IPComp: Interpolation Based Progressive Lossy Compression for Scientific Applications

Zhuoxun Yang
Florida State University
Tallahassee, FL, USA
zy24b@fsu.edu

Sheng Di
Argonne National Laboratory
The University of Chicago
Lemont, IL, USA
sdi1@anl.gov

Longtao Zhang
Florida State University
Tallahassee, FL, USA
lzhang11@fsu.edu

Ruoyu Li
Florida State University
Tallahassee, FL, USA
rl13m@fsu.edu

Ximiao Li
Florida State University
Tallahassee, FL, USA
xl24g@fsu.edu

Jiajun Huang
University of South Florida
Tampa, FL, USA
jiajun.huang.cs@gmail.com

Jinyang Liu
University of Houston
Houston, TX, USA
jliu217@central.uh.edu

Franck Cappello
Argonne National Laboratory
The University of Chicago
Lemont, IL, USA
cappello@mcs.anl.gov

Kai Zhao*
Florida State University
Tallahassee, FL, USA
kzhao@cs.fsu.edu

Abstract

Compression is a crucial solution for data reduction in modern scientific applications due to the exponential growth of data from simulations, experiments, and observations. Compression with progressive retrieval capability allows users to quickly access coarse approximations of data and then incrementally refine these approximations to higher fidelity. Existing progressive compression solutions suffer from low reduction ratios or high operation costs, effectively undermining the approach's benefits. In this paper, we propose our interpolation-based progressive lossy compression solution that has both high reduction ratios and low operation costs. The interpolation-based algorithm has been verified as one of the best for scientific data reduction, but previously, no effort exists to make it support progressive retrieval. Our contributions are three-fold: (1) We thoroughly analyze the error characteristics of the interpolation algorithm and propose our solution, IPComp, with multi-level bitplane and predictive coding. (2) We derive optimized strategies toward minimum data retrieval under different fidelity levels indicated by users through error bounds and bitrates. (3) We evaluate the proposed solution using six real-world datasets from four diverse domains. Experimental results demonstrate our solution archives up to 487% higher compression ratios and 698% faster speed than other state-of-the-art progressive compressors, and reduces the data volume for retrieval by up to 83% compared to baselines under the same error bound, and reduces the error by up to 99% under the same bitrate.

CCS Concepts

• **Theory of computation** → **Data compression**; • **Information systems** → **Data compression**; • **Computing methodologies** → *Graphics file formats*.

Keywords

error-bounded lossy compression, scientific data management

*Corresponding author

1 Introduction

The increasing volume of scientific data generated by simulations, instruments, and observations has outpaced the data processing, storage, and transfer capabilities of modern computer systems, including both workstations and supercomputers. For example, in climate research, the Coupled Model Intercomparison Project (CMIP) [8] aims to advance our understanding of the climate system by coordinating standardized experiments with Earth System Models (ESMs) and thus enabling comprehensive comparisons of how different models represent past, present, and future climate conditions. Thanks to the rapid evolution of leading HPC systems, each successive phase of CMIP has seen significant increases in data volume – CMIP3 generated around 40 TB [1, 3], CMIP5 about 2 PB [3], and CMIP6 [9] exceeding 28 PB. Such an increase in volume poses unprecedented challenges to store, process, and analyze the data.

Data compression, particularly lossy compression, has emerged as a critical tool to mitigate these challenges for scientific applications [2, 4, 10, 11, 16, 17, 20–25, 27, 30, 38, 40], enabling scientists to fully harness the ever-increasing performance of new computing systems. Lossy compression reduces data size by approximating original information and discarding less critical details. This process results in smaller file sizes with an acceptable loss of fidelity. Unlike domains such as natural images and videos, lossy compression designed for scientific scenarios often includes the ability to restrict the maximum point-wise error, which is essential for preserving the accuracy of scientific computations. For example, the SZ lossy compressors [41] can reach 10∼100 times higher compression ratios than lossless alternatives while keeping the maximum error within a predefined bound [42].

Motivation: Although many scientific lossy compression solutions have been proposed in the recent decade, most of them only support decompression to a single fidelity level once the data is compressed, which restricts their broad utilization. On the one hand, scientific analyses often require different data fidelity levels as the subject of the study or the stage of investigation has diverse tolerance for data precision. For example, in hydrodynamic simulation, reconstructing viscosity may require a 2^{-5} finer precision compared with reconstructing vorticity from the same data field [35]. As a result, without progressive capability, researchers must compress data conservatively at the highest possible fidelity – even though only a handful of analyses truly require it – ultimately diminishing the overall effectiveness of data reduction. On the other hand, when analyzing multiple snapshots, fields, or regions, researchers often first identify patterns or areas of interest at a coarse level before committing resources to detailed analysis [28, 35, 39]. Without the ability of progressive retrieval, scientists have to always load the entire compressed data and decompress it at full precision. This not only increases the time and resources required for data loading and decompression but also delays subsequent analyses and scientific discoveries.

Limitation of state-of-art approaches: Despite the necessity of progressive retrieval in scientific compression, as highlighted in the above scenarios, supporting this functionality is challenging due to several factors. First, achieving both high fidelity and high compression ratios simultaneously is challenging, as these objectives often depend on fundamentally different algorithms, where optimizing one may come at the expense of the other. Second, straightforward progressive techniques often introduce significant operational overhead, requiring multiple passes of decompression and reprocessing – contradicting the goal of progressive decompression, which is to save time and resources. Third, it is already non-trivial to guarantee error bounds for partial decompression, not alone to say the progressive technique which requires the data accumulated from multiple levels to be within acceptable error margins. Consequentially, few such progressive solutions exist, and they often fall short due to low compression ratios, high operational costs, and a lack of stringent error restrictions. These limitations hinder their adoption in scientific applications.

Key insights and contributions: In this paper, we propose the first high ratio, fast, and error-bounded progressive lossy compression

solution based on the interpolation algorithm. Our contributions are three-fold:

- We thoroughly analyze the error characteristics to build a progressive compressor based on the interpolation algorithm and propose our progressive solution, IPComp, based on the prediction model with multi-level bitplane and predictive coding. Besides having high effectiveness on retrieval, our solution supports retrieval under arbitrary error-bound settings, and it only executes decompression once for each retrieval request, compared with residual-based alternatives, which support a limited number of error bounds and require multiple passes of decompression for a single request.
- We derive optimized strategies towards minimum data retrieval under different fidelity levels indicated by users through error bounds and bit rate. Our strategies are highly effective while being extremely lightweight with negligible overhead to the scientific workflow.
- *Experimental methodology and artifact availability:* We evaluate the proposed solution using six real-world datasets from four diverse domains over four state-of-the-art compressors. Experimental results demonstrate our solution archives up to 487% higher compression ratios and 698% faster speed than other state-of-the-art progressive compressors and reduces the data volume for retrieval by up to 83% compared to baselines under the same error bound and reduces the error by up to 99% under the same bitrate. The source code of IPComp is available at https://github.com/szcompressor/IPComp.

Limitations of the proposed approach: The primary focus of our solution in this paper is to design an effective progressive approach that is universally applicable across different hardware platforms. As a result, our method does not include hardware-specific optimizations yet, such as speed acceleration using tensor cores. Incorporating such optimizations will be part of our future work to further boost the speed of data retrieval by at least 10 time under similar reconstruction quality.

The remaining of the paper is structured as follows. In Section 2, we provide an overview of related work. Section 3 formulates the research problem and the overview of our design. Our developed progressive compressor is detailed in Section 4 to Section 5. Section 6 presents and discusses the evaluation results. Finally, we draw conclusions in Section 7.

2 Related Work

In this section, we survey existing approaches to scientific lossy compression and discuss methods that apply progressive retrieval to enhance these compression techniques.

Scientific lossy compressors aim to reduce data size while guaranteeing that the loss or error remains under a specified threshold required by the applications. Many scientific lossy compressors apply statistical models, including linear regression [38], interpolation [33, 34, 41], and neural networks [14, 31, 32] to predict the value of the variable based on the value of the coordinates. The statistical model will be stored to reconstruct the value such that the storage of original data could be eliminated. To bound the error, the difference between prediction and real value is quantized and

stored together with the model. On the other hand, instead of finding prediction models, some compressors choose to transform data to another domain that is more compressible, and then keep partial of the transformed data based on the error bound. Examples of transformation methods include wavelet transform [27], orthogonal discrete transform [30], and singular value decomposition [4].

Progressive techniques for compression have evolved along two main directions: multi-resolution approaches, which produce output in various sizes, and multi-precision approaches, which generate output in various precisions.

For multi-resolution approaches, researchers have employed tree structures [17, 18], adaptive meshes [5], and wavelet transforms [26] to partition the spatial domain hierarchically, which enables gradual refinement of data resolution. Such techniques are primarily designed for space partitioning and visualization tasks. Although they can support a variety of compression algorithms by applying them to each partitioned block, these methods often suffer from low compression ratios and slower speeds due to the storage and computational overhead introduced by the hierarchical structures. More importantly, they could not bound the error in each data point since the output is smaller than the original.

On the other hand, in the direction of multi-precision retrieval, one straightforward solution is to progressively refine the lossy compression error [35]. Specifically, this involves executing compression multiple times, with each pass compressing the residual error from the previous pass but with a smaller error bound. This progressive strategy is orthogonal to the underlying compression method and offers versatility. However, it does not fully exploit the strengths of the base algorithm, and it incurs high operational costs as decompression must be executed multiple times to achieve a given fidelity level. As a result, researchers also aim to exploit progressive characteristics inherent in specific compression algorithms. PMGARD [28, 39], for instance, is a progressive solution based on the MGARD compressor. Such a solution may lead to suboptimal performance, as shown in Section 6, because its underlying MGARD algorithm has lower compression ratios and speed than other state-of-the-art in many datasets [33, 34, 41].

