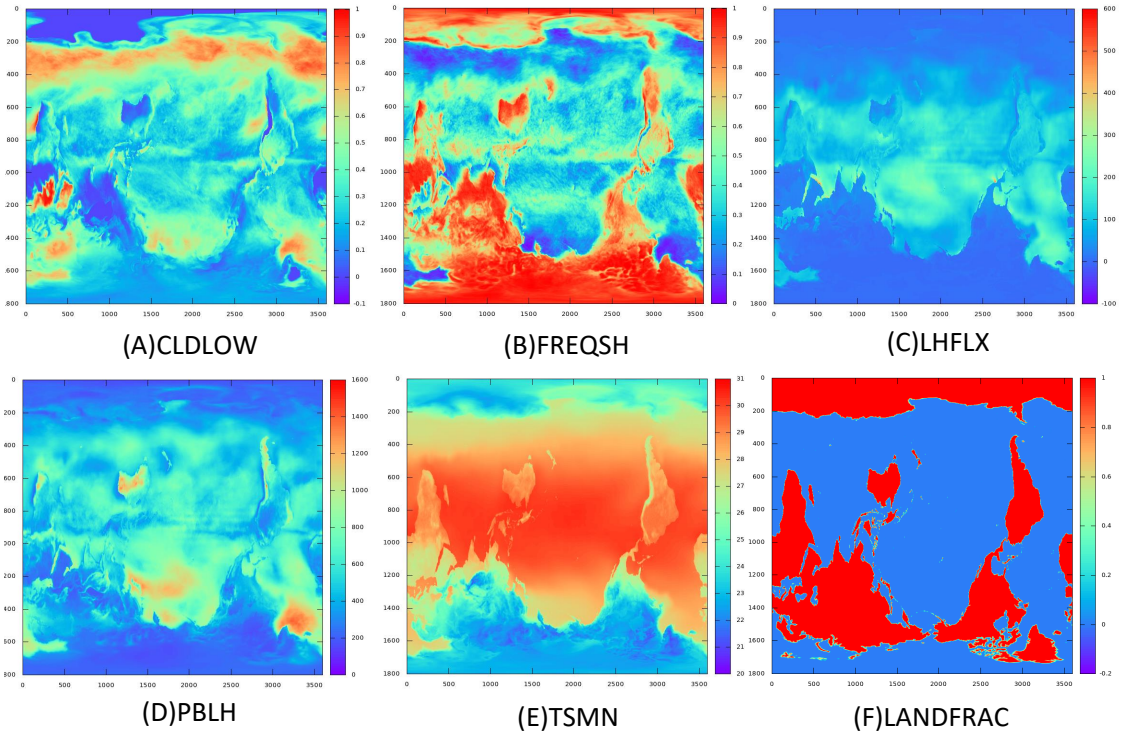


Figure 6.12: Six Fields in CESM: the visualization indicates that bitmap-separated precisions may be suitable to compress these fields.

decreases due to a larger error bound set in the corresponding area.

Table 6.9 demonstrates that the region-based multi-error-bounded compression method sigificantly outperforms all other solutions in compression quality. The key reason is two-fold. (1) The bitmap mehtod 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 comipression ratio. This can be verified by comparing the setting C and D in the table. (2) As discussed in Section 5.6, the interpolation predictor is much more effective than linear-regression predictor used by SZ2.1. This can be verified by comparing setting A and C in the table. The artifact issue described in Section 5.6 does not exist any more when applying the interpolation predictor, based on experiments. The visualization image is omitted here because of space limit. In fact, its visualization for the interpolation method is identical to Figure 5.5 (D).

Table 6.8: Compression Setting Definition

Setting	Description
A	SZ2.1 [Liang et al., 2018a]: 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 [Zhao et al., 2021]
D	The proposed region based error bounded compressor with two error bounds set by a bitmap

Situation 2

In the second situation where the bitmap array needs to be stored together with the compressed data, the bitmap array is compressed by integer-based Huffman encoding Di and Cappello [2016] and Zstd Zstd. Specifically, the input data would be the integer bitmap array with the same number of elements as the original dataset.

