Improving Prediction-Based Lossy Compression Dramatically via Ratio-Quality Modeling

Sian Jin[⋆], Sheng Di[†], Jiannan Tian[⋆], Suren Byna[‡], Dingwen Tao[⋆], Franck Cappello[†]

[⋆]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA
[†]Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
[‡]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract—**Error-bounded lossy compression is one of the most effective techniques for reducing scientific data sizes. However, the traditional trial-and-error approach used to configure lossy compressors for finding the optimal trade-off between reconstructed data quality and compression ratio is prohibitively expensive. To resolve this issue, we develop a general-purpose analytical ratio-quality model based on the prediction-based lossy compression framework, which can effectively foresee the reduced data quality and compression ratio, as well as the impact of lossy compressed data on post-hoc analysis quality. Our analytical model significantly improves the prediction-based lossy compression in three use-cases: (1) optimization of predictor by selecting the best-fit predictor; (2) memory compression with a target ratio; and (3) in-situ compression optimization by fine-grained tuning error-bounds for various data partitions. We evaluate our analytical model on 10 scientific datasets, demonstrating its high accuracy (93.47% accuracy on average) and low computational cost (up to 18.7× lower than the trial-and-error approach) for estimating the compression ratio and the impact of lossy compression on post-hoc analysis quality. We also verify the high efficiency of our ratio-quality model using different applications across the three use-cases. In addition, our experiment demonstrates that our modeling-based approach reduces the time to store the 3D RTM data with HDF5 by up to 3.4× with 128 CPU cores over the traditional solution.**

I. INTRODUCTION

Large-scale scientific simulations on parallel computers play an important role in today's science and engineering domains. Such simulations can generate extremely large amounts of data. For example, one Nyx [1] cosmological simulation with a resolution of $4096 \times 4096 \times 4096$ cells can generate up to 2.8 TB of data for a single snapshot; a total of 2.8 PB of disk storage is needed, assuming the simulation runs 5 times with 200 snapshots dumped per simulation. Despite the ever-increasing computation power can be utilized to run the simulations nowadays, managing such large amounts of data remains challenging. It is impractical to save all the generated raw data to disk due to: (1) limited storage capacity even for large-scale parallel computers, and (2) the I/O bandwidth required to save this data to disk can create bottlenecks in the transmission [2]–[4].

Compression of scientific data has been identified as a major data reduction technique to address this issue. More specifically, the new generation of error-bounded lossy compression techniques, such as SZ [5]–[7] and ZFP [8], have been widely used in the scientific community [4]–[13]. Compared to lossless compression that typically achieves only 2× compression ratio [14] on scientific data, error-bounded lossy compressors provide much higher compression ratios with controllable loss of accuracy.

Scientific applications on large-scale computer systems such as supercomputers typically use parallel I/O libraries, such as HDF5 [15], for managing the data. In specific, Hierarchical Data Format 5 (HDF5) is considered to provide high parallel I/O performance, portability of data, and rich API for managing data on these systems. HDF5 has been used heavily at supercomputing facilities for storing, reading, and querying scientific datasets [16], [17]. This is because HDF5 has specific designs and performance optimizations for popular parallel file systems such as Lustre [18], [19]. Moreover, HDF5 also provides users dynamically loaded filters [20] such as lossless and lossy compression [21], which can automatically store and query data in compressed formats. HDF5 with lossy compression filters can not only significantly reduce the data size, but also improve performance of managing scientific data.

However, for HDF5 to take advantage of lossy compressors, it is essential for users to identify the optimal trade-off between the compression ratio and compressed data quality, which is fairly complex. Since there is no analytical model available to foresee/estimate the compression quality accurately, the configuration setting (such as error bound types and values) of error-bounded lossy compressors for scientific applications relies on empirical validations/studies based on domain scientists' trial-and-error experiments [11]–[13]. The trial-and-error method [1] suffers from two significant drawbacks, which leads to significant issues in practice. First, this method has an extremely high computational cost, in that users need to run applications with diverse combinations of input data, and each run may cost tremendous computational resources. For example, to find an optimized error bound for a given Nyx simulation with a qualified power spectrum analysis, about 10 trials of compression-decompression-analysis are needed before compressing the data with the optimized configuration [12], [13]. Second, the identified configuration setting is still dependent on specific conditions and input data, so

---

[1]The trial-and-error experiment is to compress and decompress the data with different feasible error bounds (or combinations of error bounds) and measure the compression ratio and data quality to choose the best error bound(s).

it cannot be applied across datasets generically because of the lack of a theoretical compression quality model.

In this paper, we theoretically develop a novel, analytical model in terms of the prediction-based lossy compression framework, that can efficiently and accurately estimate the compression quality such as ratio and data distortion for any given dataset, consolidating the confidence of lossy compression quality for users. Specifically, we perform an in-depth analysis for the critical components across multiple stages of the prediction-based error-bounded lossy compression framework, including distribution of prediction errors, Huffman encoding efficiency, effectiveness of quantization, post-hoc analysis quality, and so on. Our model features three critical characteristics: (1) it is a general model suiting most scientific datasets and applications, (2) it has a fairly high accuracy in estimating both ratio and post-hoc analysis quality, and (3) it has very low computational overhead.

To the best of our knowledge, this work is the first attempt to develop an analytical model theoretically for lossy compression quality. which fundamentally differs from all existing compression modeling approaches. Lu et al. [9], for example, focus only on the quantization stage and compression ratio estimation by extrapolation, while our model considers all compression stages for both ratio and quality, which can significantly improve the modeling accuracy. Wang et al. [22] developed a simple model to estimate compression ratios based on an empirical study of the correlation between compression ratio and multiple statistical metrics of the data (such as prediction hit ratio). It can only provide a rough estimation of compression ratio ($\sim$10%$\sim$60% error rate), which cannot satisfy the real-world application demand [13]. Jin et al. [23] leveraged a simplified error distribution of SZ compressor to estimate the post-hoc analysis quality for cosmology application. The compression quality estimation, however, relies on an empirical study which is specific to Nyx cosmology datasets. Compared to all existing solutions in estimating the lossy compression quality, our proposed model can offer in-situ optimization of lossy compression quality with significantly higher compression ratios and low computational overhead.

The contributions of this work are summarized as follows:

- We decouple prediction-based lossy compressors to build a modularized model for ratio and quality estimation.
- We theoretically analyze how to estimate the encoder efficiency and provide essential parameters for compression ratio estimation. We build a fine-tuning mechanism to improve the lossy compression quality estimation accuracy for different predictors.
- We propose a theoretical analysis to estimate the qualification of lossy decompressed data on post-hoc analysis based on the estimated error distribution considering both uniform and nonuniform distributions.
- We evaluate our model using 10 real-world scientific datasets involving 17 fields. Experiments verify that our approach can minimize the overhead of compression optimization and provide accurate ratio and quality estimation.
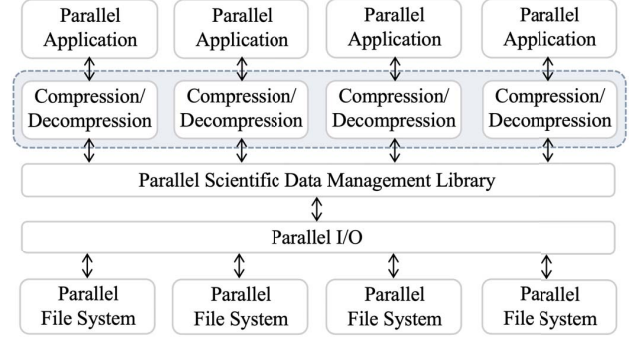

Fig. 1: Scientific data management with compression.

- We evaluate our model on three use-cases and show that it can significantly improve the performance of prediction-based lossy compressors in terms of optimization time overhead and overall compression time for predictor optimization, memory compression optimization, and fine-grained ratio-quality optimization.

II. RESEARCH BACKGROUND

In this section, we present the background information on lossy compression and discuss the research challenges.

A. Data Management in Scientific Applications

In recent years, data management for scientific applications has become a fairly non-trivial challenge. Researchers must develop efficient data management approaches and software to handle extreme data sizes and unusual data movement characteristics. For example, the Advanced Photon Source (APS) [24] requires large datasets movement between the synchrotron and compute/storage facilities, and specialized data management libraries are proposed to handle such data movement [25]. In general, HDF5 [15], netCDF [26], and Adaptable IO System (ADIOS) [27] are the most widely used data management software libraries for scientific applications running on high-performance computers. However, these scientific data management techniques still suffer from extremely large datasets and subsequent I/O bottlenecks, therefore, compression techniques are often adopted by them. Figure 1 shows the abstraction of different layers in these data management systems with compression. Note that compression functions as an individual layer in the data management system. Specifically, compression is performed between generating and storing the data, while decompression is performed between querying and deploying the data. As a result, compression with high ratio can significantly improve the overall performance of large-scale scientific data management.