3 Overview

In this section, we first discuss the problems we are targeting, and then propose our solution for such problems.

3.1 Problem Formulation

3.1.1 Definitions. Here we list five commonly adopted metrics for evaluating scientific lossy compression. The scientific dataset [42] is denoted as x. All symbols used in the paper are explained in Table 1.

Compression Ratio (CR) and Bitrate: CR compares the original data size with compressed data size by $CR = \frac{size(\text{original data})}{size(\text{compressed data})}$. Bitrate is reverse proportional to CR. It measures the average number of bits for storing each scalar value in the compression format. A higher compression ratio or lower bit rate indicates more efficient compression in terms of storage space.

Decompression Error describes the deviation between the original and decompressed data. There are many ways to quantify the deviation, while the L_∞ norm, defined as the maximum point-wise

difference between original and decompressed data, is the most widely used one.

Error Bound (*eb***)** is a user-defined parameter that specifies the maximum allowable error produced by lossy compressors.

Peak Signal-to-Noise Ratio (PSNR) assesses the fidelity of the data with lossy error. The definition is $20 \cdot \log_{10}\left(\frac{\max(x)-\min(x)}{\sqrt{MSE(x,\hat{x})}}\right)$, where $MSE(x, \hat{x})$ denotes the mean squared error between the original dataset x and the decompressed dataset $\hat{x}$. A higher PSNR value indicates better data fidelity.

Table 1: Definition of symbols

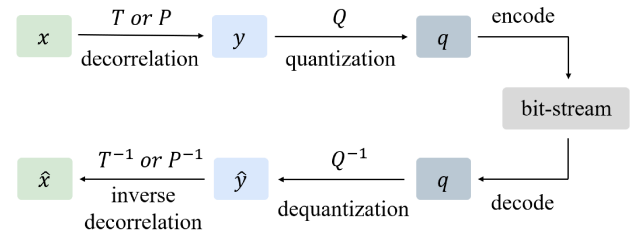| Symbol	Description
n	Numbers of elements of the input
x	The input dataset
$\hat{x}$	The decompressed output with lossy error
$y, \hat{y}$	The decorrelated data and its lossy version
q	The quantized data
eb	The bound of maximum lossy error
T, P	The transform and prediction function
Q	The quantization function
$\|\cdot\|_\infty$	L_∞ norm - the maximum absolute value of the input
V_l	The sublinear space of the l-th level. V_l and V_m are orthogonal to each other when $l \neq m$
Π_l	The L_2 norm projection operator of V_l
x_i, y_i	The subscript i denotes the i-th component of the vector.
x_l	the vector is projected to V_l. $x_l = \Pi_l x$, which applies to any vector with subscription l

Figure 1: A typical lossy compression workflow. T/P represents decorrelation and Q means quantization. The quantization stage is lossy thus $\hat{y}$ is the lossy version of y. Definitions for x, y, etc can be found in Table 1
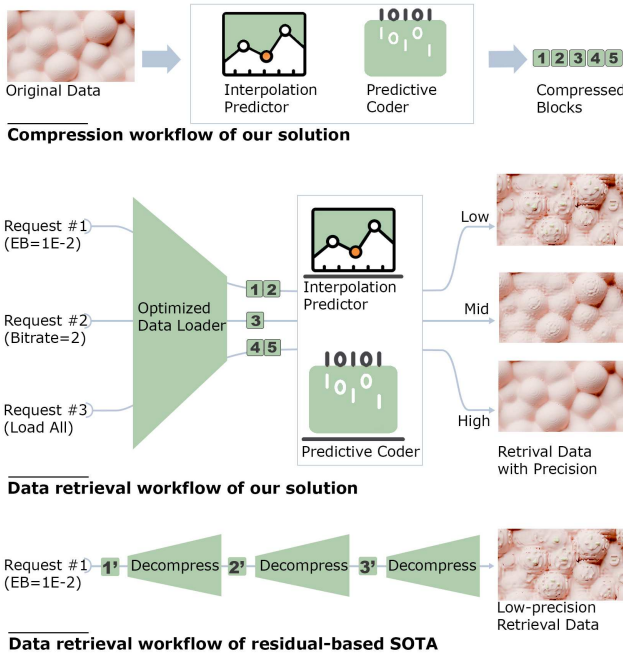
3.1.2 Objectives. The primary objective of this work is to develop a progressive lossy compression framework tailored for scientific data that meets the following four key goals.

- **High compression ratio**: We aim to reach compression ratios higher than or equal to all other SOTAs, as a high compression ratio (CR) is essential for compressors to be effectively utilized in the scientific field. Some progressive designs, such as SZ3-M shown in Section 6, sacrifice compression ratio for progressiveness, but we argue that the low compression ratio will prevent the adoption of such solutions and thus make their progressive capability useless.

- **Progressive retrieval**: We aim to support users to retrieve the output at a lower fidelity level F_1 and incrementally refine it to higher fidelity levels $F_2, \ldots, F_n$ without the need for multiple decompression passes.
- **Fast speed**: Our speed of compression and decompression should be equal to or faster than all other SOTAs, and the decompression process should require only a single pass to retrieve data at any specified fidelity level F_i.
- **Error guarantee**: We guarantee that for each fidelity level F_i, if F_i has error bound restriction of ϵ_i, the reconstruction error E_i satisfies $E_i \leq \epsilon_i$. This ensures that the compressed data remains within acceptable accuracy limits for scientific computations.

To the best of our knowledge, no compressor simultaneously achieves all these objectives yet.

3.2 Overview of IPComp

Compression workflow of our solution

Data retrieval workflow of our solution

Data retrieval workflow of residual-based SOTA

Figure 2: Overall design of our solution IPComp (the compressed data contains multiple decompressible blocks, represented as 1-5 in the diagram)

We introduce IPComp, a progressive lossy compressor designed to efficiently meet the diverse precision and fidelity requirements of scientific applications. IPComp is based on the interpolation algorithm which has been verified as the leading non-progressive reduction method, and we are the first to make it progressive. IPComp mainly has three innovative modules, shown in Figure 2.

- The **interpolation predictor** can reconstruct data to multiple fidelity levels progressively, and only needs a single pass of decompression for each retrieval request.
- The **predictive coder** can compress the interpolation output to independent bitplanes with high compression ratios.

- The **optimized data loader** can determine the minimum set of data for reconstruction to satisfy various fidelity requirements.

We demonstrate the compression and decompression workflow of IPComp in Figure 2. For comparison, we also highlight the differences between IPComp and residual-based SOTAs.

The compression workflow begins with the original dataset, which is processed by an **Interpolation Predictor** to decorrelate the dataset based on spatial and numerical relationships, and quantize the output from floating point format to integers based on the error bound. Following this, the **Predictive coder** encodes the prediction integers by bitplanes into multiple independent blocks, each contributing incrementally to the accuracy of the decompressed data.

The data retrieval workflow is designed to support multi-fidelity retrieval requests efficiently, leveraging an **Optimized Data Loader** that extracts only the required blocks for a given query. Figure 2 demonstrates how the workflow works using three requests in increasing fidelity orders. Requests 1 asks for an error bound of 1E-2, so the data loader retrieves the first few precision blocks (e.g., blocks 1, 2) and passes them to the **Interpolation Predictor** and **Progressive coder**. These blocks are processed together at one single pass to generate low-precision output, suitable for quick exploration or coarse analyses. For the second request which indicates a targeting bitrate of 2, the data loader determines one additional block (e.g., block 3) is optimal for such a request. As a result, the predictor and coder reconstruct this mid-precision representation using block 3 on top of the low-precision result. Finally, the last request asks to retrieve all blocks, so blocks 4 and 5 are loaded consequently to build the high-precision reconstruction.

In contrast, residual-based SOTA approaches involve significant computational overhead and redundant operations. Those approaches will first compress in a large bound, then compress the residual from the last time using a smaller bound, and repeat such residual compression until a targeted bound is reached. As a result, they require multiple decompression passes to fulfill a single retrieval request. For example, in the SOTA workflow of Figure 2, the error bound for blocks $1'$, $2'$, $3'$ are 1, 1e-1, and 1e-2, respectively. For a single request of 1e-2, those approaches need to load three blocks and execute the decompression three times to apply the result on top of each block.

In terms of implementation, our solution is developed using the FZ framework [13], a premier solution for developing scientific compressors. FZ is a comprehensive code platform providing various existing compression-related techniques wrapped in ready-to-use modules, such as the tool to compute lossy error metrics, and the function to parse compression settings from the command line and configure files. Moreover, to help developers with testing and integration, FZ provides a universal and robust API in many languages, including C, C++, Python, and Fortran, as well as seamless integration with I/O libraries like HDF5 [15]. In conclusion, FZ significantly reduces our need to re-implement existing techniques or tools, allowing us to concentrate on designing and implementing innovative progressive compression features.
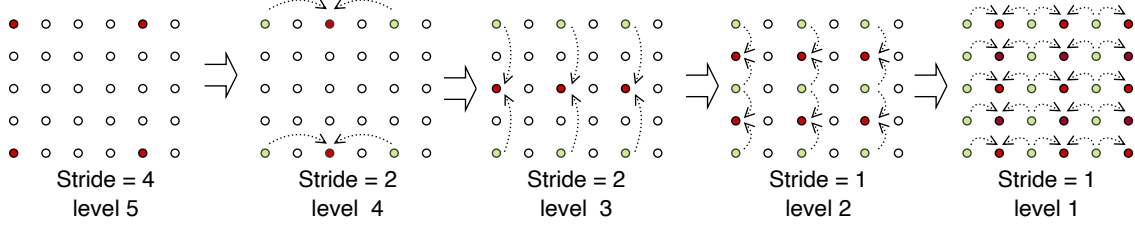
Figure 3: Illustration of how the none-progressive interpolation algorithm works for a 2d input – target points (in red color) are predicted from nearby known points (in green color), indicated by arrows

4 Progressive design of IPComp

In this section, we first analyze the interpolation-based algorithm, which is the leading non-progressive scientific compression strategy, and then we propose our solution to make it progressive by the prediction model and predictive coder.