Table 6.9 also shows the compression ratio of the region-based multi-error-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 bitmap array, this column shows only the compression ratios for setting B and setting D. It is clear to observe that CR' is close to the CR (i.e., the compression ratio without storing bitmap array) in most of cases. This is because the bitmap array is fairly easy to compress with high ratios (reach $\sim$800 in this example) because of the limited number of error bounds. In fact, it is very common to have a very limited number of error bounds in practice because because of the limited number of value intervals of interest or the regions of interest in general. Accordingly, the error bound 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.

Table 6.9: 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	A: eb=0.01	21	-	44.94	46.74	49.59
min=-0.1	B: eb=0.01, 0.1	30	29.0	29.71	46.74	29.73
max=1	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	A: eb=0.01	16	-	44.73	46.76	48.97
min=0	B: eb=0.01, 0.1	22	21.4	28.67	46.76	28.67
max=1	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	A: eb=1	30	-	60.27	62.28	64.55
min=-100	B: eb=1, 10	48	45.4	49.36	62.28	49.55
max=600	C: eb=1	106	-	62.41	64.58	66.40
	D: eb= 1, 10	216	171.6	47.81	64.63	47.84
PBLH	A: eb=5	37	-	53.04	55.20	57.07
min=0	B: eb=5, 15	45	42.7	47.72	55.20	48.55
max=1600	C: eb=5	107	-	55.03	57.24	58.99
	D: eb= 5, 15	169	140.5	49.23	57.26	49.93
TSMN	A: eb=1	66	-	44.78	47.04	48.64
min=200	B: eb=1, 10	191	155.4	36.19	47.04	36.51
max=310	C: eb=1	292	-	47.14	49.41	50.99
	D: eb= 1, 10	812	411.5	31.64	49.24	31.66

6.7 Compression Time and Scalability

To evaluate the compression time and scalability, a series of tests are performed in parallel on thousands of CPU cores from Argonne LCRC Bebop supercomputer Bebop.

The dataset used to do the performance/scalability tests is QMCPack. This dataset is chosen because the size is more than 600MB as per file and it is stored in binary form so that the compression does not involve HDF5 plugins and other data extraction steps. The benefit is that the time cost by each step of the algorithms can be accurately recorded. Further, the multi-interval method takes the most amount of time among all multi-precision methods. The result is demonstrated using multi-interval algorithm so that it can serve as an upper bound for other methods. To be consistent with the previous quality tests, the multi-interval

error-bound requirement is set as follows: the data points in the range of [-8, -5) has much higher precision than the data points in other value ranges. According to the visualization result obtained from the previous section, it is observed that no data distortion can be viewed by naked eyes as long as a relatively low error bound 0.15 is used. Considering the potential impact of the lossy compression to user's analysis, a very low error bound (1E-5) is set for the range of interest: [-8,-5). That is, the compressor needs to use the error bound of 1E-5 for the value range [-8,-5), while the error bound needs be less than 0.15 in the other ranges. Preserving this constraint, experiments are performed on the Bebop supercomputer with different numbers of cores (each core has 600MB of raw data to compress). The results of BDW partitions is shown in Figure 6.13.