In this paper, considering HDF5 and its plugins [28], [29] are well received by the scientific community as a system supporting data management, we mainly focus our performance evaluation on HDF5 without loss of generality. Specifically, previous works propose several tools built on HDF5 that support querying. For example, Apache Drill [28] and Pandas [29] allow querying HDF5 metadata and data. Furthermore, its data reorganization has also been studied using FastBit indexes and

transparently redirecting data accesses to the reorganized data or indexes [30]. In addition, H5Z-SZ [21] provides a filter for integrating SZ into HDF5. Therefore, a deep understanding on lossy compressors can potentially significantly improve the overall performance of data management with HDF5.

B. Error-Bounded Lossy Compression

Lossy compression can compress data with extremely high compression ratio by losing non-critical information in the reconstructed data. Two types of most important metrics to evaluate the performance of lossy compression are: (1) compression ratio, i.e., the ratio between original data size and compressed data size, or bit-rate, i.e., the number of bits on average for each data point on average (e.g., 32/64 for single/double-precision floating-point data before compression); and (2) data distortion metrics such as peak signal-to-noise ratio (PSNR) to measure the reconstructed data quality compared to the original data. In recent years, a new generation of high accuracy lossy compressors for scientific data have been proposed and developed for scientific floating-point data, such as SZ [5]–[7] and ZFP [8]. These lossy compressors provide parameters that allow users to finely control the loss of information due to lossy compression. Generally, lossy compressors provide multiple compression modes, such as error-bounding mode. Error-bounding mode requires users to set an error type, such as the point-wise absolute error bound and point-wise relative error bound, and an error bound level (i.e., 10^{-3}). The compressor ensures that the differences between the original data and the reconstructed data do not exceed the user-set error bound level.

In this paper, we mainly focus on ratio-quality modeling for prediction-based lossy compression. The workflow of prediction-based lossy compression [31]–[34] consists of three main stages: prediction, quantization, and encoding. First, each data point's value is predicted using a generic or specific prediction method. For example, the Lorenzo predictor can generally provide an accurate prediction for many simulation datasets [35]–[37], while the spline interpolation based predictor can make a better prediction on seismic data (as proved in a recent study [36]). Then, each prediction error (the error between the predicted value and the original value) is quantized to an integer (called "quantization code") based on a user-set error mode and error bound. For instance, if the user needs to control the global upper bound of pointwise compression errors (the error between the original value and the reconstructed value), linear-scaling quantization scheme will be used with the quantization interval size [32] equal to 2 times of the user-set error bound. Lastly, one or multiple encoding techniques such as Huffman coding [38] (variable-length encoder) and LZ77 [39] (dictionary encoder) will be applied to the quantization codes to reduce the data size.

Most previous works regarding lossy compression mainly aimed to improve the lossy compression ratio based on a specific algorithm. Specifically, some focus on how to improve the prediction efficiency and/or encoding efficiency [31]–[33], [35], [36]; some focus on application-specific optimizations

based on empirical studies or trial-and-error methods [12], [13]. In contrast, this paper is the first efficient attempt to provide an accurate estimation of compression ratio/quality and theoretically guide the use of lossy compression in database/scientific applications.

Without an analytical model, existing lossy compression users have to use trial-and-error approach to obtain expected compression ratio and quality empirically. In other words, one needs to experimentally run compression and decompression on the given dataset with a series of different compression configurations (e.g., error bound) to measure the compression ratio/quality and generate the rate-distortion. Due to the high time cost, it is impossible to use this approach for in-situ optimization in database/scientific applications. For example, a comprehensive framework, Foresight [13], has been developed to automate this process, but it is still limited to offline scenarios due to its high overhead [12]. Moreover, the offline optimization performs poorly in terms of compression-ratio/quality control over different data partitions or timesteps [23]. In this paper, we build a systematic model for prediction-based lossy compression, supporting accurate and efficient ratio-quality estimation.

C. Research Goals and Challenges

Our work is the first work that provides a generic modeling approach for lossy compressors to accurately estimate its compression quality and hence avoid the trial-and-error overhead, which is an essential research issue for today's lossy compression work. To achieve this, four main challenges need to be addressed: (1) How to decompose prediction-based lossy compression into multiple stages and model the compression ratio for each stage? We target to accomplish theoretical analysis for every compression stage (i.e., prediction, quantization and encoding) independently and propose the overall ratio-quality model based on them. (2) How to reduce the time cost of extracting data information needed by the model? We target to design an efficient sampling strategy that can guarantee our prediction accuracy. (3) How to model the quality degradation in terms of diverse post-analysis metrics? We target to provide a guideline to incorporate new application-specific analysis metrics into our model by performing theoretical or empirical analysis. (4) How does our model benefit real-world applications? We target to design various optimization strategies for multiple use-cases to balance the compression ratio and the post-hoc analysis quality on reconstructed data.

III. RATIO-QUALITY MODELING

In this section, we describe the overall design of our proposed ratio-quality model for prediction-based lossy compressors and present the detailed analysis of each component.

A. Overall Design

Figure 2 illustrates the workflow of our ratio-quality model, which is fully modularized with a high extensibility. Our ratio-quality model is built based on two main estimates: compression ratio and post-hoc analysis quality. We build a ratio-
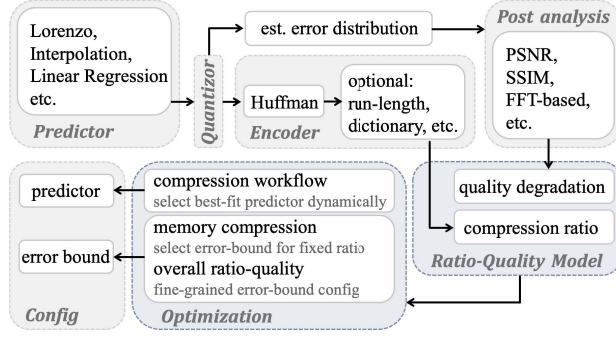
Fig. 2: An overview of ratio-quality modeling workflow for prediction-based lossy compression and scientific data analysis.

Fig. 3: Compression ratio from Huffman encoder and optional lossless encoder from Zstandard and Gzip on quantization code.

quality model based on error bound and provide optimization for different predictors and error bounds.

To model compression ratio, we first examine the existing error-bound modes offered by lossy compressors. For example, a logarithm transformation is needed before compression for pointwise relative error bound mode [35]. Then, the main compression-ratio estimation consists of three modules for predictor, quantization, and encoder, respectively, as shown in Figure 2. We first model the predictor to provide an estimated histogram of prediction errors. For example, a specifically designed sampling strategy is used for interpolation prediction. After that, we model the quantization stage to estimate both error distribution and quantization-code histogram. Finally, we estimate the compression ratio based on our theoretical analysis of the efficiencies of Huffman encoding and optional lossless compression.

To model the post-hoc analysis quality, we first determine the error distribution from the compressor quantization step. Then, we analyze the impact on post-hoc analysis quality by a theoretical derivation for error propagation based on hypothetical error injection to the datasets. For example, we predict the value of PSNR based on the variance of the estimated error distribution.

In addition, our ratio-quality model can be leveraged for multiple use-cases, which will be described in detail in Section IV. Note that we provide a thorough modeling of multiple lossy compressor modules, while it is not necessary to apply the entire model for certain use-cases. The users, for example, can conduct memory compression based on a target ratio without post-hoc analysis quality estimation. This can minimize the computational overhead on demand. We will detail our modularized model in the following sections. We follow three consecutive steps: (1) model compression ratio of popular encoders (i.e., Huffman encoder and run-length encoder); (2) refine compression ratio modeling for various predictors and quantizers; and (3) model quality degradation for both generic and specific post-hoc analysis.

B. Modeling Encoder Efficiency

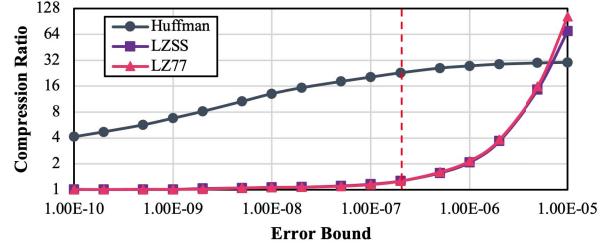When encoding the quantization code, a Huffman encoder is applied first, followed by other optional lossless encoding

techniques (e.g., run-length encoding, dictionary based encoding). Prior studies [31]–[34] have pointed out that using the Huffman encoder for the quantization code plays a major role on the best overall encoding efficiency. On the one hand, based on our experiments, we find the encoding efficiency provided by Huffman encoding is highly separated from the encoding efficiency provided by the optional lossless encoders, shown in Figure 3. This means that the encoding efficiency provided by the optional lossless encoders only complements Huffman encoding after it reaches a certain limit ($\sim$1 bit per symbol). On the other hand, we find that applying run-length encoding on the output of Huffman encoding can get the compression ratio very close to the one that uses an entire lossless compressor (e.g., Zstandard) after Huffman encoding. This is because the predictor always tries best to predict data points as accurately as possible, such that a large majority of the predicted values would fall within the error bound around the corresponding real values. The corresponding quantization code is marked as 'zero' in this situation. Accordingly, zero would always dominate the Huffman codes, especially when the compression is performed with a relatively high error bound. We will demonstrate the effectiveness of this approximation approach in Section V (i.e., the "Lossless Error" column in Table II, where the lossless-encoding efficiency is predicted based on RLE only). Thus, we estimate the encoder efficiency based on (i) Huffman encoding that encodes the frequency information and (ii) run-length encoding that encodes the spatial information (for high error bounds). Modeling each of these two coding methods includes two key compression modes: fix-error-bound (a.k.a., fix-accuracy) mode and fix-rate mode [40], which will be detailed in the following discussion, respectively.