4.1 Introduction to none-progressive interpolation algorithm

Most scientific lossy compression workflows are composed of three key steps – decorrelation, quantization, and encoding [2, 27, 30, 41]. As shown in Figure 1, the input data is defined as a vector x in a linear space $\mathbb{R}^n$, where n represents the total elements in the dataset. In the decorrelation step, a transform T or prediction function P is applied to x, resolving a decorrelated vector y. After a lossy quantization Q on y, the quantized integers are further lossless coded to bitstream as the final compressed data. Accordingly, the decompression process executes the three steps reversely.

One critical aspect of designing an effective lossy compressor is to construct the best-suit decorrelation algorithm. Our solution IPComp uses interpolation-based decorrelation, which has been proven to be the leading solution in the scientific domain [2, 33, 34, 41]. Its core idea is to estimate unknown data points at fixed relative indices using interpolation formulas, such as linear interpolation and cubic spline interpolation. Unlike many other predictors (such as regression-base ones) that store coefficients for reconstruction, the interpolation approach eliminates such storage overhead by relying on predefined prediction formulas for fixed indices. As a result, this approach achieves much higher compression ratios than many alternatives [41].

In a plain example, consider four data points located equidistantly, with indices $i - 3$, $i - 1$, $i + 1$, and $i + 3$. Their corresponding values are denoted as x_{i-3}, x_{i-1}, x_{i+1}, and x_{i+3}. The goal is to estimate x_i using interpolation formulas.

For linear interpolation [41], the estimation is computed as the average of the neighboring points x_{i-1} and x_{i+1}, given by:

$$y_i = \frac{1}{2}(x_{i-1} + x_{i+1}). \tag{1}$$

For cubic spline interpolation [41], which takes into account all four points to achieve higher accuracy, the estimation can be:

$$y_i = -\frac{1}{16}x_{i-3} + \frac{9}{16}x_{i-1} + \frac{9}{16}x_{i+1} - \frac{1}{16}x_{i+3}. \tag{2}$$

Both of the interpolation formulas predict values using neighboring data points in fixed relative positions, such that the coefficients

are always the same (e.g., $\frac{1}{2}$ in the linear case) and there is no need to save them during compression. Figure 3 shows how to extend the interpolation algorithm from one scalar value to a multi-dimensional dataset. The stride separates the data points by distance (the data distance in stride i is 2^i). In each stride, the algorithm operates recursively along each dimension of the dataset.

4.2 Transform vs. prediction models for progressive design

The interpolation-based decorrelation discussed in Section 4.1 can be employed either as a transform model or a prediction model in lossy compression. In this section, we discuss our rationale for using it as a prediction model in IPComp.

Because we aim to build a progressive lossy compressor, it is crucial to account for the distortion introduced by any lossy operation, as such distortion can accumulate over successive data retrievals. Toward this, we first highlight the differences between the transform and prediction approaches, then provide a theoretical analysis of their respective distortion behaviors, and finally explain why IPComp adopts interpolation as its prediction model.

4.2.1 Transform models. Transform models can be viewed as a linear mapping between original data and decorrelate data. Many transform models, including Fourier transform and wavelet transform, have been adopted in lossy compression. For example, ZFP [30] uses nearly orthogonal block transform, and SPERR [27] is based on the cdf97 wavelet.

Deriving the distortion between input data x and decompressed data $\hat{x}$ in such solutions is relatively straightforward. If measured by L_∞ norm, the distortion can be computed as:

$$\|x - \hat{x}\|_\infty = \|T^{-1}y - T^{-1}\hat{y}\|_\infty \leq \|T^{-1}\|_\infty \|y - \hat{y}\|_\infty$$

The value of $L_\infty(T^{-1})$ depends on the specific transformation function. Taking the basic but widely used transform function $x_i = x_i - x_{i-1}$ as an example, its corresponding transform and inverse transform formulas are:

$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

In this case, the L_∞ norm of (T^{-1}) equals the maximum sum of the rows in the matrix T, which equals the input size n. As a result, any distortion in the transformed domain will be amplified in the

original domain, and the maximum error could become n times the maximum error in the transformed domain. Given that n in most scientific datasets is at least 10^7, such a degree of distortion amplification may be unacceptable for progressive compression.

$$\|x - \hat{x}\|_\infty \leq L_\infty(T^{-1})L_\infty(\hat{y} - y) = n\|\hat{y} - y\|_\infty \quad (3)$$

4.2.2 Prediction models. Prediction models are more complex than transform models, as they tightly couple the decorrelation function P with quantization Q. For transform models, the whole dataset is first transformed by T before applying quantization Q. However, in prediction models, the data usually is split into orthogonal layers or groups, and the prediction P and quantization Q are applied layer by layer [2, 12] on $\hat{x}$ (data with lossy error) instead of x (original data). As a result, the prediction model can be viewed as a non-linear mapping between input and decorrelate data.

Figure 3 shows how to divide the whole linear space into multiple orthogonal levels by setting a shrinking stride distance. We represent level as the sublinear spaces $V_l \subseteq \mathbb{R}^n$. The total number of levels is defined as L. The prediction P_l will be executed $L - 1$ times, covering from the top level L to level 1.

For each level l, we process y_l which is the difference between the original data x_l and its prediction. The prediction P_l is based on interpolation from $\hat{x}_{l+1}$ (the previous level's data with lossy error).

$$y_l = x_l - P_l\hat{x}_{l+1}$$

The prediction difference y_l will be quantized to integers, and dequantized back to floating point $\hat{y}_l$. The quantization guarantees that the point-wise error between y and $\hat{y}_l$ is less than the predefined global error bound eb. Note that in decompression we do not have the original prediction difference y since it is lossy quantized here.

$$y_l - \hat{y}_l = \varepsilon_l, \|\varepsilon_l\|_\infty \leq eb$$

In decompression, which is the reverse process, we first recover the prediction data $P_l\hat{x}_{l+1}$, then compute the output $\hat{x}$ by adding $\hat{y}$ (the prediction difference with error) on top of the prediction.
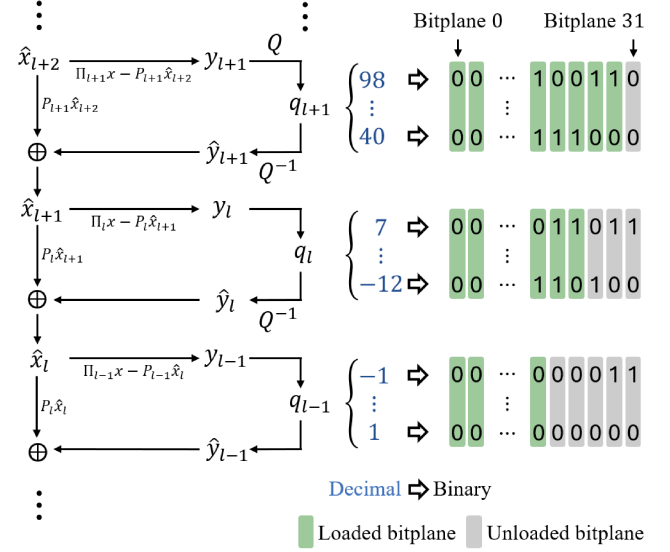
$$\hat{x}_l = P_l\hat{x}_{l+1} + \hat{y}_l$$

By simplifying the three formulas above, we can get the conclusion that for each level, $\hat{x}_l$ only differs with original data x_l within the point-wise error bound eb.

$$x_l = \hat{x}_l + \varepsilon_l, \|\varepsilon_l\|_\infty \leq eb$$

Then for the whole dataset, the maximum point-wise error could be bounded by

$$\|x - \hat{x}\|_\infty = \max_l \|x_l - \hat{x}_l\|_\infty = \max_l \varepsilon_l \leq eb \quad (4)$$

Compared with this Equation (4), the error in transform models (shown in Equation (3)) is proportional to the data size, which means it will be very difficult to bound the error in transform models if the input data is large. Since progressive may cause the error to accumulate across retrievals, we want to have an error control as tight as possible. As a result, we choose to use interpolation as a prediction model for our progressive solution.

Figure 4: Our progressive solution splits the quantization integers by bitplanes and encodes them separately

4.3 Progressive interpolation algorithm

In this section, we present our solution to support progressive retrieval utilizing the interpolation-based prediction model discussed in Section 4.1 and Section 4.2.