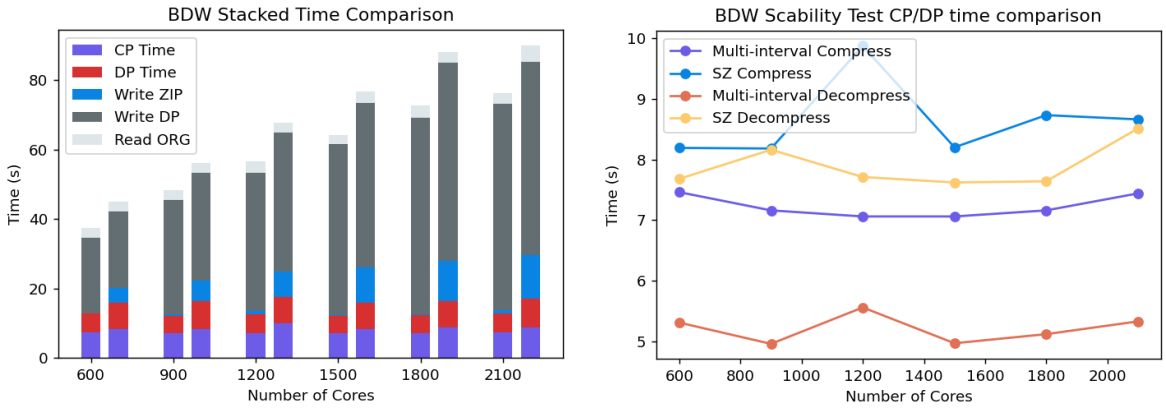


Figure 6.13: BDW partition: for each pair of bars, the left side is multi-interval solution's result, and the right side is SZ's result. CP/DP Time are compression/decompression time respectively. Write ZIP/Write DP are the I/O time to write the compressed/decompressed file respectively. Read ORG is the time to read the original file.

Based on the evaluation result, it is observed 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 they are both well-scalable is that the lossy compression adopted in practice follows an embarrassing parallel mode: there is no communication among the execution ranks/cores. The key reason our algorithm has less compression/decompression times than SZ is that our compression allows setting higher error bounds for non-interesting

ranges, which can lead to higher compression ratios.

Based on bar graphs of Figure 6.13, writing times take increasing portions with the number of cores. Specifically, 'Write Zip' and 'Write DP' refer to 'writing compressed data to PFS' and 'writing decompressed data to PFS', respectively. Obviously, the I/O cost has much worse scalability than the proposed lossy compression/decompression performance, especially because of the limited number of I/O nodes used by the system. In fact, no matter how powerful a supercomputer is, it always has a limited number of I/O node (thus an upper bound in I/O throughput), which may easily cause a serious I/O bottleneck when a large number of cores are used in an exascale simulation.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This thesis proposes multiple novel error-bounded lossy compression methods that allow preserving various user-defined constraints, which is the first attempt to the best of my knowledge. Based on the evaluation using real-world simulations, the key findings are summarized as follows:

- 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 multi-value-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 the proposed solution can possibly be generalized to suit all kinds of fine-grained error bound settings.
- Experiments on a supercomputer - Argonne Bebop Bebop with up to 3500+ cores show that the proposed multi-precision lossy compressors have a very good scalability.

As future works, one can explore new data fidelity requirements used by more applications in practice.

REFERENCES

Libpressio. URL `https://github.com/CODARcode/libpressio`.

Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science*, 19(5):65–76, Dec 2018. ISSN 1433-0369.

C. A. Andrews, J. M. Davies, and G. R. Schwarz. Adaptive data compression. *Proceedings of the IEEE*, 55(3):267–277, 1967. doi: 10.1109/PROC.1967.5481.

ANL Theta. `https://www.alcf.anl.gov/theta`, 2019. Online.

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*, 38(3):517–528, 2019. ISSN 0167-7055, 1467-8659. doi: 10.1111/cgf.13707.

Allison H. Baker, Haiying Xu, John M. Dennis, Michael N. Levy, Doug Nychka, Sheri A. Mickelson, Jim Edwards, Mariana Vertenstein, and Al Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pages 203–214, New York, NY, USA, 2014a. ACM. ISBN 978-1-4503-2749-7. doi: 10.1145/2600212.2600217. URL `http://doi.acm.org/10.1145/2600212.2600217`.