1) Modeling Huffman Coding: In what follows, we describe how to estimate the bit-rate (i.e., compression ratio) based on a given error bound first, and then derive the required error bound based on a target bit-rate.

Estimate bit-rate: We model the bit-rate B which resulted from Huffman coding (i.e., the average bit length of each data point), in terms of the quantization codes that were generated from the previous steps (prediction + quantizer) as follows:

$$B = \sum_{i=0}^{n} P(s_i)L(s_i) \approx -\sum_{i=0}^{n} P(s_i)\log_2 P(s_i), \quad (1)$$

where n is the number of different Huffman codes, P is the probability (or frequency) of given code s_i, L is the length of given code s_i. We further represent the Huffman code length

based on its probability with the binary base-2 numeral system. In Equation (1), when processing the code with the highest frequency, we need to adjust its bit-rate $-log_2 P(s_i)$ to be the minimum code length (i.e., 1 bit).

Optimize error bound based on bit-rate: The optimized error bound e^* for the target bit-rate B^* based on the existing bit-rate B is shown in the following equation:

$$e^* = 2^{B-B^*} e, \qquad (2)$$

where e is the profiled error bound with a bit-rate of B, by Equation (1). We derive the above equation as follows.

Proof. Consider that a given error bound e can provide a bit-rate of B, when doubling the error bound to $2e$, the quantization-code histogram also shrinks accordingly where the total number of symbols is reduced by $2\times$ and the probability (i.e., frequency) of each symbol would increase by $2\times$. In this case the bit-rate should be:

$$B' = -\sum_{i=0}^{n/2} P'(s_i) \log_2 P'(s_i) \approx -\sum_{i=0}^{n/2} 2P(s_{2i}) \log_2 2P(s_{2i})$$
$$= -\sum_{i=0}^{n/2} 2P(s_{2i}) \log_2 P(s_{2i}) - \sum_{i=0}^{n/2} 2P(s_{2i}) \log_2 2 \approx B-1. \qquad (3)$$

Applying the above equation iteratively, we obtain the situation with any specific target bit-rate B^* in Equation (2). □

When the number of quantization bins is fairly small, the above estimation method is not applicable, because the approximation in Equation (3) no longer holds. We found Equation (3) starts to fall when the percentage of code zero (i.e., p_0) exceeds 50% based on our extensive experiments and datasets. In this case, we profile the histogram of quantization codes at $p_0 = [0.5, 0.8, 0.95]$ and compute their corresponded B from Equation (1). Then, based on these pairs of (p_0, B), we can interpolate a continuous function to provide a relationship between error bound and Huffman encoding efficiency. Note that we profile the histogram at p_0 by keeping enlarging the width of the central bin of the histogram until its portion reaches p_0 where its width is $2e^*$.

2) Modeling Run-Length Encoding (RLE): As aforementioned, lossless encoders contribute to the compression ratio only when Huffman encoder reaches the limit, where zeros dominate the quantization codes. Moreover, the quantization codes are independently random after an effective prediction due to its high decorrelation efficiency, causing an extremely low probability of consecutive non-zero codes. Thus, we hereby model the RLE on zeros only.

Estimate compression ratio: We model the compression ratio of RLE R_{rle} by the following equation:

$$R_{rle} = 1/(C_1(1-p_0)P_0 + (1-P_0)). \qquad (4)$$

Here P_0 is the percentage of footprint the code zero takes with respect to the full Huffman encoded data size, where p_0 is percentage of the number of zeros.

Proof. We first model the efficiency E_0 (defined as the reciprocal of the reduction ratio) for encoding zeros by the average length of consecutive zeros n_0 and the length l_0 of representation for consecutive zeros:

$$E_0 = C_1/n_0 l_0, \qquad (5)$$

where C_1 is the fixed size of data to represent consecutive code. l_0 is 1 as the length for zero in Huffman codebook. The overall compression ratio by RLE is:

$$R_{rle} = 1/(E_0 P_0 + (1 - P_0)), \qquad (6)$$

considering that the data distribution is independent random as aforementioned, we have n_0 equal to:

$$\sum_{n=1}^{\infty} n p_0^{n-1}(1-p_0) = (1-p_0)\frac{d}{dp_0}\sum_{n=1}^{\infty} p_0^n = \frac{1}{1-p_0}, \quad (7)$$

From Equations (5), (6) and (7), we can get Equation (4). □

Optimize error bound based on bit-rate: The optimized error bound e^* is profiled from quantization code by the target percentage of zeros p_0 that is deduced from Equation (4):

$$p_0 = \sqrt{1 - R_{rle}^{-1} - ((C_1-1)/2)^2} + (C_1-1)/2 \qquad (8)$$

Note we let $P_0 \approx p_0$ to derive Equation (8) in the case of using RLE because code zero dominates the Huffman encoded data.

C. Modeling Quantized Prediction Error Histogram

Quantuized prediction error histogram needs to be modeled, since the prediction-based lossy compression relies on an efficient predictor and quantizer to concentrate the input data information for high encoding efficiency. To this end, we calculate the distribution of prediction errors, based on which the quantization code histogram would be constructed for the encoder module. Note that the prediction-error distribution is different from the quantization-code distribution for the encoder module, where a highly accurate estimated distribution of prediction errors is demanded to estimate the distribution of quantization codes. In order to obtain accurate prediction error distribution with low overhead, we must apply suitable sampling strategy based on different predictors' design principles, to be detailed later. For sparse scientific data, this step also determines the sparsity and removes the corresponding zeros in the prediction error distribution for getting high model accuracy. We analyze and design the sampling strategy for all the three predictors used in SZ: Lorenzo, linear interpolation, and linear regression predictors. We set the sample rate always to 1% to balance the accuracy and overhead based on our evaluation in Section V. Since our design is generic and modularized, new predictors and sampling methods can be added in the future work.

1) Lorenzo Predictor: The Lorenzo predictor [41] in SZ uses the previous few layers of data to build 1-2 levels of Lorenzo prediction for current value. In this case we randomly sample the given data and for each sampled point, then apply the Lorenzo predictor and collect the difference between predicted value and actual value.

2) Linear Interpolation Predictor: This predictor [36] uses the surrounding data points to predict the current value. The prediction starts with the vertices of the input data to predict the middle point; then for each partition divided by this middle point, it performs the same procedure until reaching the smallest granularity [36]. We propose to sample the data points randomly to extract the information from the entire dataset by considering different sampling rates in different interpolation levels. Specifically, the sampling data in the current level is 2^{-n} than the previous level, where n is the data dimension.

3) Linear Regression Predictor: The linear regression predictor [33] is performed by separating the data into small blocks (e.g., 6×6× for 3D dataset) and uses a linear regression function to fit the data in each block. As it differs from the previous two predictors, we must sample the dataset by blocks to perform linear regression. Thanks to the small block size used in SZ, performing sampling in the unit of block is able to represent the entire data for most scientific datasets even with a relatively low sampling rate.

4) Quantization with Error Bound: In most of cases, we use the original value to perform the prediction in the sampling step instead of the reconstructed value used in the actual compression, since we observe that the error distribution differs little in between. Then, we quantize the sampled data based on certain error bounds to calculate an estimated quantization code histogram for the latter analysis such as modeling encoder efficiency.