The foundation of our progressive design is to load a partial of the information of $\hat{y}$ according to the bitplane. As illustrated in Figure 4, in our approach, the quantized data (denoted as q_l at level l) are 32-bit integers. The bits from the same position across multiple quantized integers form a bitplane (represented by small rectangular boxes in the figure). We independently encode the 32 bitplanes at the same level, which allows lower-fidelity outputs to be reconstructed by loading only some of the bits rather than all of them. When retrieving data, our optimizer discussed in Section 5 will determine the optimal loading strategy – the minimum bitplanes to load from each level to meet the retrieval requirement.

Algorithm 1 describes the reconstruction process of our solution. As shown in Figure 4, the reconstruction starts from the top level, where the loaded data is decoded into the reconstructed diff $\hat{y}$. We then use the $\hat{y}$ from each level, in sequential order, to progressively reconstruct the original data. The parameter L_p indicating in which level we start progressive compression. The first step from line 3 to 7 is to retrieve data of non-progressive levels. Then the second part is to load the bitplanes we are requiring, and decoded them in addition to the vector we get in the first step.

Algorithm 1 is designed for cases where data is reconstructed from scratch. On the other hand, when users find the current precision is insufficient, Algorithm 2 shows how to update the output from lower precision to higher precision by loading incremental bitplanes. By combining these newly loaded bitplanes with the previously loaded data, our solution can reconstruct a higher fidelity output without reloading all the compressed data.

Algorithm 1 Reconstruction Algorithm

Require: $bitplaneList[L]$
1: $\hat{x} \leftarrow 0$
2: $\Pi_L \hat{x} \leftarrow P_L(\mathbf{0})$
3: **for** $l \leftarrow L - 1$ **downto** $L_p + 1$ **do**
4: $\quad q_l \leftarrow \textbf{decode}(bitplaneList[l])$
5: $\quad \hat{y}_l \leftarrow \textbf{dequantizition}(q_l)$
6: $\quad \hat{x}_l \leftarrow \textbf{Predict}(\hat{x}, \hat{y}_l)$
7: **end for**
8: $\Delta \leftarrow 0$
9: **for** $l \leftarrow L_p$ **downto** 1 **do**
10: $\quad q_l \leftarrow \textbf{decode}(bitplaneList[l])$
11: $\quad \hat{y}_l \leftarrow \textbf{dequantizition}(q_l)$
12: $\quad \hat{x}_l \leftarrow \textbf{Predict}(\Delta, \hat{y}_l)$
13: $\quad \Delta_l \leftarrow \textbf{Predict}(\Delta, \hat{y}_l)$
14: **end for**
15: **return** $\hat{x}$

Algorithm 2 Incremental Reconstruction Algorithm

Require: $\hat{x}^{(\text{old})}, bitplaneList[L]$
1: $\hat{x}^{(\text{new})} \leftarrow \hat{x}^{(\text{old})}$
2: $\Delta \leftarrow 0$
3: **for** $l = L_p$ **to** 1 **do**
4: $\quad q_l \leftarrow \textbf{decode}(bitplaneList[l])$
5: $\quad \hat{y}_l = \textbf{dequantization}(q_l)$
6: $\quad x_l^{(\text{new})} \leftarrow \textbf{Predict}(\Delta, \hat{y}_l)$
7: $\quad \Delta_l \leftarrow \textbf{Predict}(\Delta, \hat{y}_l)$
8: **end for**
9: **return** $\hat{x}^{(\text{new})}$

4.4 Predictive negabinary Coding

In this section, we propose a novel coding method that addresses two key challenges in progressive coding: preserving data correlation across bitplanes and handling sign bits. Our solution achieves high compression ratios by capturing cross-bitplane correlations using predictive coding and encoding sign bits using negabinary coding.

4.4.1 Predictive Bitplane Coding. We propose a predict-based strategy to exploit the correlation between bitgroups. To support progressive retrieval, each quantized integer is split into bitplanes, which means that the bits of a single integer are encoded separately. As a result, the correlation between bits from the same integer is totally ignored. We observe that during decompression when retrieving a certain bitplane at a specific level, the previously loaded bitplanes are already known. This observation allows us to leverage the correlation between bitplanes by predicting the bit value based on previous bits from the same integer, and encode the results of the prediction instead of raw bits. For example, if the prefix bit is 0, we predict that the current bit b will also be 0. This prediction mechanism is equivalent to applying an XOR operation between the prefix bit and b as encoded bit = prefix bit $\oplus$ b.

We can further extend this method by utilizing more prefix bits – we perform an XOR operation on all the preceding bits and then XOR the result with the target bit b to obtain the encoded bit. The

Table 2: Our predictive bitplane coding strategy reduces the entropy in quantized integers (lower entropy indicates better compressibility)

Fields	Original	1-bit prefix	2-bits prefix	3-bits prefix
Density	0.358505	0.352111	0.34732	0.350525
SpeedX	0.868908	0.861458	0.855473	0.859443
Wave	0.440292	0.433391	0.427993	0.431599

process can be mathematically expressed as:

$$\text{encoded\_bit} = (\text{prefix\_bit}_1 \oplus \text{prefix\_bit}_2 \oplus ... \oplus \text{prefix\_bit}_n) \oplus b$$

We observe from Table 2 that using 1-bit, 2-bit, or 3-bit prefix bits for prediction consistently reduces the entropy compared to the original data. Among these, 2-bit prediction generally achieves the best entropy drop. As a result, our approach leverages the two prefix bits to predict the next bit.

4.4.2 Negabinary coding. Unlike their non-progressive counterparts, progressive compression schemes must efficiently handle sign bits, since loading them is a prerequisite for reconstructing values, but storing sign bits separately and loading them first before all other bits incurs significant overhead. To address this issue, we evaluate the suitability of three widely used methods for encoding sign bits in a progressive setting – two's complement [36], sign-magnitude [36], and negabinary [19, 30], and select negabinary encoding as the preferred approach for two reasons.

First, negabinary encoding may lead to better compressibility than other methods for values that fluctuate around zero (which is predominant in the last layer of quantized integers). In two's complement, the higher-order bits often contain many 1s for values near zero, making the corresponding high-order bitplanes difficult to compress. For example, the 8-bit representations of 1 and -1 are 00000001 and 11111111 in two's complement, 00000001 and 10000001 in sign-magnitude, and 00000001 and 00000011 in negabinary. As a result, negabinary encoding keeps the higher-order bits as 0, which leads to more compressible high-order bitplanes compared to the dense 1s in two's complement. Second, compared to sign-magnitude encoding, negabinary encoding is more balanced. When the least significant bits are set to zero (due to quantization or truncation), the resulting error uncertainty is smaller in negabinary encoding. This leads to more predictable error behavior, which is beneficial for data compression and reconstruction. The uncertainty could be formulated by

$$\text{uncertainty (negabinary)} = \begin{cases} \frac{2}{3} \times 2^d - \frac{1}{3}, & \text{when } d \text{ is odd} \\ \frac{2}{3} \times 2^d - \frac{2}{3}, & \text{when } d \text{ is even} \end{cases}$$

$$\text{uncertainty (sign-magnitude)} = 2^d - 1$$

where d represents the number of bits discarded. As d increases, the uncertainty of negabinary is only around two-thirds of sign-magnitude encoding.

5 Optimized data loading

In this section, we present our optimized data load strategy that can minimize the volume of data loaded under the retrieval constraints. Our optimizer supports both error-bound mode and bitrate mode which represents most of the scenarios in scientific data retrieval.

- Error bound mode: in this mode, users specify the required precision and the optimizer selects the optimal strategy to ensure that the reconstructed data's error remains strictly within the given bound while minimizing the amount of data loaded.
- Fixed rate/size mode: in this mode, the user specifies a maximum allowable bitrate or size, typically constrained by I/O bandwidth. The optimizer then ensures that the reconstructed data has the smallest possible error while staying within the specified bitrate limit. Since the total data size n is constant for the input, requiring a specific bitrate is equivalent to requiring a fixed retrieval size.

We first discuss the upper bound theory which both modes rely on, then show how the optimizer makes the best decisions on bitplane selection based on dynamic programming.

5.1 Fundamental theory for our optimizer

In this section, we discuss the fundamental theory for our optimizer which measures the error caused by only a partial of bitplanes being loaded.

THEOREM 1. *The L_∞ error in progressive retrieval can be bounded based on the information loss due to the unloaded bitplanes.*

$$\|x - \hat{x}\|_\infty \leq \sum_{l=0}^{L-1} p^l \|\delta y_{l+1}\|_\infty + eb \qquad (5)$$

In this equation, $p = 1$ for linear interpolation, and $p = 1.25$ for cubic interpolation. δy_l describes the information loss in level l caused by the unloaded bitplanes, and its value can be pre-computed during compression.

