2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00145

Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation

Kai Zhao*, Sheng Di†, Maxim Dmitriev‡, Thierry-Laurent D. Tonellot‡, Zizhong Chen*, and Franck Cappello†§
* University of California, Riverside, CA, USA
† Argonne National Laboratory, Lemont, IL, USA
‡ EXPEC Advanced Research Center, Saudi Aramco, Saudi Arabia
§ University of Illinois at Urbana-Champaign, Champaign, IL, USA
kzhao016@ucr.edu, sdi1@anl.gov, maxim.dmitriev@aramco.com,
thierrylaurent.tonellot@aramco.com, chen@cs.ucr.edu, cappello@mcs.anl.gov

***Abstract*—Today's scientific simulations are producing vast volumes of data that cannot be stored and transferred efficiently because of limited storage capacity, parallel I/O bandwidth, and network bandwidth. The situation is getting worse over time because of the ever-increasing gap between relatively slow data transfer speed and fast-growing computation power in modern supercomputers. Error-bounded lossy compression is becoming one of the most critical techniques for resolving the big scientific data issue, in that it can significantly reduce the scientific data volume while guaranteeing that the reconstructed data is valid for users because of its compression-error-bounding feature. In this paper, we present a novel error-bounded lossy compressor based on a state-of-the-art prediction-based compression framework. Our solution exhibits substantially better compression quality than all of the existing error-bounded lossy compressors, with comparable compression speed. Specifically, our contribution is threefold. (1) We provide an in-depth analysis of why the best-existing prediction-based lossy compressor can only minimally improve the compression quality. (2) We propose a dynamic spline interpolation approach with a series of optimization strategies that can significantly improve the data prediction accuracy, substantially improving the compression quality in turn. (3) We perform a thorough evaluation using six real-world scientific simulation datasets across different science domains to evaluate our solution vs. all other related works. Experiments show that the compression ratio of our solution is higher than that of the second-best lossy compressor by 20%∼460% with the same error bound in most of the cases.**

I. Introduction

With the ever-increasing scale of today's scientific simulations, vast amounts of scientific data are produced at every simulation run. Climate simulation [1], for example, can generate hundreds of terabytes of data in tens of seconds. A cosmology simulation, such as Hardware/Hybrid Accelerated Cosmology (HACC) [2] can produce dozens of petabytes of data when it performs an N-body simulation with up to several trillion particles. Such a vast amount of scientific data needs to be stored for post hoc analysis, creating a huge challenge to the scientific data management systems [3]–[7]. Many scientists also need to share the large amounts of data across different sites (i.e., endpoints) through a data-sharing web service (such as the Globus toolkit [8]) on the Internet. Thus, the ability to significantly compress extremely large scientific datasets with controlled data distortion is critical to today's science work.

In the scientific research domain, the users often adopt scientific data libraries such as NetCDF [9] and HDF5 [10] to manage the scientific data due to their performance advantage and better support of multidimensional objects over traditional database management systems. Those scientific data libraries have database features including metadata, data indexing, data manipulation, and data visualization tools [11], [12]. In particular, due to the vast amount of data to deal with, these data management libraries also support integrating different data compressors. For example, HDF5 offers a filter mechanism [13] to allow users to call various compressors (including SZ [14], ZFP [15], Zlib [16], etc.) transparently when storing scientific data.

Error-bounded lossy compression techniques [17]–[19] have been developed for several years, and they have been widely recognized as an optimal solution to reduce the storage demand of scientific data management systems. For example, many researchers [20], [21] have verified that the data reconstructed through error-bounded lossy compressors is totally acceptable for users' post hoc analysis. Many successful stories also showed that error-bounded lossy compressors not only can significantly reduce the storage space but also may substantially improve the data-moving performance with user-acceptable data distortions. For example, Liang et al. [22] showed that an error-bounded lossy compressor can improve the overall I/O performance by 60X, with no degradation of visual quality on the reconstructed data. Kukreja et al. [23] showed that using error-bounded lossy compression can get high compression ratios without affecting the convergence or final solution of the full waveform inversion solver clearly.