For the situation with fairly high error bounds (which is determined by a threshold of p_0), the quantization code histogram estimated using original data values could suffer from large distortion. For example, assume two points in a 1D array with the original values of [..., 0.0, 1.3] and the reconstructed values of [..., 0.5, 0.5] under the error bound of 1.0 with the Lorenzo predictor, the prediction errors for the two points are [..., 0.0, 1.3] based on the original values, which fall into two different quantization bins with the bin size of 2.0 (i.e., two times of the error bound); however, the prediction errors for the two points are [..., -0.5, 0.8] based on the reconstructed values, which belong to the same bin. Thus we add a correction layer in our estimation. Specifically, when Lorenzo or linear interpolation predictor is used, we let each bin of the histogram transfer some quantization codes to a different bin to correct the estimation, making it close to using the lossy reconstructed value to predict the next point. For linear interpolation, the theoretical maximum possible bin transfer of each quantization code is ±1. For Lorenzo, the number is ±7. However, we simplify the possible bin transfer to ±1 for both predictors because of the extremely low possibility of higher cross bins transfer compared to ±1, based on our observations. Thus, we propose to adjust the estimated histogram by transferring a certain number of codes between neighboring quantization bins, to simulate this uncertainty. We use the percentage of highest code in current quantization code histogram p_0 to model this transfer number, since it is highly related to the centralization of quantization codes and also fast to compute. The estimated number of codes in each bin will be evenly transferred to its neighboring bins, when the percentage of the most frequent code p_0 exceeds a threshold of 80%. We conclude to add the following random bin transfer when providing the histogram to the encoder module for this:

$$N_{tran} = P_{tran} \cdot N = C_2 \cdot (1 - p_0) \cdot N, \quad \text{when } p_0 \geq \theta_2, \quad (9)$$

where N_{trans} is the number of codes transferred from one bin to its neighboring bins, C_2 is an empirical parameter for the predictor based on our experiment, N is the number of values in given bin. Specifically, $C_2 = 0.2$ for Lorenzo predictor and

$C_2 = 0.1$ for linear interpolation predictor.

D. Modeling Post-hoc Analysis Quality

Post-hoc analysis quality is highly related to the analysis metrics used for specific scientific applications. In this work, we introduce two widely used analysis metrics: PSNR and SSIM. We first provide an estimated error distribution of reconstructed data. Then, we provide a theoretical analysis on each of the analysis metrics by propagating the compression error distribution function in the metric computation. For more domain-specific analysis metrics, the same principle can be adopted to build the post-hoc analysis quality model. We also provide a guideline in the following sections.

1) Error Distribution: We first provide the average and variance of the error distribution that are used for the error propagation analysis. Error distribution of the reconstructed data is determined by the user defined error bound mode and the error bound value. In most of the cases, the error distribution of prediction-based lossy compressors forms a uniform distribution. In which case $\mu(E) = 0$, and we have:

$$\sigma(E)^2 = \sum_{i=0}^{N} (E[i]^2 - \mu^2) \approx \int_{-e}^{e} \frac{1}{2e} x^2 dx = \frac{1}{3} e^2 \quad (10)$$

Here e is the error bound. However, under high error bounds, we observe that the error distribution combines uniform distribution and centralized distribution. This is because when the quantization bin size is fairly large, the error distribution will contain both a near-uniform distribution from non-central bins and a centralized distribution from the central bin. More specifically, the weight is the percentage of the central bin in the quantization code histogram p_0, which is also the highest bin. Thus, we can separate the values within the central bin and others to have (with $\mu(E) = 0$):

$$\sigma(E)^2 = \sum_{i=0}^{(1-p_0)N} (E[i]^2 - \mu^2) + \sum_{i=0}^{p_0 N} (E[i]^2 - \mu^2)$$
$$= (1 - p_0) \frac{1}{3} e^2 + p_0 \sigma(B[0]), \quad (11)$$

where $\sigma(B[0])$ is the variance of values inside the central bin $B[0]$, which can be computed by our sampled data from the predictor module.

2) Modeling PSNR: We model the PSNR of reconstructed-original data as follows:

$$PSNR(D', D) = 20 \log_{10}(minmax) - 10 \log_{10}(\sigma(E)^2) \quad (12)$$

where D' and D is the reconstructed data and original data, respectively, and $minmax$ is the value range.

Proof. We start with modeling the mean squared error (MSE) based on static error distribution (i.e., uniform distribution). The MSE between reconstructed data and original data equals to the variance of the compression error distribution:

$$MSE(D', D) = \sum_{i=0}^{N} (D'[i]^2 - D[i]^2)$$
$$= \sum_{i=0}^{N} (E[i]^2 - \mu^2) = \sigma(E) \quad (13)$$

where E is the error distribution, μ is the average of E that is usually zero from our evaluated predictors. Then, we compute the estimated PSNR by:

$$PSNR(D', D) = 10 \log_{10} \left(\frac{minmax^2}{MSE} \right) \quad (14)$$

The above equation can deduce to Equation (12). □

3) Modeling SSIM: We model the Structural SIMilarity index (SSIM) of reconstructed-original data as follows:

$$SSIM(D', D) = \frac{2\sigma_D^2 + C_3}{2\sigma_D^2 + C_3 + \sigma(E)^2} \quad (15)$$

Where C_3 is a constant parameter when computing SSIM.

Proof. We also propagate the error distribution function in the computation of the SSIM.

$$SSIM(D', D) = \frac{(2\mu_{D'}\mu_D + C_4)(2\sigma_{D'D} + C_3)}{(\mu_{D'}^2 + \mu_D^2 + C_4)(\sigma_{D'}^4 + \sigma_D^2 + C_3)} \quad (16)$$

Here C_3, C_4 are both constant values. Considering the error distribution of reconstructed data from the original data, we assume $\mu(E) = 0$ on a large number of values in our error propagation analysis:

$$SSIM(D', D) = \frac{2\sigma_{D'D} + C_3}{\sigma_{D'}^2 + \sigma_D^2 + C_3} \quad (17)$$

For the variance of reconstructed data $\sigma_{D'}$, we have:

$$\sigma_{D'}^2 = \sum_{i=0}^{N}((D[i] + E[i]) - \mu_D)^2 \approx \sigma_D^2 + \sigma(E)^2 \quad (18)$$

Similarly, for the covariance $\sigma_{D'}^2$ between the reconstructed data and the original data, we have:

$$\sigma_{D'D} = \sum_{i=0}^{N}((D[i] - \mu_D)((D[i] + E[i]) - \mu_D))$$
$$= \sigma_D^2 + \sum_{i=0}^{N}(D[i] - \mu_D)E[i] \approx \sigma_D^2 \quad (19)$$

Based on Equations (16), (17) and (18), we can get Equation (15). Note that we simplify one of the terms in Equations (18) and (19) to its expected value 0 because this term would also show on both sides of the fraction in Equation (15) and its variance has little impact on the overall estimation. □

4) Data Specific Post-hoc Analysis: Specifically designed analysis metrics are also used for some scientific dataset, such as the Power Spectrum analysis and Halo Finder analysis used for Nyx dataset to identify the halo distribution. Previous research has performed error propagation analysis for FFT based Power Spectrum and Halo Finder with given error distribution [23]. However, it uses uniform error distribution when modeling the analysis quality that can result differently under high error bound range. By using our newly proposed error distribution model, we can further improve the estimation accuracy of FFT-based analysis that is evaluated in Section V. A general guideline to quantify the degradation of domain-specific post-hoc analysis on reconstructed data is similar to our analysis process for PSNR and SSIM; we can adapt the post-hoc analysis computation to include the estimated compression error distribution function.

IV. USE-CASES OF THE RATIO-QUALITY MODEL

In this section, we introduce three use-cases leveraging our ratio-quality model to significantly improve the performance of prediction-based lossy compressors.

A. Compression Predictor Selection

The first use-case of our proposed ratio-quality model is to adaptively select the best predictor for any dataset and error bound. In general, using larger error bounds usually results in higher compression ratios and lower data quality. However, the efficiency of each predictor differs in a certain range of ratio-quality curves. With our proposed model, we can provide ratio-quality estimations for each predictor and select the best-fit predictor for a given error bound or target ratio. Compared to the existing methods [42], [43] that use the trial-and-error approach and sample prediction error for every error bound to select the optimal configuration, our method can help significantly reduce the optimization overhead with one-time sampling and efficient estimation. Moreover, the existing methods make decisions only based on the compression ratio, whereas ours considers both ratio and post-analysis quality.

B. Memory Compression with Target Ratio

For applications that stores compressed data in memory and require a specific maximum footprint, our model estimates the compression ratio of any given dataset, which can provide an optimization strategy to efficiently utilize available memory. For these memory compression optimizations, we provide two optimization strategies. First, if the application is not strictly limiting the bit-rate, such as saving compressed data on GPUs where the spilled data can be migrated to CPU, we create a target bit-rate for given one or multiple datasets that is 20% lower than the limitation to allow uncertainty between estimation and real compression. We then compress the data with optimization towards the bit-rate target and allow data movements when exceeding the bit-rate limitation. This expensive but rare situation introduces little overhead to the system thanks to the high accuracy of our compression ratio modeling and the slightly lower target bit-rate for optimization. Second, if the application strictly limits the bit-rate, we can also apply a similar strategy to provide a first-round optimization. Then, for rare situations where the actual size of compressed data still exceeds the limitation, we adjust the second-round optimization and re-compress the data to prevent overflows.