PROOF. As illustrated in Algorithm 1, the retrieval process proceeds iteratively through all levels, with each level relying on the preceding level's output to inform its predictions. This introduces a significant hurdle for error analysis, as uncertainty can propagate from one level to the next. To address this, we conceptualize the error in level i as being comprised of three components. The first component, δy_l, represents the information loss resulting from not loading certain bitplanes at this level. The second component, $\delta y_{prop,l}$, accounts for information loss carried over from the prior level. The third component, ε_l, captures the error introduced by lossy quantization.

$$x_l - \hat{x}_l = \delta y_l + \delta y_{prop,l} + \varepsilon_l \qquad (6)$$

The information loss propagated until the previous level $l + 1$ is $\delta y_{prop,l+1}$. It will be first combined with the information loss in the current level δy_{l+1}, and then transfered by prediction and passed to the next level as $\delta y_{prop,l}$.

$$\delta y_{prop,l} = P_l(\delta y_{prop,l+1} + \delta y_{l+1}) \qquad (7)$$

Combine the two equations, we can have:

$$x_l - \hat{x}_l = \delta y_l + \sum_{m=0} P_l P_{l+1}...P_{l+m} \delta y_{l+m+1} + \varepsilon_l \qquad (8)$$

In our approach, the L_∞ of interpolation P_l is greater or equal to 1, which implies the information loss would be amplified to the lower level. In other words, lower levels have larger errors, and the

maximum error of the dataset is in the first level. As a result, we can quantify the error of the dataset with Equation (8) by:

$$\|x - \hat{x}\|_\infty = \|x - \hat{x}_1\|_\infty$$
$$= \|\delta y_1 + \sum_{l=0} P_1 P_{1+1}...P_l \delta y_{l+1} + \varepsilon_1\|_\infty$$
$$\leq \sum \|P_1\|_\infty \|P_2\|_\infty...\|P_l\|_\infty \|\delta y_{l+1}\|_\infty + eb \qquad (9)$$

Now we simplify Equation (9) to Equation (5). The value of $L_\infty(P)$ depends on the prediction method. Liner interpolation, as shown in Equation (1), has two coefficients both at 0.5. Such that $L_\infty(P) = 0.5 + 0.5 = 1$. Similarly, cubic interpolation, as shown in Equation (2), has four coefficients, and $L_\infty(P) = 2 \times \frac{1}{16} + 2 \times \frac{9}{16} = 1.25$. With the derived value of $L_\infty(P)$, Equation (9) is converted to Equation (5).

5.2 Optimized loading based on error-bounds

The goal of error-bound mode is to generate a loading strategy specifying the minimum number of bitplanes to load in each layer, in order to let all point-wise lossy errors be less or equal to the user-defined bound (denoted as E). Finding such bitplanes can be formalized as an optimization problem.

$$\max_{b_l, l \in \{1,2,...L\}} \sum_l SavedSize(l, b_l),$$
$$\text{subject to } \sum_l err(l, b_l) + eb \leq \text{E}.$$

b_l indicates the number of bitplanes discarded at the level l. $SavedSize(l, b_l)$ denotes the amount of data saved by discarding b_l bitplanes at level l. $err(l, b_l)$ denotes the error at level l by discarding b_l bitplanes. As stated in Section 4.4.2, $err(l, b_l) = p^{l-1} \|\delta y_l\|_\infty$, where $p = 1$ for linear interpolation, $p = 1.25$ for cubic interpolation, and δy_l is a function of b_l.

By such definitions, we essentially reformulate this optimization problem as the classical knapsack problem, enabling it to be efficiently solved using dynamic programming (DP) with minimal computational overhead. Let $DP(l, e)$ be the maximum total saved size when the last layer is l and the maximum error is e. Then $DP(L, E)$ would be our best solution to load data for the given error bound E, and its value can be derived recursively by the following DP transition function:

$$DP(l, e) = \max_{b_l, s.t.err(l, b_l) \leq e} \{DP(l-1, e - err(l, b_l)) + SavedSize(l, b_l)\}$$

The time complexity of such a DP process is $O(\#level \times \#bitplane \times \#discrete\ error\ values)$. The number of levels is $\log_2 n$ where n is the input size, and the number of bitplanes is 32. The discrete error values fall within the range of $[128, 1023]$ by normalized retrieval bound E by compression bound eb. If comparing the time of the DP to the time of simply traversing the input, the ratio would be $\frac{(1023-128+1)*32*\log_2 n}{n} \approx \frac{3*10^4 \log_2 n}{n} \approx 3\%$ for datasets listed in Table 3. Since compression takes much more computation than just traversing the input, the overhead of the DP relative to compression is totally negligible.

5.3 Optimized loading based on bitrates

In the fixed bitrate mode, the user specifies a maximum allowed retrieval size S (or equivalently, a maximum allowable bitrate). The optimizer then seeks to minimize the reconstruction error while ensuring that the loaded data does not exceed S.

With $LoadedSize(l, b_l)$ representing the loaded size of level l, this problem can be formalized as follows:

$$\min_{b_l, l \in \{1,2,...L\}} \quad \sum_l err(l, b_l) + eb,$$

$$\text{subject to} \quad \sum_l LoadedSize(l, b_l) \leq S.$$

Similar with Section 5.2, we effectively convert this problem to the knapsack dynamic programming problem, so that it can be solved similarly as in the error bound mode, also with negligible overhead.

6 Experimental Evaluation

In this section, we describe the experimental setup and evaluate our solution on six datasets against four state-of-the-art baseline compressors.

6.1 Experimental Setting

6.1.1 Execution Environment. The experiments are performed on the Purdue Anvil supercomputer [37] through NSF ACCESS [6]. Each computing node in Anvil features two AMD EPYC 7763 CPUs with 64 cores at a 2.45GHz clock rate and 256 GB DDR4-3200 RAM.

Table 3: Data in our experiments

Name	Explanation	Precision	Shape
Density [42]	mass per unit volume in turbulence	64	$256 \times 384 \times 384$
Pressure [42]	thermodynamic pressure in turbulence	64	$256 \times 384 \times 384$
VelocityX [42]	x-direction velocity in turbulence	64	$256 \times 384 \times 384$
Wave [41]	wavefield evolution in seismic	64	$1008 \times 1008 \times 352$
SpeedX [42]	x-direction wind speed in weather	64	$100 \times 500 \times 500$
CH4 [42]	mass fraction of CH4 in combustion	64	$500 \times 500 \times 500$

6.1.2 Datasets. The experiments are evaluated on six datasets across four diverse scientific domains, as listed in Table 3.

6.1.3 State-of-the-Art lossy compressors in our evaluation. The experiments include four SOTA scientific lossy compressors as baselines – SZ3-M, SZ3-R, ZFP-R, and PMGARD.

SZ3-M [39]: SZ3-M (where "M" stands for multi-fidelity) is the straightforward multi fidelity version of SZ3 based on multiple outputs. SZ3 is the leading non-progressive scientific lossy compressor. SZ3 uses interpolation as prediction, together with linear-scale quantization, Huffman coding, and zstd lossless coding [7]. SZ3 has excellent compression ratio and fidelity over others while its speed may not be as fast as ZFP. SZ3-M compresses the input with different error bounds independently and groups those compressed data all together as output. Such a solution supports multi-fidelity retrieval but is not progressive yet, as it cannot reuse the low-fidelity data to build high-fidelity results.

SZ3-R [35, 39]: SZ3-R (where "R" stands for residual) is the progressive version of SZ3 based on residuals. It first compresses the input with a large bound, then compresses the lossy error (residual)

from the last time using a smaller bound, and repeats such residual compression until a targeted bound is reached.

ZFP-R [35]: ZFP-R is the progressive version of ZFP based on residuals. It handles residuals the same way as SZ3-R. ZFP [30] is the leading transform-based lossy compressor based on orthogonal transformation. ZFP is usually the fastest scientific lossy compressor because of its highly efficient transform function, although its compression ratio may not be as high as others.

PMGARD [28, 39]: PMGARD (where "P" stands for progressive) is the progressive version of the MGARD [28] compressor. MGARD [2, 29] is a multigrid-based lossy compressor that uses hierarchical decomposition to remove redundancy in scientific data while preserving error bounds.
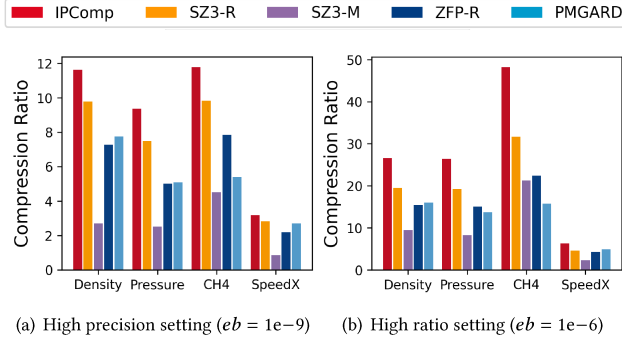
We note that the residual-based approaches (SZ3-R and ZFP-R) have drawbacks that affect our evaluation. First, they have limited error-bound flexibility. More specifically, the retrieval is only possible at a few predefined error bounds. This creates a trade-off – setting too many error bounds reduces the overall compression ratio and significantly degrades compression/decompression throughput, while setting too few nodes limits flexibility in selecting error bounds during decompression. For the experiments, we configure five error bounds for them, where each successive one is a factor of 2^2 (4×) apart. Specifically, the error bounds are set to $2^{16}eb$, $2^{14}eb$, $2^{12}eb$, $2^{10}eb$, $2^8 eb$, $2^6 eb$, $2^4 eb$, $2^2 eb$, and eb. Second, those methods are not well-suited for retrievals based on pre-defined bitrates, as the residuals are compressed by error bounds, and their size is not aligned with the bitrate of the retrieval requests. As a result, we select the largest residual that fits within the user-specified bitrate constraint for SZ3-R and ZFP-R in the experiments.
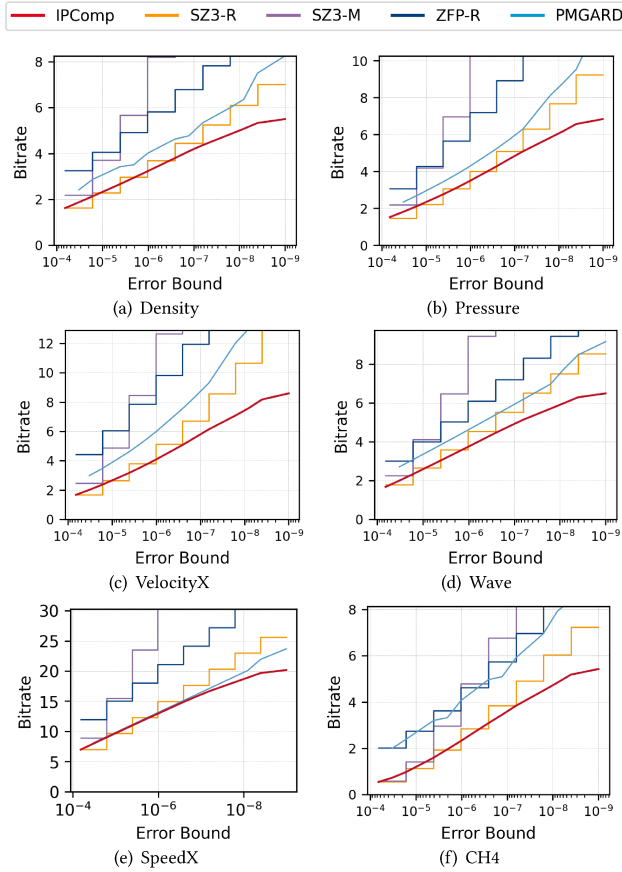
6.2 Evaluation Results and Analysis