The SZ compression library has been recognized by independent assessments [17], [21], [24], [25] as the best-in-class error-bounded lossy compressor for scientific datasets, especially because it has gone through many careful optimizations [14], [17], [18], [26]–[29].

In this paper, we significantly improve the error-bounded lossy compression quality for scientific datasets, by designing a dynamic best-prediction-selection strategy and proposing a

Corresponding author: Sheng Di, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 Cass Avenue, Lemont, IL 60439, USA

978-1-7281-9184-3/21/$31.00 ©2021 IEEE DOI
10.1109/ICDE51399.2021.00145

novel, spline interpolation based prediction approach with a series of optimizations. This predictor completely eliminates the serious storage overhead compared with the linear-regression predictor used in SZ. Our contribution is threefold.

- We provide an in-depth analysis of the latest version of SZ and identify significant drawbacks of its prediction method; the analysis also sheds light on our new design. The critical challenge in the current design of SZ comes from its linear-regression prediction method, which has two significant drawbacks. On the one hand, it suffers from limited accuracy in predicting nonlinear varied datasets. Many scientific simulations (such as seismic wave simulation [30] and quantum chemistry research [31]) , however, may produce a vast amount of data with nonlinear features, such that SZ cannot work very effectively on them. On the other hand, the linear-regression-based prediction needs to store several coefficients (e.g., four coefficients per block for 3D compression) in each block of data, introducing significant overhead especially when a relatively high compression ratio is required.
- We propose a novel prediction method that can significantly improve the compression ratio compared with the linear-regression prediction method. On the one hand, cubic spline interpolation is included in our novel approach to represent high order data variation, which obtains much higher prediction accuracy over linear-regression for datasets with nonlinear data variation characteristics. On the other hand, we derive the constant coefficients in our interpolation approach such that the coefficient storage overhead can be completely eliminated. We further propose a dynamic optimization strategy to select the best predictor from between the novel spline interpolation approach and the multilevel Lorenzo predictor to improve the overall compression quality.
- We perform a comprehensive assessment of our solution versus five state-of-the-art error-controlled lossy compressors, using multiple real-world simulation datasets across different scientific domains. Experiments show that our solution improves the compression ratio by 20%$\sim$460% over the second-best compressor with the same error bound and experiences no degradation in the post-analysis accuracy.

The rest of the paper is organized as follows. In Section II we discuss the related work. In Section III we formulate the research problem. In Section IV we offer an in-depth analysis of the pros and cons of SZ In Section V we describe our solution and detailed optimization strategies. In Section VI we present the evaluation results compared with five other state-of-the-art lossy compressors using real-world applications. In Section VII we conclude with a discussion of future work.

II. Related Work

Data compression is becoming a critical technique for database management systems. For time series databases, Gorilla [32] is proposed to improve query performance using lossless compression techniques including XOR and variable-length encoding. AMMMO [33] utilizes machine learning to select the best lossless compression scheme for each data point in time series databases. SciDB [34] is a science-oriented database system that supports several lossless algorithms including run-length encoding and adaptive huffman encoding. Snappy [35] is a high-speed lossless compression framework used by many databases such as InfluxDB [36]. Zlib [16] and Zstandard [37] are another two state-of-the-art lossless compressors.

Although lossless compression techniques are widely used in database management systems, they are not suitable for scientific data. Lossless compressors suffer from very limited compression ratios (generally $\sim$2 or even less) on scientific data [38] since lossless compression techniques rely on repeated byte-stream patterns whereas scientific data is often composed of diverse floating-point numbers. Thus, lossy compression for scientific data has been studied for years.

Unlike traditional lossy compression techniques (such as Jpeg2000 [39]) that were designed mainly for image data, the error-bounded lossy compression can not only get a fairly high compression ratio (several dozens, hundreds, or even higher) but also guarantee that the reconstructed data is valid for scientific post-analysis in terms of the user-defined compression error bound. Error-bounded lossy compression can be categorized as higher-order singular value decomposition (HOSVD)-based models such as TTHRESH [40], transform-based models such as ZFP [15], and prediction-based models including SZ [14], [17].