C. In-Situ Compression Optimization

When applying error-bounded lossy compression to a scientific dataset, one of the most common requested optimization is to balance between the compression ratio and reconstructed data quality. For dataset that is considered as a combination of multiple partitions, we are able to characterize each partition specifically based on a number of metrics (e.g., post-hoc analysis used and certain local data information extraction) which would then be used to decide which compression configuration to apply. Thus, we can optimize the compression performance individually for each partition with overall compression ratio and overall analysis quality as ojectives. Note the data partitions referred to as the partition of the entire data used for post-hoc analysis, such as data on multiple ranks or have multiple timesteps for post-hoc analysis. Such optimization is infeasible by previous solutions because of the exponentially increasing combinations for trial-and-error with increasing number of partitions.

V. EXPERIMENTAL EVALUATION

In this section, we present the evaluation results of our proposed ratio-quality model for prediction-based lossy compressors. We compare our approach with previous strategies

TABLE I: Details of Tested Datasets

Name	Dim	Size	Description	Format
CESM [44]	2D	1.47GB	Climate simulation	NetCDF [45]
EXAFEL [46]	4D	51MB	Instrument imaging	HDF5 [15]
Hurricane [47]	3D	1.25GB	Weather simulation	Binary
HACC [48]	1D	19GB	Cosmology simulation	GIO [49]
Nyx [17]	3D	2.7GB	Cosmology simulation	HDF5
SCALE [50]	3D	4.9GB	Climate simulation	NetCDF
QMCPACK [51]	3D	1GB	Atoms' structure	HDF5
Miranda [52]	3D	1.87GB	Turbulence simulation	Binary
Brown [53]	1D	256MB	Synthetic Brown data	Binary
RTM [54]	3D	682GB	Reverse time migration	HDF5
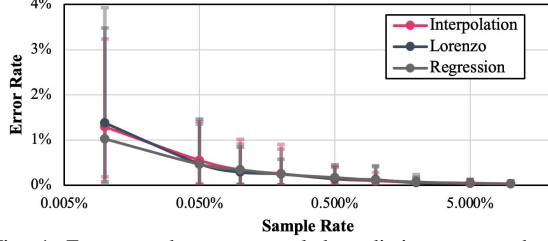
Fig. 4: Error rate between sampled prediction error and original prediction error under different sampling rates with three predictors. The error bar indicates the max and min values.

in terms of both accuracy and performance. Next, we evaluate our model on different use-cases that can significantly improve the compression performance.

A. Evaluation Setup

We perform our evaluation with the SZ3 [55], which is a modularized prediction-based lossy compression framework. We conduct our experiments on the Bebop cluster [56] at Argonne, each node is equipped with two 18-core Intel Xeon E5-2695v4 CPUs and 128GB DDR4 memory. Considering that our workflow can naturally scale up due to no inter-node communication, we evaluate our model accuracy and use-cases study on a single node and the parallel data management performance on 8 nodes with 128 CPU cores. Moreover, we use 10 real-world scientific datasets from the Scientific Data Reduction Benchmarks [53] in the evaluation. Table I shows the detail of our tested datasets.

B. Accuracy of Compression Ratio Model

We first evaluate the accuracy of our modeled compression ratio. Based on our analysis in Section III-A, we first evaluate the accuracy of our sampled prediction error from the predictor module. Then, we conduct our evaluation with two encoder setup situations: (1) we encode the quantization code with Huffman encoder only, and (2) we encode the quantization code with both Huffman encoder and an optional Lossless encoder, in which case we use Zstandard to measure the actual compression ratio in this paper. Note that as aforementioned Section III-B, we model the efficiency of optional lossless encoder based on RLE regardless of which lossless encoder is used, since the quantization codes for lossless encoding are highly decorrelated and zero-dominated.
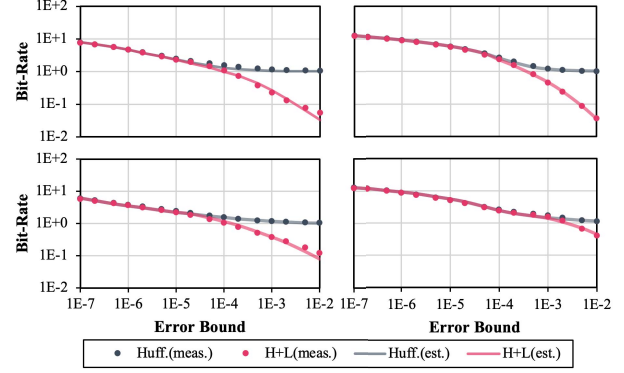

Fig. 5: Compression ratio (bit-rate) estimation accuracy compared to measurement by the encoders.

1) Sampled Prediction Error: To evaluate the effectiveness of our sampled prediction error, we compare the standard deviation of the sampled data and the overall data. Figure 4 shows the decreasing error rate with increasing sampling rate with multiple predictors. We can observe that different predictors behave similarly in terms of error rate with the same sampling rate. The sampled prediction error is used for compression ratio modeling and requires high fidelity to the original to provide accurate estimation. Based on our experiment, we choose the sampling rate of 1% in this paper to balance between the sampled data accuracy and sampling overhead. The detailed sampling error on all datasets with 1% sampling rate can be found in Table II. Overall, our sampling strategy can achieve the sample error (i.e., the standard deviation relative to the value range) of only 0.12% on average.

2) Huffman Encoding Efficiency: Next, we evaluate our modeling of Huffman coding efficiency. The dark black dot and line in Figure 5 shows the measured bit rate after Huffman encoding and the estimated bit rate. The modeling matches the measurements very well above bit-rate of about 2 bits based on Equation (3). After this point, the model switches to the fitted function based on the three anchor points. The lowest estimated bit rate threshold is 1 bit, as expected for Huffman coding in extreme situations. To quantify the accuracy of our modeled Huffman efficiency, we introduce the error computation based on the standard deviation of the ratio between estimated values and measured values:

$$E = 1 - (1 + STD(\frac{R}{R'} - 1))^{-1}, \qquad (20)$$

where E is the accuracy, and R and R' are the measured values and estimated values, respectively. This equation is also used to quantify the prediction error of the following evaluations in this paper. Based on Equation (20), the detailed Huffman encoding efficiency estimation accuracy on all datasets can be found in Table II. Overall, our Huffman encoding model exhibits a high accuracy of up to 98.4% and 94.8% on average. Here we illustrate the error rate, while it can be easily converted to the accuracy (e.g., error rate of 5.16% means the prediction accuracy of 94.84% for Huffman coding).

3) RLE Efficiency: We compare our model to the extra compression ratio provided by Zstandard lossless compressor.

TABLE II: Details of Evaluation Results on Tested Data and Fields

Name	Field	Dim	Sample Err.	Huff Err.	Lossless Err.	Huff+LL. Err.	PSNR Err.	SSIM Err.
RTM	1000	235x449x449	0.03%	5.67%	**9.82%**	8.72%	0.77%	**9.34%**
	2000	235x449x449	0.02%	3.32%	**9.01%**	7.76%	1.56%	**6.56%**
	3000	235x449x449	0.06%	1.88%	**9.15%**	7.57%	2.84%	**4.12%**
CESM	TS	1800x3600	0.06%	6.88%	**11.26%**	8.85%	**3.97%**	2.54%
	TROP_Z	1800x3600	0.20%	7.56%	**10.52%**	9.66%	2.97%	**4.44%**
Hurricane	U	100x500x500	0.10%	**4.62%**	3.46%	5.75%	1.56%	**5.43%**
	TC	100x500x500	0.12%	**5.44%**	2.96%	5.95%	2.42%	**3.80%**
Nyx	Dark Matter	512x512x512	0.14%	**7.53%**	4.36%	7.67%	1.78%	**6.55%**
	Temperature	512x512x512	0.13%	3.92%	**5.13%**	3.99%	1.89%	**4.34%**
	Velosity Z	512x512x512	0.07%	6.85%	**8.65%**	8.08%	2.64%	**3.90%**
HACC	xx	280953867	0.26%	**2.29%**	1.34%	3.22%	1.98%	–
	vx	280953867	0.27%	**3.71%**	1.49%	3.83%	3.67%	–
Brown	Pressure	8388609	0.11%	**5.99%**	5.68%	6.46%	4.42%	–
Miranda	vx	256x384x384	0.13%	**7.90%**	6.95%	8.71%	2.55%	**8.92%**
QMCPACK	einspine	69x69x115	0.13%	6.84%	**8.83%**	6.20%	5.67%	**7.43%**
SCALE	PRES	98x1200x1200	0.16%	1.65%	**2.79%**	2.36%	1.72%	**5.35%**
EXAFEL	raw	10x32x185x388	0.12%	**5.64%**	4.25%	6.23%	3.80%	–
Average	–	–	0.12%	5.16%	**6.21%**	6.53%	2.72%	**5.59%**

* Bold items highlight the larger prediction error between the two encoders and between the two post analyses

From Table II, our model and assumption can accurately provide the compression ratio from the extra lossless compression. The accuracy is up to 98.5% and is 93.8% on average, which is worse than the prediction with only Huffman coding, due to the approximation from lossless compression to RLE.