Allison H. Baker, Haiying Xu, John M. Dennis, Michael N. Levy, Doug Nychka, Sheri A. Mickelson, Jim Edwards, Mariana Vertenstein, and Al Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *HPDC'14*, pages 203–214, 2014b.

Allison H. Baker, Haiying Xu, Dorit M. Hammerling, Shaomeng Li, and John P. Clyne. Toward a multi-method approach: Lossy data compression for climate simulation data. In *High Performance Computing*, pages 30–42, Cham, 2017. Springer International Publishing. ISBN 978-3-319-67630-2.

Bebop. `https://www.lcrc.anl.gov/systems/resources/bebop`. Online.

Blosc compressor. `http://blosc.org/`, 2018. Online.

Martin Burtscher and Paruj Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Transactions on Computers*, 58(1):18–31, 2008.

Jon Calhoun, Franck Cappello, Luke N Olson, Marc Snir, and William D Gropp. Exploring the feasibility of lossy compression for pde simulations. *The International Journal of High Performance Computing Applications*, 33(2):397–410, 2019. doi: 10.1177/1094342018762036.

Franck Cappello, Sheng 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*, 33(6):1201–1220, 2019.

Marc Davis, George Efstathiou, Carlos S Frenk, and Simon DM White. The evolution of large-scale structure in a universe dominated by cold dark matter. *The Astrophysical Journal*, 292:371–394, 1985.

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*, 12(9):4099–4113, 2019.

Sheng Di and Franck Cappello. Fast error-bounded lossy HPC data compression with SZ. In *IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS)*, pages 730–739. IEEE, 2016.

James Diffenderfer, Alyson Fox, Jeffrey Hittinger, Geoffrey Sanders, and Peter Lindstrom. Error analysis of zfp compression for floating-point data. *SIAM Journal on Scientific Computing*, 02 2019.

FastLZ. `http://fastlz.org/`, 2018. Online.

Andreas Fischer. A special Newton-type optimization method. *Optimization*, 24(3-4):269–284, 1992.

Ian Foster, Mark Ainsworth, Bryce Allen, Julie Bessac, Franck Cappello, Jong Youl Choi, Emil Constantinescu, Philip E Davis, Sheng Di, Wendy Di, et al. Computing just what you need: online data analysis and reduction at extreme scales. In *European Conference on Parallel Processing*, pages 3–19, 2017.

Brian Friesen, Ann Almgren, Zarija Lukić, Gunther Weber, Dmitriy Morozov, Vincent Beckner, and Marcus Day. In situ and in-transit analysis of cosmological simulations. *Computational Astrophysics and Cosmology*, 3(1):1–18, 2016.

GA Gabriele and KM Ragsdell. OPTLIB: An optimization program library. *Mechanical Engineering Modern Design Series*, 4, 1977.

A. M. Gok, S. Di, Y. Alexeev, D. Tao, V. Mironov, X. Liang, and F. Cappello. PaSTRI: Error-bounded lossy compression for two-electron integrals in quantum chemistry. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–11, Sep. 2018. doi: 10.1109/CLUSTER.2018.00013.

Gzip. `https://www.gzip.org/`. Online.

Salman Habib, Vitali Morozov, Nicholas Frontiere, Hal Finkel, Adrian Pope, Katrin Heitmann, Kalyan Kumaran, Venkatram Vishwanath, Tom Peterka, Joe Insley, et al. HACC: extreme scaling and performance across diverse architectures. *Communications of the ACM*, 60(1):97–104, 2016.

David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. doi: 10.1109/JRPROC.1952.273898.

Hurricane ISABELA Simulation Datasets. `http://vis.computer.org/vis2004contest/data.html`.

Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. In *Computer Graphics Forum*, volume 22, pages 343–348. Wiley Online Library, 2003.