The evaluation covers four aspects – compression ratio, progressive retrieval effectiveness under error-bound or bitrate constraints, compression and decompression/retrieval speed, and retrieval quality. To be more specific, the compression ratio comparison demonstrates our solution IPComp leads to the smallest compressed data size. The progressive retrieval evaluation shows that IPComp requires the lowest data volume to reconstruct to the same fidelity compared with others, and IPComp leads to the highest reconstruction fidelity under the same bit rate budget. The speed test confirms IPComp's high compression efficiency, especially compared to residual-based alternatives (SZ3-R and ZFP-R) whose speed drops significantly when the number of residual compressions increases.

6.2.1 Compression ratios. CR is a key metric for evaluating any compressor, regardless of whether it is progressive or not. As discussed in Section 1, the fundamental goal of compression is to reduce data size, as smaller data sizes alleviate the burden of storage, transfer, and processing on applications.

Figure 5 presents the compression ratios of all solutions under the same error-bound settings. We select two error bounds which are 1e-6 and 1e-9 to cover both high precision (cr<10) and high ratio (cr up to 50) cases. As this figure shows, our solution IPComp achieves a compression ratio advantage of around 20% to 500% over other progressive compressors across the vast majority of datasets. This allows users to perform progressive decompression while utilizing the least possible storage space.
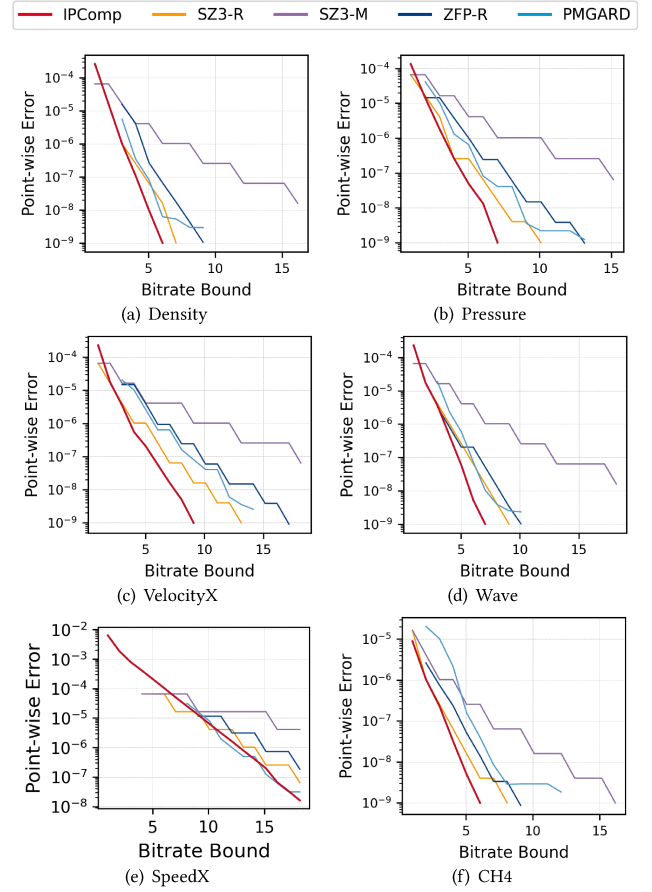
(a) High precision setting ($eb = 1e-9$) (b) High ratio setting ($eb = 1e-6$)

Figure 5: Our compressor IPComp leads the compression ratio among all baselines

(a) Density

(b) Pressure

(c) VelocityX

(d) Wave

(e) SpeedX

(f) CH4

Figure 6: Our compressor IPComp takes the smallest amount of data to reconstruct toward the same L_∞ error compared with all the baselines. Moreover, IPComp supports arbitrary error bound, while SZ3-R and ZFP-R are limited to a few pre-defined bounds

In fact, although not shown in the figure, IPComp achieves an even higher compression ratio than the non-progressive SZ3 particularly in high-precision scenarios, despite both being based on the interpolation prediction algorithm. The primary reason for this

improvement lies in the encoding process. The non-progressive SZ3 uses Huffman entropy encoding, which is then further compressed using zstd. Since Huffman coding assigns variable-length codes based on frequency, it may disrupt certain repetitive patterns in the byte or word level after encoding. This, in turn, reduces the effectiveness of zstd's lossless compression, as zstd relies on detecting and exploiting repetitive patterns at the byte/word level to achieve higher compression efficiency. [7] In contrast, our solution IPComp employs a customized predictive encoding method, discussed in Section 4.4, followed by zstd for entropy encoding and pattern extraction. By avoiding the disruption caused by Huffman and preserving more repetitive structures, our method enhances the final compression ratio.

(a) Density

(b) Pressure

(c) VelocityX

(d) Wave

(e) SpeedX

(f) CH4

Figure 7: Under given bitrate budget, our solution IPComp reconstructs the highest fidelity output than all the baselines (a lower lossy error indicates a high fidelity)

6.2.2 Progressive retrieval efficiency. After checking the compression ratio, next we evaluate the decompression or retrieval efficiency. Since users may want retrieval by fidelity (specified as error bounds) or size (specified as bitrate), our evaluation is split into two figures for those two scenarios.
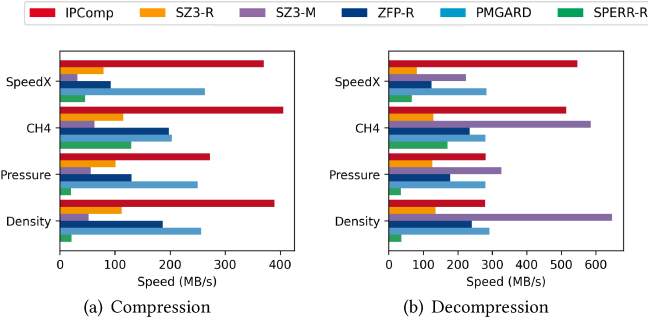
Figure 6 demonstrate the evaluation results of the error bound mode. Compressors with higher efficiency should result in smaller

data retrieval size under the error restrictions. The retrieval size is shown as bitrate in Figure 6, so lower lines in the figure indicate better results.
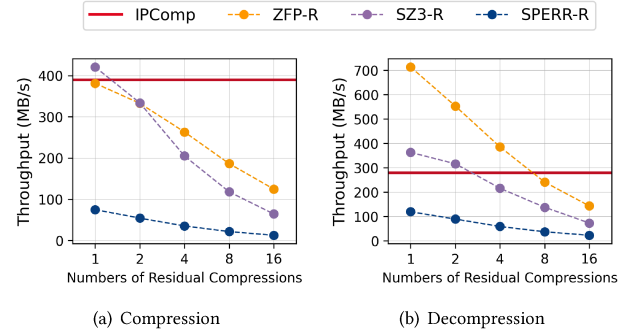
Figure 7 presents the results in fixed bitrate mode. Compressors with higher efficiency should result in higher fidelity (indicated by lower error) with the same retrieval size, still lower lines in the figure indicate better results. Since our solution is the only progressive compressor that directly supports the fixed bitrate mode, manual efforts are needed to test other baselines. We enable fix bitrate mode for residual-based compressors by selecting the largest anchor point that fits within the user-specified bitrate constraint. Similarly, for PMGARD, we manually define anchor points ranging from $2^{16}eb$, $2^{15}eb$, $2^{14}eb$ down to eb, to allow for bitrate-based decompression.

As shown in both Figure 6 and Figure 7, our compressor IPComp demonstrates its superior compression efficiency by consistently achieving the smallest data load size under the same max error, and the lowest max error under the same bitrate budget. This advantage primarily comes from the predictive coding method discussed in Section 4.4 and the optimized loading strategy discussed in Section 5. The residual-based solutions SZ3-R and ZFP-R have a staircase line in both of the figures because the fidelity of such solutions is limited to a few pre-defined residual levels. In comparison, our solution provides higher flexibility as it supports retrieval on arbitrary fidelity and bitrate.