4) Overall Compression Ratio: The overall compression ratio is the combination of Huffman encoding efficiency and RLE efficiency. The red dot and line in Figure 5 shows the measured bit rate and the estimated bit rate of overall encoder efficiency, respectively. We can observe that our modeling achieves a high accuracy compared to the measurements. Detailed accuracy of overall compression ratio estimation on all datasets can be found in Table II. The accuracy is up to 97.6% and is 93.5% on average. The result shows that our compression-ratio estimation with only Huffman coding is almost always more accurate than that with both Huffman-coding and lossless-encoding stages. Moreover, we observe that our model performs slightly differently across datasets. For example, the estimation errors of Huffman coding and lossless encoding on the HACC dataset are lower than on the other datasets. This is because HACC is 1D data and its quantization codes are more randomly distributed, which lowers the possibility of false quantization code prediction caused by using the original values, thus the compression ratio of HACC is easier to predict. Compared to the previous estimation approach [22] with the accuracy of about 40%∼90%, our theoretical approach provides much higher estimation accuracy consistently across different datasets.

C. Accuracy of Post-Hoc Analysis Quality Model

In this subsection, we evaluate the accuracy of our modeling of post-hoc analysis, including on PSNR, SSIM and data specific post-hoc analysis such as FFT.

1) PSNR: Figure 6 shows the measured PSNR compared to the estimated PSNR based on the error distribution. The dashed red line is the PSNR estimation based on the error
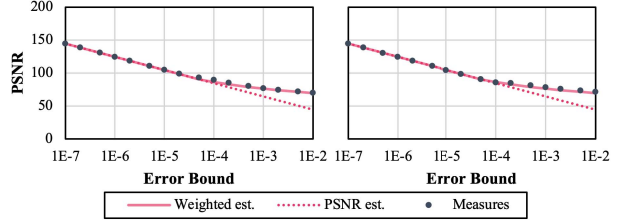

Fig. 6: PSNR estimation accuracy compared to measurement. Evaluated on Nyx dark matter density field with both Linear Interpolation predictor (left) and Lorenzo predictor (right).
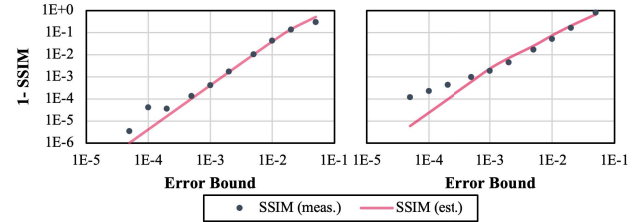

Fig. 7: SSIM estimation accuracy compared to measurement. Evaluated on CESM dataset (left) and Aramco RTM dataset (right).

distribution defined by Equation (10), which only considers the uniform distribution. The solid red line is the PSNR estimation that utilizes both Equations (10) and (11). We can observe that under high error bound situations, the refined distribution of Equation (11) can benefit the refinement of post-hoc analysis quality estimation. Similar observation can also be found for SSIM and FFT analysis. Similarly, the detailed PSNR estimation accuracy for all datasets can be found in Table II. Overall, our model achieves up to 99.2% and on average 97.3% of accuracy for modeling the PSNR.

2) SSIM: Figure 7 shows the measured SSIM compared to the estimated SSIM based on the error distribution. Note we use the $(1 - SSIM)$ in log scale on y-axis to show the difference between estimation and measurement under lower error bound. The estimation is slightly off under lower
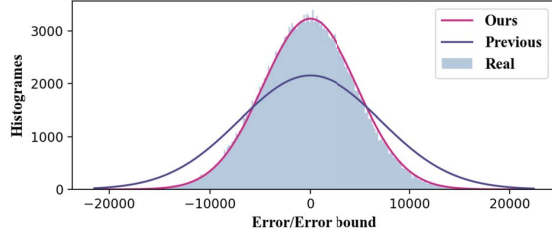
Fig. 8: FFT quality degradation estimation compared to measurement. Evaluated on Nyx temperature field at ABS 500.
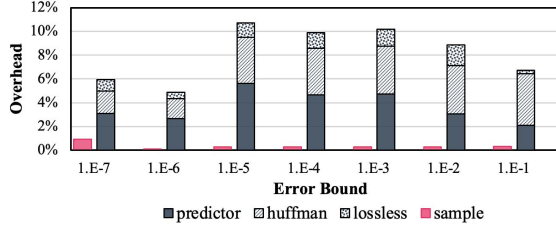

Fig. 9: Performance comparison between proposed modeling solution and previous trial-and-error approach.

Fig. 10: Rate-distortion curve of multiple predictors with different error bound. Evaluated with RTM dataset.

error bounds. This is because the approximation we used for Equation (17) is less accurate since the terms we simplified in Equation (18) & (19) are no longer negligible in the case where Equation (17) is close-to-one. On the other hand, under very high error bounds, the estimation is also degraded since the approximation in Equations (18) and (19) are less accurate when $E[i]$ is larger. Detailed SSIM estimation accuracy for all datasets can be found in Table II. Our evaluation shows that our model on SSIM can provide an accuracy of 94.4% on average. Compared to the SSIM estimation, our model performs better on the PSNR estimation.

3) Data Specific Post-hoc Analysis: Previous study shows cosmology specific post-hoc analysis quality modeling with the SZ lossy compressor [23]. However, it only considered uniform error distribution. With our proposed modeling and guideline for post-hoc analysis quality modeling, we can also accurately estimate the FFT quality degradation under high error bound situations. Figure 8 shows that the proposed estimation that considers error distribution from both Equations (10) and (11) outperforms previous solution that only considered uniform error distributions.

D. Evaluation on Performance Overhead of Our Modeling

We compared the performance of our modeling strategy with the trial-and-error approach. In SZ3, to optimize the predictor for a given error bound, the lossy compressor sampled a proportion of data blocks in given dataset and pre-compresses the structured sampling data with multiple predictor candidates. This module can also be used to just estimate the compression ratio with specified error bounds. When conducting the common use-cases of evaluating the error bound to ratio of a given dataset, our modeling only requires one time data sampling and computes the estimation. However, the previous trial-and-error must re-compress for each combination of error bound and predictor. Figure 9 shows the performance comparison between our workflow
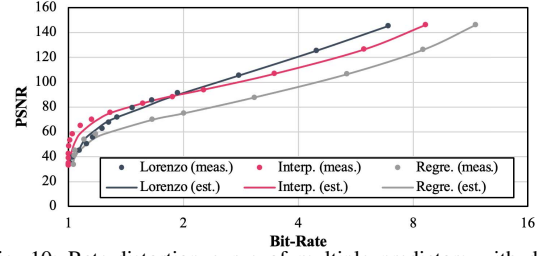
and the previous approach on average across 3 Reverse Time Migration (RTM) datasets. Our solution outperforms the trial-and-error solution by $18.7\times$ on average when considering 7 candidate error bounds to estimate with the Lorenzo and interpolation predictors as candidates. Note that the overhead is relative to the overall compression time. We can observe that by running the compression process, the trial-and-error solution spends a large amount of time on Huffman encoding and lossless compression. Moreover, our solution spends less time even compared to only the predictor part of previous solution, thanks to our newly designed sampling strategy that allows lower sample rates with even higher accuracy. Note that for each predictor our sampling only happens once (at the error bound of 1E-7) for all error bounds, since we can compute the distribution of quantization codes based on our model instead of repeating the prediction with different error bounds. It is worth noting that the trial-and-error approach cannot utilize our new sampling strategy due to the difference of using the original values and the reconstructed values in prediction. In addition, the overhead of our solution is relatively stable across all tested datasets, which demonstrates our consistent high performance in parallel applications (with multiple processes/datasets) using lossy compression.

E. Use-Cases Study

In this section, we investigate the effectiveness of utilizing our ratio-quality model for the three use-cases on the 3D RTM dataset [54]. RTM is an important seismic imaging method for oil and gas exploration [57], [58]. It has forward modeling and backward propagation stages that write and then read the state of the computed solution at specific timesteps, so lossy compression is used to significantly reduce the I/O and memory overheads for 3D RTM.