Sian Jin, Sheng Di, Xin Liang, Jiannan Tian, Dingwen Tao, and Franck 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*, HPDC '19, pages 159–170, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6670-0. doi: 10.1145/3307681.3326608. URL `http://doi.acm.org/10.1145/3307681.3326608`.

JE Kay, C Deser, A Phillips, A Mai, C Hannay, G Strand, JM Arblaster, SC 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*, 96(8):1333–1349, 2015.

S. Kim, M. Kim, J. Kim, and H. Lee. Fixed-Ratio Compression of an RGBW Image and Its Hardware Implementation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(4):484–496, 2016.

Davis King. Dlib C++ Library - Optimization. URL `http://dlib.net/optimization.html#global_function_search`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Daniel Laney, Steven Langer, Christopher Weber, Peter Lindstrom, and Al Wegener. Assessing the effects of data compression in simulations using physically motivated metrics. *Scientific Programming*, 22(2):141–155, 2014.

S. Li, S. Di, X. Liang, Z. Chen, and F. Cappello. Optimizing lossy compression with adjacent snapshots for n-body simulation data. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 428–437, Dec 2018. doi: 10.1109/BigData.2018.8622101.

X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello. An efficient transformation scheme for lossy data compression with point-wise relative error bound. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 179–189, Sept. 2018. doi: 10.1109/CLUSTER.2018.00036.

Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. Error-controlled lossy compression optimized for high compression

ratios of scientific datasets. In *2018 IEEE International Conference on Big Data*. IEEE, 2018a.

Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, 2018b.

Peter Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics*, 20(12):2674–2683, 2014.

Peter Lindstrom. Error distributions of lossy floating-point compressors. In *Joint Statistical Meetings*, pages 2574–2589, 2017.

Peter Lindstrom and Martin Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

LZ4. `http://lz4.github.io/lz4/`, 2018. Online.

Cédric Malherbe and Nicolas Vayatis. Global optimization of Lipschitz functions. URL `http://arxiv.org/abs/1703.02628`.

Reverse Time Migration. `http://www.seismiccity.com/RTM.html`. Online.

Miranda. `https://wci.llnl.gov/simulation/computer-codes/miranda`.

NYX simulation. `https://amrex-astro.github.io/Nyx`. Online.

M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In G. Di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, volume 83, pages 255–297. Springer US, 2006. ISBN 978-0-387-30063-4 978-0-387-30065-8. doi: 10.1007/0-387-30065-1_16. URL `http://link.springer.com/10.1007/0-387-30065-1_16`.

Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46, 2009.

QMCPack. `https://qmcpack.org/`. Online.

Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. Fast lossless compression of scientific floating-point data. In *Data Compression Conference (DCC'06)*, pages 133–142, New York, NY, USA, 2006. IEEE, IEEE.

A.H. Robinson and C. Cherry. Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE*, 55(3):356–364, 1967. doi: 10.1109/PROC.1967.5493.

J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

Matthew J Saltzman. COIN-OR: an open-source library for optimization. In *Programming languages and systems in computational economics and finance*, pages 3–32. Springer, 2002.

Naoto Sasaki, Kento Sato, Toshio Endo, and Satoshi Matsuoka. Exploration of lossy compression for application-level checkpoint/restart. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, IPDPS '15, pages 914–922, Washington, DC, USA, 2015a. IEEE Computer Society. ISBN 978-1-4799-8649-1. doi: 10.1109/IPDPS.2015.67. URL `http://dx.doi.org/10.1109/IPDPS.2015.67`.

Naoto Sasaki, Kento Sato, Toshio Endo, and Satoshi Matsuoka. Exploration of lossy compression for application-level checkpoint/restart. In *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 914–922, 2015b.

Norman W. Scheffner, David J. Mark, C. A. Blain, J. J. Westerink, and Jr. R. A. Luettich. Adcirc: An advanced three-dimensional circulation model for shelves, coasts, and estuaries. report 5. a tropical storm database for the east and gulf of mexico coasts of the united states. *DRP Technical Report DRP-92-6*, 1994.