(a) Compression   (b) Decompression

Figure 8: Our solution IPComp has the fastest speed in both compression and decompression than all others except for SZ3-M which is multi fidelity but not progressive

6.2.3 Speed.
We evaluate the compression and retrieval speed of all the compressors. To ensure a fair comparison, we use $eb = 10^{-9} \times \text{Range}(dataset)$ to compress and retrieve the data for all compressors, except for PMGARD as it is a lossless compress with lossy retrieval by design. To reach the target eb, residual-based compressors need to compress and decompress in multiple iterations. The speed tests are shown in Figure 8. It confirms our solution IPComp is up to around 300% faster than other approaches in most cases, except for SZ3-M. SZ3-M supports multi fidelity by compressing input in different error bounds and storing all those outputs together. Such a solution is not progressive since the retrieval cannot use previous passes of data. As a result, its compression ratio and retrieval efficiency are both extremely limited in the experiments. We include one additional compressor SPERR-R in this figure. SPERR-R is the residual progressive implementation of the wavelet-based

(a) Compression   (b) Decompression

Figure 9: The speed of residual-based progressive solutions (SZ3-R, ZFP-R, etc) will drop significantly when the setting of residual count increases

lossy compressor SPERR [27]. The speed of SPERR-R is extremely slow – less than 50 MB/s in more than half of the cases. Such a slow speed will offset the savings of reduced data size for I/O, and potentially slow down the scientific workflow. As a result, we do not include SPERR-R as one of our baselines in the full evaluation.
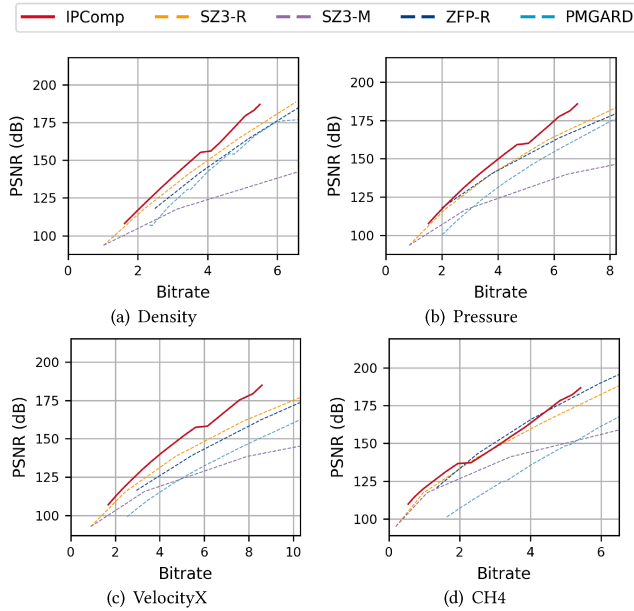
Another evaluation in this section is the speed of residual-based compressors. As discussed in Section 6.1.3, residual-based ones must compress and decompress multiple times based on the number of pre-defined error bounds. More pre-defined error bounds would provide more flexibility in retrieval, however, as Figure 9 shows, this will instead reduce their speed significantly. We also observe from Figure 9 that the residual results are curved instead of straight lines. The reason is that although having more predefined error bounds increases the number of iterations, each iteration takes less time because the looser bounds result in a smaller range of quantized integers. However, the total time still increases significantly due to the cumulative effect of all iterations. This drawback of residual-based compressors highlights the advantages of our solution which can deliver highly flexible retrieval at high speed.

6.2.4 Retrieval quality.
The quality of retrieved data can be assessed in various metrics, including L_∞ norm, PSNR, error correlation, and visualization. Section 6.2.2 covers the L_∞ norm, and this section focuses on the rest three metrics.

PSNR is mathematically related to the L_2 norm of error. Although our design primarily targets the L_∞ norm and we do not explicitly optimize for PSNR, Figure 10 shows our approach still maintains competitive or superior PSNR compared to other baselines.

Table 4: Comparison of retrieval error's correlation lengths across compressors. A shorter length indicates less correlated errors thus better quality. (ZFP cannot achieve ratio below 1.0% on the Density data)

Retrived Ratio	0.1%	0.3%	1.0%	3.0%
IPComp	2.586	2.174	1.311	1.131
SZ3	7.5669	3.030	1.439	1.160
ZFP	–	–	1.316	1.172
SPERR	1.542	1.162	1.189	1.150

(a) Density

(b) Pressure

(c) VelocityX

(d) CH4

Figure 10: Our solution leads to higher PSNR under the same bitrate retrieved in most of the cases compared with other solutions

Error correlation measures the coherence of errors introduced by lossy retrieval – lower correlation implies more localized, less structured errors and thus better retrieval quality. We capture this via the correlation length defined as the radial distance at which the linearly interpolated autocorrelation of the retrieval error first falls below a threshold. Consequently, a shorter correlation length denotes fewer correlated errors and superior retrieval quality. As Table 4 shows, increasing the volume of retrieved data reduces the correlation length, and IPComp achieves consistently shorter lengths than the non-progressive interpolation-based SZ3.

The visualization results of our solution are shown in Figure 11. We load 0.1%, 0.3%, and 1.0% from the same data, and assess the impact on visualization of Curl and Laplacian operations. While loading 0.3% of data is enough for Curl in terms of visualization, 1% of data is needed for Laplacian. This confirms the necessity of progressive retrieval in scientific applications.

7 Conclusions and Future Work

In this work, we present IPComp, an interpolation-based progressive lossy compression solution designed to address the growing need for efficient scientific data storage and retrieval. Our approach accomplishes progressiveness effectively with an interpolation prediction model, multi-level bitplanes, and predictive coding techniques. It is equipped with an optimizer to minimize the data volume during retrieval under given error bound or bitrate targets.

Experimental evaluations conducted on six real-world scientific datasets demonstrate the effectiveness of our solution. IPComp consistently achieves the fastest speed, the highest compression ratios, the lowest data retrieval volume, and the highest data fidelity compared to state-of-the-art alternatives. Additionally, compared with residual-based solutions that only support limited retrieval

(a) Curl (0.1% retrieved)

(b) Laplace (0.1% retrieved)

(c) Curl (0.3% retrieved)

(d) Laplace (0.3% retrieved)

(e) Curl (1% retrieved)

(f) Laplace (1% retrieved)

Figure 11: The visualization of two post-analysis metrics Curl and Laplace on the same Density data. Loading 0.3% is fine for Curl but Laplace requires 1% – demonstrating the necessity of progressive retrieval

options, our approach is very flexible on fidelity control as it takes arbitrary error bounds and bitrates as retrieval options.

Our findings suggest that IPComp represents a significant advancement in progressive lossy compression and is a practical choice for scientific applications. The future work will focus on optimizing hardware acceleration (e.g., GPU and tensor cores), integrating with scientific workflows like HDF5, and expanding large-scale HPC evaluations. These improvements will further refine IPComp for broader application scenarios.

8 Acknowledgments

References

[1] Sameh Abdulah, Allison H. Baker, George Bosilca, Qinglei Cao, Stefano Castruccio, Marc G. Genton, David E. Keyes, Zubair Khalid, Hatem Ltaief, Yan Song, Georgiy L. Stenchikov, and Ying Sun. 2024. Boosting Earth System Model Outputs And Saving PetaBytes in Their Storage Using Exascale Climate Emulators. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis* (Atlanta, GA, USA) *(SC '24)*. IEEE Press, Article 2, 12 pages. doi:10.1109/SC41406.2024.00008

[2] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. 2018. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science* 19, 5 (01 Dec 2018), 65–76.

[3] Venkatramani Balaji, Karl E Taylor, Martin Juckes, Bryan Lawrence, Paul Durack, Michael Lautenschlager, Chris Blanton, Luca Cinquini, Sebastien Denvil, Mark Elkington, Francesca Guglielmo, Eric Guilyardi, David Hassell, Slava Kharin, Stephan Kindermann, Sergey Nikonov, Aparna Radhakrishnan, Martina Stockhause, Tobias Weigel, and Dean Williams. 2018. Requirements for a global data infrastructure in support of CMIP6. *Geoscientific Model Development* 11 (09 2018), 3659–3680. doi:10.5194/gmd-11-3659-2018

[4] Rafael Ballester-Ripoll, Peter Lindstrom, and Renato Pajarola. 2019. TTHRESH: Tensor Compression for Multidimensional Visual Data. https://arxiv.org/abs/1806.05952. *IEEE Transaction on Visualization and Computer Graphics* 26, 09 (sep 2019), 2891–2903. Issue 9. Online.

[5] Harsh Bhatia, Duong Hoang, Nate Morrical, Valerio Pascucci, Peer-Timo Bremer, and Peter Lindstrom. 2022. AMM: Adaptive Multilinear Meshes. *IEEE Transactions on Visualization and Computer Graphics* 28, 6 (June 2022), 2350–2363. doi:10.1109/TVCG.2022.3165392

[6] Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. 2023. ACCESS: Advancing Innovation: NSF's Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 173–176. doi:10.1145/3569951.3597559

[7] Yann Collet and Murray Kucherawy. 2018. Zstandard Compression and the application/zstd Media Type. RFC 8478. doi:10.17487/RFC8478

[8] Coupled Model Intercomparison Project. 2024. https://wcrp-cmip.org/. Online.

[9] Coupled Model Intercomparison Project Phase 6 dashboard. 2024. http://esgf-ui.cmcc.it/esgf-dashboard-ui/federated-view.html. Online.

[10] cuZFP. 2019. https://github.com/LLNL/zfp/tree/develop/src/cuda_zfp. Online.

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

[12] Sheng Di, Jinyang Liu, Kai Zhao, Xin Liang, Robert Underwood, Zhaorui Zhang, Milan Shah, Yafan Huang, Jiajun Huang, Xiaodong Yu, et al. 2024. A Survey on Error-Bounded Lossy Compression for Scientific Datasets. *arXiv preprint arXiv:2404.02840* (2024).