1) Predictor Selection: For the first use-case, we perform our ratio-quality model on RTM 3D dataset with all three predictors and use PSNR as the analysis metric. Figure 10 shows the estimation and measured rate-distortion correlation. First, we can observe that the estimated rate-distortion curve based on our model is highly accurate compared to measured data points. Secondly, we can clearly see that the linear interpolation predictor provides higher PSNR with the same bit-rates under lower bit-rate ranges. Our model estimation suggests to switch to the linear interpolation from the Lorenzo predictor as the preferred predictor when the estimated bit-rate is lower than 1.89. This estimation is accurate compared to
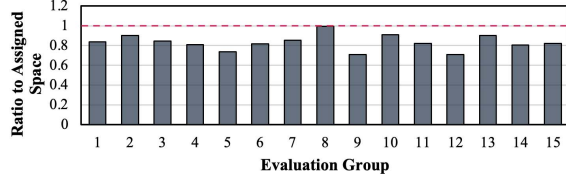
Fig. 11: Ratio of measured space consumption to assigned space. Evaluated with RTM dataset, randomly choose time steps and error bound for 15 groups.
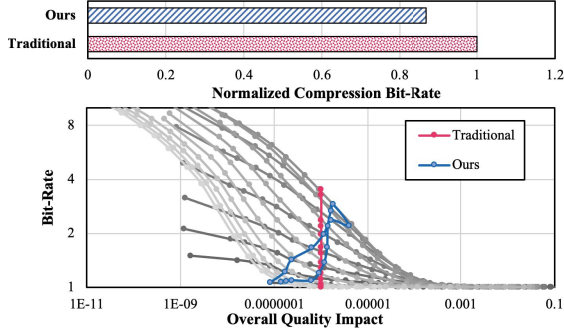

Fig. 12: Error bound optimization for RTM dataset with multiple time steps in consideration for post-hoc analysis.

the measured bit-rates for the predictor switch between [1.47, 1.93]. Our solution also provides 21.8× performance improvement by reducing the overhead from 109.97% to 5.04% compared to the previous sampling solution that requires a trial-and-error for each error bound.

2) Memory Limitation Control: Figure 11 shows the result of compressed file size relative to the assigned memory based on our compression ratio model. We can observe that although many evaluated groups result in larger file sizes than estimated (i.e., 80%), they still stay within the assigned space thanks to the high accuracy of our compression ratio model. The compressed file that exceeds the space limitation may require re-optimization based on a lower target bit-rate. Considering the low possibility of such a situation (e.g.., around 5% for RTM dataset) and the low overhead to recompute the error bound with a given target bit-rate, our solution is still highly practical for fix-rate compression.

3) In-Situ Compression Optimization: The RTM data used for PSNR analysis is a stacked image built form images of multiple timesteps. We consider the images from different timesteps as partitions that form the final dataset. With our ratio-quality model, we optimize the error bound for each image by balancing the ratio of bit-rate and impact on the overall quality degradation. Figure 12 shows the optimized error bounds from our model for every timestep. The quality of each timestep here influences the overall quality of the stacked image. We can observe the trade-offs between timesteps for ratio and quality. Overall, we can provide an extra 13% of compression ratio with same post-hoc analysis quality, or an extra 31% of post-hoc analysis quality with the same compression ratio, compared to using the same error bound for all timesteps. By leveraging our ratio-quality model, we can
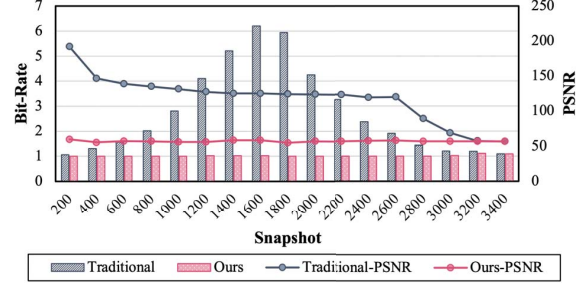

Fig. 13: Comparison between our modeling based method with offline optimization method in terms of both bit-rate and corresponding PSNR across different snapshots when target PSNR is 56 dB.

perform in-situ optimization for parallel applications, which is infeasible with previous solutions.

F. Evaluation on Overall Performance of Data Management

In this section, we use the RTM data to demonstrate the effectiveness of our ratio-quality model when deploying it to the scientific data management systems. More specifically, we use parallel HDF5 [59] built upon MPI-IO [60] as our data management system. Similar to other scientific applications, such as Nyx [17] and HACC [48], the RTM simulation needs to store one snapshot every few iterations for future use and analysis. The overall simulation time spent in I/O can easily reach beyond 50%, thus, lossy compression of data before storing it can significantly improve the I/O performance.

Previously, researchers must conduct offline trial-and-error experiments or use a benchmark toolkit (e.g., Foresight [61]) to determine the bestfit compression configurations for a given set of data. Such static analysis takes a long time, but to make matters worse, it can only choose the worst case configurations to guarantee all the reconstructed data quality, similar to the Liebig's barrel. In the following experiments, we call this static offline analysis method the traditional approach. More specifically, we let the compressor to experiment the error bound from 5 candidates (i.e., ABS 1E-4, 1E-5, 1E-6, 1E-7, 1E-8) for all snapshots and choose one error bound that fits all. During the performance evaluation, we apply this error bound (i.e., ABS 1E-7 based on our experiment) to all snapshots. Different from the traditional approach, our ratio-quality model allows us to in-situ determine the optimized error bound for different snapshots based on the desired reconstructed data quality (i.e., PSNR). Figure 13 shows the ratio-quality comparison on the RTM data across multiple snapshots between the traditional static, offline approach and our adaptive, in-situ approach. In this example, our target is to ensure the PSNR is higher than 56 dB for all snapshots, which guarantees the reconstructed data quality of every snapshot for postprocessing stages. We can observe that the traditional solution chooses only one error bound for all snapshots, causing the PSNR of most snapshots to be much higher than the target. By comparison, our in-situ solution with the ratio-quality model can provide consistent and low bit-rate across all snapshots while satisfying the requirement on reconstruction data quality.
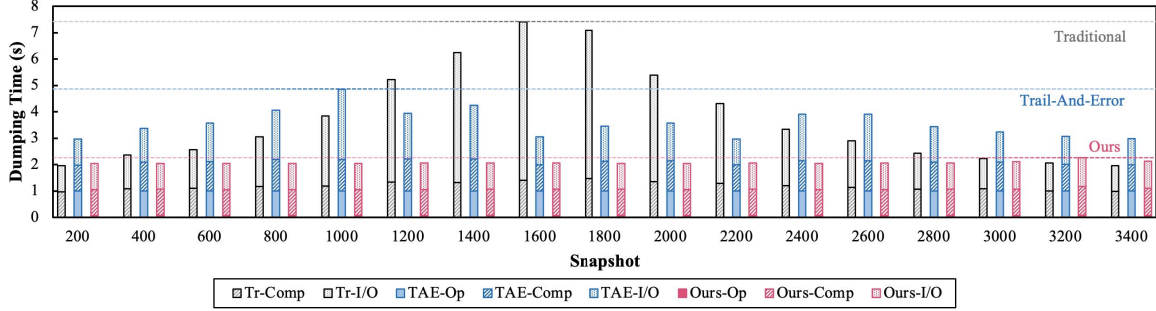
Fig. 14: Overall data dumping performance with parallel HDF5. Comparison between traditional method, trial-and-error and our modeling based method. Dashed lines highlight the maximum dumping time occurred in the simulation. "Tr" refers to the traditional approach, "TAE" refers to the in-situ trial-and-error approach. 'Comp', 'I/O', and 'Op' refer to times of compression, I/O, and optimization, respectively.

Other than the traditional approach, we also implement the trial-and-error method as an in-situ process in our data management system for a fair comparison, referred to as the in-situ trial-and-error (TAE) approach in the following experiment. More specifically, we let the compressor to experiment the error bound from 5 candidates for a given snapshot before actually compressing it. Different from the traditional offline approach, we preform this optimization online and choose the optimal error bound for a given snapshot, which introduces an additional optimization overhead. Figure 14 shows the overall data dumping time of all three methods across different snapshots: the traditional approach, the in-situ TAE approach, and our in-situ optimized approach with the ratio-quality model. This experiment is conducted with 128 processes on 8 nodes, each process holding a portion of each snapshot. It includes three types of time: (1) optimization time, the time spent on compression configuration optimization, including the experimenting time in the in-situ TAE approach; (2) compression time, the time spent for compressing the data with specified configuration; and (3) I/O time, the time spent to actually store the compressed data with parallel HDF5. Note that the baseline data-dumping time without any compression is 29.4s for each snapshot, which is higher than any of the three approaches. Compared with the traditional and the in-situ TAE approaches, our approach can significantly reduce the overall data-dumping time thanks to the accurate error bound control that provides high compression ratio. When specifically compared with the in-situ TAE approach, our approach can significantly reduce the optimization time while providing the higher compression ratio to reduce the I/O time. This is because the in-situ TAE approach must experiment several configurations, that is not only time consuming, but also provides limited error bound granularity. Moreover, note that the dumping time of our solution is highly stable, and the longest dumping time is noticeably lower than that of the other two methods, which is critical for the overall stable throughput of the data management system. Overall, our optimization solution with the proposed ratio-quality model can reduce the data management time by up to 3.4× compared to the traditional static, offline solution and by up to 2.2× compared to the in-situ trial-and-error implementation.