Scientific Data Reduction Benchmark. `https://sdrbench.github.io/`. Online.

Yair. Shapira. *Matrix-Based Multigrid Theory and Applications*. Numerical Methods and Algorithms, 2. Springer US, New York, NY, 2. edition, 2008. ISBN 0-387-49765-X.

Snappy. `https://google.github.io/snappy`, 2018. Online.

Seung Woo Son, Zhengzhang Chen, William Hendrix, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. Data compression for the exascale computing era-survey. *Supercomputing Frontiers and Innovations*, 1(2):76–88, 2014.

Spring8. `http://www.spring8.or.jp/en/`, 2019. Online.

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)*, pages 314–318, 2018.

Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. In-depth exploration of single-snapshot lossy compression techniques for n-body simulations. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 486–493, 2017a. doi: 10.1109/BigData.2017.8257962.

Dingwen Tao, Sheng Di, Zizhong Chen, and Franck 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 (IEEE IPDPS)*, pages 1129–1139. IEEE, 2017b.

Dingwen Tao, Sheng Di, Xin Liang, Zizhong Chen, and Franck Cappello. Improving performance of iterative methods by lossy checkponting. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '18, page 52–65, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357852.

Dingwen Tao, Sheng Di, Hanqi Guo, Zizhong Chen, and Franck Cappello. Z-checker: A framework for assessing lossy compression of scientific data. *The International Journal of High Performance Computing Applications*, 33(2):285–303, 2019a. doi: 10.1177/1094342017737147.

Dingwen Tao, Sheng Di, Xin Liang, Z. Chen, and Franck Cappello. Optimizing Lossy Compression Rate-Distortion from Automatic Online Selection between SZ and ZFP. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1857–1871, 2019b. ISSN 1045-9219. doi: 10.1109/TPDS.2019.2894404.

Pavlo Triantafyllides, Tasmia Reza, and Jon C. Calhoun. Analyzing the impact of lossy compressor variability on checkpointing scientific simulations. In *In the Proceedings of the 2019 IEEE International Conference on Cluster Computing*, Cluster '19, Washington, DC, USA, 2019. IEEE Computer Society.

Ulrich Trottenberg and Anton Schuller. *Multigrid*. Academic Press, Inc., Orlando, FL, 2001. ISBN 0-12-701070-X.

Robert Underwood, Sheng Di, Jon C. Calhoun, and Franck Cappello. FRaZ: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data. `https://arxiv.org/abs/2001.06139`, 2020. Online.

Data Transfer with Globus. `https://www.globus.org/data-transfer`. Online.

Xin-Chuan Wu, Sheng Di, Emma Maitreyee Dasgupta, Franck Cappello, Yuri Alexeev, Hal Finkel, and Frederic T. Chong. Full state quantum circuit simulation by using data compression. In *IEEE/ACM 30th The International Conference for High Performance computing, Networking, Storage and Analysis (IEEE/ACM SC2019)*, pages 1–12, 2019a.

Xin-Chuan Wu, Sheng Di, Emma Maitreyee Dasgupta, Franck Cappello, Hal Finkel, Yuri Alexeev, and Frederic 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*, SC '19, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450362290.

C. S. Zender. Bit grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netcdf operators (nco, v4.4.8+). *Geoscientific Model Development*, 9(9):3199–3211, 2016.

Jialing Zhang, Xiaoyan Zhuo, Aekyeung Moon, Hang Liu, and Seung Woo 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.

Kai Zhao, Sheng Di, Maxim Dmitriev, Thierry-Laurent D. Tonellot, Zizhong Chen, and Franck Cappello. Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1643–1654, 2021. doi: 10.1109/ICDE51399.2021.00145.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, April 2004. ISSN 1941-0042. doi: 10.1109/TIP.2003.819861.

Zstd. `https://github.com/facebook/zstd/releases`. Online.