[13] FZ Lossy Compression Framework. 2024. https://fzframework.org//. Online.

[14] Jun Han and Chaoli Wang. 2023. CoordNet: Data Generation and Visualization Generation for Time-Varying Volumes via a Coordinate-Based Neural Network. *IEEE Transactions on Visualization and Computer Graphics* 29, 12 (2023), 4951–4963. doi:10.1109/TVCG.2022.3197203

[15] HDF5. 2024. HDF5. https://www.hdfgroup.org/solutions/hdf5/. Online.

[16] Duong Hoang, Harsh Bhatia, Peter Lindstrom, and Valerio Pascucci. 2021. High-Quality and Low-Memory-Footprint Progressive Decoding of Large-Scale Particle Data. In *2021 IEEE 11th Symposium on Large Data Analysis and Visualization (LDAV)*. 32–42. doi:10.1109/LDAV53203.2021.00011

[17] Duong Hoang, Harsh Bhatia, Peter Lindstrom, and Valerio Pascucci. 2023. Progressive Tree-Based Compression of Large-Scale Particle Data. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–18. doi:10.1109/TVCG.2023.3260628

[18] Duong Hoang, Harsh Bhatia Brian Summa, Peter Lindstrom Pavol Klacansky, Will Usher, Peer-Timo Bremer, , and Valerio Pascucci. 2021. Efficient and Flexible Hierarchical Data Layouts for a Unified Encoding of Scalar Field Precision and Resolution. *IEEE Trans Vis Comput Graph.* 27, 2 (2021), 603–613.

[19] Duong Hoang, Pavol Klacansky, Harsh Bhatia, Peer-Timo Bremer, Peter Lindstrom, and Valerio Pascucci. 2018. A study of the trade-off between reducing precision and reducing resolution for data analysis and visualization. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 1193–1203.

[20] Jiajun Huang, Sheng Di, Xiaodong Yu, Yujia Zhai, Jinyang Liu, Yafan Huang, Ken Raffenetti, Hui Zhou, Kai Zhao, Zizhong Chen, Franck Cappello, Yanfei Guo, and Rajeev Thakur. 2024. POSTER: Optimizing Collective Communications with Error-bounded Lossy Compression for GPU Clusters. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '24)*. 454–456. doi:10.1145/3627535.3638467

[21] Jiajun Huang, Sheng Di, Xiaodong Yu, Yujia Zhai, Jinyang Liu, Yafan Huang, Ken Raffenetti, Hui Zhou, Kai Zhao, Xiaoyi Lu, Zizhong Chen, Franck Cappello, Yanfei Guo, and Rajeev Thakur. 2024. gZCCL: Compression-Accelerated Collective Communication Framework for GPU Clusters. In *Proceedings of the 38th ACM International Conference on Supercomputing* (Kyoto, Japan) *(ICS '24)*. 437–448. doi:10.1145/3650200.3656636

[22] Jiajun Huang, Sheng Di, Xiaodong Yu, Yujia Zhai, Jinyang Liu, Zizhe Jian, Xin Liang, Kai Zhao, Xiaoyi Lu, Zizhong Chen, Franck Cappello, Yanfei Guo, and Rajeev Thakur. 2024. hZCCL: Accelerating Collective Communication with Co-Designed Homomorphic Compression. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis* (Atlanta, GA, USA) *(SC '24)*. Article 104, 15 pages.

[23] Jiajun Huang, Sheng Di, Xiaodong Yu, Yujia Zhai, Zhaorui Zhang, Jinyang Liu, Xiaoyi Lu, Ken Raffenetti, Hui Zhou, Kai Zhao, Zizhong Chen, Franck Cappello, Yanfei Guo, and Rajeev Thakur. 2024. An Optimized Error-controlled MPI Collective Framework Integrated with Lossy Compression. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 752–764. doi:10.1109/IPDPS57955.2024.00072

[24] Jiajun Huang, Jinyang Liu, Sheng Di, Yujia Zhai, Zizhe Jian, Shixun Wu, Kai Zhao, Zizhong Chen, Yanfei Guo, and Franck Cappello. 2023. Exploring Wavelet Transform Usages for Error-bounded Scientific Data Compression. In *2023 IEEE International Conference on Big Data (BigData)*. 4233–4239. doi:10.1109/BigData59044.2023.10386386

[25] Pu Jiao, Sheng Di, Hanqi Guo, Kai Zhao, Jiannan Tian, Dingwen Tao, Xin Liang, and Franck Cappello. 2023. Toward Quantity-of-Interest Preserving Lossy Compression for Scientific Data. In *nternational Conference on Very Large Data Bases (VLDB)*.

[26] Samuel Li, Stanislaw Jaroszynski, Scott Pearse, Leigh Orf, and John Clyne. 2019. VAPOR: A Visualization Package Tailored to Analyze Simulation Data in Earth System Science. *Atmosphere* 10 (08 2019), 488.

[27] Shaomeng Li, Peter Lindstrom, and John Clyne. 2023. Lossy Scientific Data Compression With SPERR. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 1007–1017. doi:10.1109/IPDPS54959.2023.00104

[28] Xin Liang, Qian Gong, Jieyang Chen, Ben Whitney, Lipeng Wan, Qing Liu, David Pugmire, Rick Archibald, Norbert Podhorszki, and Scott Klasky. 2021. Error-controlled, progressive, and adaptable retrieval of scientific data with multilevel decomposition. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 88, 13 pages. doi:10.1145/3458817.3476179

[29] Xin Liang, Ben Whitney, Jieyang Chen, Lipeng Wan, Qing Liu, Dingwen Tao, James Kress, David Pugmire, Matthew Wolf, Norbert Podhorszki, and Scott Klasky. 2022. MGARD+: Optimizing Multilevel Methods for Error-Bounded Scientific Data Reduction. *IEEE Trans. Comput.* 71, 7 (2022), 1522–1536. doi:10.1109/TC.2021.3092201

[30] Peter Lindstrom. 2014. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2674–2683.

[31] J. Liu, S. Di, S. Jin, K. Zhao, X. Liang, Z. Chen, and F. Cappello. 2023. Scientific Error-bounded Lossy Compression with Super-resolution Neural Networks. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE Computer Society, Los Alamitos, CA, USA, 229–236. doi:10.1109/BigData59044.2023.10386682

[32] Jinyang Liu, Sheng Di, Kai Zhao, Sian Jin, Dingwen Tao, Xin Liang, Zizhong Chen, and Franck Cappello. 2021. Exploring Autoencoder-based Error-bounded Compression for Scientific Data. In *IEEE International Conference on Cluster Computing*. 294–306. doi:10.1109/Cluster48925.2021.00034 arXiv:2105.11730 [cs.LG] Online.

[33] Jinyang Liu, Sheng Di, Kai Zhao, Xin Liang, Zizhong Chen, and Franck Cappello. 2022. Dynamic Quality Metric Oriented Error Bounded Lossy Compression for Scientific Datasets. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15. doi:10.1109/SC41404.2022.00067

[34] Jinyang Liu, Sheng Di, Kai Zhao, Xin Liang, Sian Jin, Zizhe Jian, Jiajun Huang, Shixun Wu, Zizhong Chen, and Franck Cappello. 2024. High-performance Effective Scientific Error-bounded Lossy Compression with Auto-tuned Multi-component Interpolation. *Proc. ACM Manag. Data* 2, 1, Article 4 (mar 2024), 27 pages. doi:10.1145/3639259

[35] Victor A. P. Magri and Peter Lindstrom. 2024. A General Framework for Progressive Data Compression and Retrieval . *IEEE Transactions on Visualization Computer Graphics* 30, 01 (Jan. 2024), 1358–1368. doi:10.1109/TVCG.2023.3327186

[36] David A. Patterson and John L. Hennessy. 2013. *Computer Organization and Design: The Hardware/Software Interface* (5th ed.). Morgan Kaufmann.

[37] Purdue Anvil supercomputer. 2024. https://www.rcac.purdue.edu/anvil. Online.

[38] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. 2017. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 1129–1139.

[39] Xuan Wu, Qian Gong, Jieyang Chen, Qing Liu, Norbert Podhorszki, Xin Liang, and Scott Klasky. 2024. Error-controlled Progressive Retrieval of Scientific Data under Derivable Quantities of Interest . In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, Los Alamitos, CA, USA, 1–16. doi:10.1109/SC41406.2024.00092

[40] Longtao Zhang, Ruoyu Li, Congrong Ren, Sheng Di, Jinyang Liu, Jiajun Huang, Robert Underwood, Pascal Grosset, Dingwen Tao, Xin Liang, Hanqi Guo, Franck Cappello, and Kai Zhao. 2025. LCP: Enhancing Scientific Data Management with Lossy Compression for Particles. *Proc. ACM Manag. Data* 3, 1, Article 50 (Feb. 2025), 27 pages. doi:10.1145/3709700

[41] Kai Zhao, Sheng Di, Maxim Dmitriev, Thierry-Laurent D. Tonellot, Zizhong Chen, and Franck Cappello. 2021. Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation. In *IEEE 37th International Conference on Data Engineering*. 1643–1654. doi:10.1109/ICDE51399.2021.00145

[42] Kai Zhao, Sheng Di, Xin Liang, Sihuan Li, Dingwen Tao, Julie Bessac, Zizhong Chen, and Franck Cappello. 2020. SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors. In *2020 IEEE International Conference on Big Data (Big Data)*. 2716–2724.