VI. Conclusion and Future Work

In this paper, we develop a general-purpose analytical ratio-quality model for prediction-based lossy compressors that can effectively estimate the compression ratio, as well as the impact of the lossy compressed data on post-hoc analysis quality. Our analytical model significantly improves the prediction-based lossy compression in three use-cases: (1) optimization of predictor and compression mode by selecting the best-fit predictor and mode automatically; (2) memory compression optimization by selecting error bounds for fixed or estimated bit-rates; and (3) overall ratio-quality optimization by fine-grained error-bound tuning of various data partitions. We evaluate our analytical model on 10 scientific datasets, demonstrating its high accuracy (93.47% accuracy on average) and low computational cost (up to 18.7× lower than the previous approach) for estimating the compression ratio and the impact of lossy compression on post-hoc analysis quality. We also verify high effectiveness of our ratio-quality model using different applications across the three use-cases. Finally, we demonstrate that our modeling based approach reduces the data management time for the RTM simulation by up to 3.4× with parallel HDF5 on 128 CPU cores, compared to the traditional static, offline solution. In the future, we plan to extend our model to other lossy compressors such as the transform-based lossy compressor ZFP [8] and more post-hoc analysis metrics. In addition, we also plan to target more aggressive memory control with higher memory usage compared to 80% used in the current approach.

REFERENCES

[1] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel, "Nyx: A massively parallel amr code for computational cosmology," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, 2013.

[2] L. Wan, M. Wolf, F. Wang, J. Y. Choi, G. Ostrouchov, and S. Klasky, "Comprehensive measurement and analysis of the user-perceived i/o performance in a production leadership-class storage system," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1022–1031.

[3] ——, "Analysis and modeling of the end-to-end i/o performance on olcf's titan supercomputer," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2017, pp. 1–9.

[4] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong, "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, 2019.

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

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

[7] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," 2018.

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

[9] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, and Z. Qiao, "Understanding and modeling lossy compression schemes on HPC scientific data," in *2018 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2018, pp. 348–357.

[10] H. Luo, D. Huang, Q. Liu, Z. Qiao, H. Jiang, J. Bi, H. Yuan, M. Zhou, J. Wang, and Z. Qin, "Identifying latent reduced models to precondition lossy compression," in *2019 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2019.

[11] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression rate-distortion from automatic online selection between sz and zfp," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[12] S. Jin, P. Grosset, C. M. Biwer, J. Pulido, J. Tian, D. Tao, and J. Ahrens, "Understanding gpu-based lossy compression for extreme-scale cosmological simulations," *arXiv preprint arXiv:2004.00224*, 2020.

[13] P. Grosset, C. Biwer, J. Pulido, A. Mohan, A. Biswas, J. Patchett, T. Turton, D. Rogers, D. Livescu, and J. Ahrens, "Foresight: analysis that matters for data reduction," in *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 2020, pp. 1171–1185.

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

[15] The HDF Group. (2000-2010) Hierarchical data format version 5. [Online]. Available: http://www.hdfgroup.org/HDF5

[16] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the hdf5 technology suite and its applications," in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, 2011, pp. 36–47.

[17] Nyx, https://github.com/AMReX-Astro/Nyx, 2021.

[18] S. Byna, M. S. Breitenfeld, B. Dong, Q. Koziol, E. Pourmal, D. Robinson, J. Soumagne, H. Tang, V. Vishwanath, and R. Warren, "Exahdf5: delivering efficient parallel i/o on exascale computing systems," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 145–160, 2020.

[19] S. Pokhrel, M. Rodriguez, A. Samimi, G. Heber, and J. J. Simpson, "Parallel i/o for 3-d global fdtd earth–ionosphere waveguide models at resolutions on the order of˜ 1 km and higher using hdf5," *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 7, pp. 3548–3555, 2018.

[20] The HDF Group. (2000-2010) Hierarchical data format version 5, Filter. [Online]. Available: https://support.hdfgroup.org/HDF5/doc/H5.user/Filters.html

[21] H5Z-SZ, https://github.com/disheng222/H5Z-SZ.

[22] J. Wang, T. Liu, Q. Liu, X. He, H. Luo, and W. He, "Compression ratio modeling and estimation across error bounds for lossy compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1621–1635, 2019.

[23] S. Jin, J. Pulido, P. Grosset, J. Tian, D. Tao, and J. Ahrens, "Adaptive configuration of in situ lossy compression for cosmology simulations via fine-grained rate-quality modeling," in *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 45–56.

[24] LCF, "The advanced photon source (aps)," https://www.aps.anl.gov/, (Accessed on 11/18/2021).

[25] ——, "Data exchange | advanced photon source," https://www.aps.anl.gov/Science/Scientific-Software/DataExchange, (Accessed on 11/18/2021).

[26] J. Li, W.-k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netcdf: A high-performance scientific i/o interface," in *SC'03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*. IEEE, 2003, pp. 39–39.

[27] W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck *et al.*, "Adios 2: The adaptable input output system. a framework for high-performance data management," *SoftwareX*, vol. 12, p. 100561, 2020.

[28] A. Drill, "Hdf5 format plugin - apache drill," https://drill.apache.org/docs/hdf5-format-plugin/, (Accessed on 11/17/2021).

[29] Pandas, "pandas.read_hdf — pandas 1.3.4 documentation," https://pandas.pydata.org/docs/reference/api/pandas.read_hdf.html, (Accessed on 11/17/2021).

[30] B. Dong, S. Byna, and K. Wu, "Expediting scientific data analysis with reorganization of data," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2013, pp. 1–8.

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

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

[33] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data*. IEEE, 2018, pp. 438–447.

[34] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[35] 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*. IEEE, 2018, pp. 179–189.

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

[37] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, and X. He, "High-ratio lossy compression: Exploring the autoencoder to compress scientific data," *IEEE Transactions on Big Data*, 2021.

[38] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[39] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[40] P. Lindstrom, M. Salasoo, M. Larsen, and S. Herbein, "zfp documentation," https://buildmedia.readthedocs.org/media/pdf/zfp/latest/zfp.pdf.

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

[42] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression rate-distortion from automatic online selection between

sz and zfp," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 8, pp. 1857–1871, 2019.

[43] X. Liang, S. Di, D. Tao, S. Li, B. Nicolae, Z. Chen, and F. Cappello, "Improving performance of data dumping with lossy compression for scientific simulation," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019, pp. 1–11.

[44] Community Earth System Model (CESM) Atmosphere Model, http://www.cesm.ucar.edu/models/, 2019, online.

[45] R. Rew and G. Davis, "Netcdf: an interface for scientific data access," *IEEE computer graphics and applications*, vol. 10, no. 4, pp. 76–82, 1990.

[46] "LCLS-II Lasers," https://lcls.slac.stanford.edu/lasers/lcls-ii, 2021, online.

[47] Hurricane ISABEL Simulation Data, http://vis.computer.org/vis2004contest/data.html, 2019, online.

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

[49] H. team, "hacc / genericio · gitlab," https://git.cels.anl.gov/hacc/genericio, (Accessed on 11/19/2021).

[50] G.-Y. Lien, T. Miyoshi, S. Nishizawa, R. Yoshida, H. Yashiro, S. A. Adachi, T. Yamaura, and H. Tomita, "The near-real-time scale-letkf system: A case of the september 2015 kanto-tohoku heavy rainfall," *Sola*, vol. 13, pp. 1–6, 2017.

[51] QMCPACK: many-body ab initio Quantum Monte Carlo code, http://vis.computer.org/vis2004contest/data.html, 2019, online.

[52] Miranda Radiation Hydrodynamics Data, https://wci.llnl.gov/simulation/computer-codes/miranda, 2019, online.

[53] SDRBench, https://sdrbench.github.io/.

[54] "Seismic toolbox," https://github.com/brightskiesinc/Reverse_Time_Migration, 2021, online.

[55] SZ3: A Modular Error-bounded Lossy Compression Framework for Scientific Datasets, https://github.com/szcompressor/SZ3.

[56] LCF, "Bebop - laboratory computing resource center," https://www.lcrc.anl.gov/systems/resources/bebop/, (Accessed on 11/17/2021).

[57] H.-W. Zhou, H. Hu, Z. Zou, Y. Wo, and O. Youn, "Reverse time migration: A prospect of seismic imaging methodology," *Earth-science reviews*, vol. 179, pp. 207–227, 2018.

[58] T. Alturkestani, H. Ltaief, and D. Keyes, "Maximizing I/O bandwidth for reverse time migration on heterogeneous large-scale systems," in *European Conference on Parallel Processing*. Springer, 2020, pp. 263–278.

[59] The HDF Group, "Parallel HDF5," https://support.hdfgroup.org/HDF5/PHDF5/, (Accessed on 11/19/2021).

[60] R. Thakur, W. Gropp, and E. Lusk, "On implementing mpi-io portably and with high performance," in *Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems*, ser. IOPADS '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 23–32. [Online]. Available: https://doi.org/10.1145/301816.301826

[61] C. Biwer, P. Grosset, S. Jin, J. Pulido, and H. Rakotoarivelo, "VizAly-Foresight: A Compression Benchmark Suite for Visualization and Analysis of Simulation Data," https://github.com/lanl/VizAly-Foresight.