There are also some machine learning (ML) based lossy compressors such as LFZIP [41]. ML compressors have two drawbacks in terms of scientific data prediction. First, the weights of ML models have non negligible size to be stored and ML models need to be retrained for data in different scientific domains. As a result, the model weights should be stored together with compressed data and this brings significant storage overhead. Second, ML compressors involves the ML inference process which has much higher computational cost than traditional methods including interpolation based predictors that are linear time complexity.

In our work, we choose the prediction-based model because SZ has been recognized as the leading compressor in the scientific data compression community. In fact, how to leverage SZ to improve compression quality has been studied for more than two years. Tao et al. [42] developed a strategy that can combine SZ and ZFP to optimize the compression ratios based on a more significant metric, peak signal-to-noise ratio (PSNR). Liang et al. [27] further analyzed the principles of SZ and ZFP and developed a method integrating ZFP into the SZ compression model, which can further improve the compression quality. Zhao et al. [28] proposed to adopt second-order Lorenzo+regression in the prediction methods and developed an autotuning method to optimize the parameters of SZ. Liang et al. [43] accelerated the performance of MultiGrid Adaptive Reduction of Data (MGARD) [44] and used SZ to compress the nodal points generated by the MGARD framework [44],

which can improve the compression ratios significantly.

All these existing SZ-related solutions have to rely on the linear regression prediction to a certain extent. This is a critical restriction to the compression quality improvement, which will be analyzed deeply in Section IV.

III. Problem Formulation

In this section we describe the research problem formulation. Given a scientific dataset (denoted by D) composed of N floating-point values (either single precision or double precision) and a user-specified absolute error bound (e), the objective is to develop an error-bounded lossy compressor that can always meet the error-bounding constraint at each data point with optimized compression quality and comparable performance (i.e., speed).

Rate distortion is arguably the metric most commonly used by the lossy compression community to assess compression quality. It can be converted to the commonly used statistical data distortion metric known as normalized root mean squared error, and it is a good indicator of visual quality. Rate distortion involves two critical metrics: peak signal-to-noise ratio and bit rate. PSNR can be written as the following: Formula (1).

$$PSNR = 20\log_{10}(vrange(D) - 10\log_{10}(mse(D,D')), \quad (1)$$

where D' is the reconstructed dataset after decompression (i.e., decompressed dataset) and vrange(D) represents the value range of the original dataset D (i.e., the difference between its highest value and lowest value). Obviously, the higher the PSNR value is, the smaller the mean squared error, which means higher precision of the decompressed data.

Bit rate is used to evaluate the compression ratio (the ratio of the original data size to the compressed size). Specifically, bit rate is defined as the average number of bits used per data point in the compressed data. For example, suppose a single-precision original dataset has 100 million data points; its original data size is 100,000,000×4 bytes (i.e., about 400 MB). If the compressed data size is 4,000,000 bytes (i.e., a compression ratio of 100:1), then the bit rate can be calculated as 32/100 = 0.32 (one single-precision number takes 32 bits). Obviously, smaller bit rate means higher compression ratio.

Two other important compression assessment metrics are compression speed (denoted by s_c) and decompression speed (denoted by s_d). They are defined as the amount of data processed per time unit (MB/s).

In our research, we focus on the optimization of compression quality (i.e., rate distortion) with high performance, which can be formulated as follows:

$$\begin{aligned} &Optimize \;\; rate\text{-}distortion \\ &subject\,to \;\; |d_i - d_i'| \le e \\ &\qquad\quad s_c(newsol.) \approx s_c(sz) \\ &\qquad\quad s_d(newsol.) \approx s_d(sz), \end{aligned} \quad (2)$$

where d_i and d_i' are referred to the value of the ith data point in the original dataset D and the decompressed dataset D' by the new compression solution, respectively. The notations s_c(*newsol.*) and s_d(*newsol.*) represent the compression speed and decompression speed of the new solution, respectively, and $s_c(sz)$ and $s_d(sz)$ represent the compression speed and decompression speed of the original SZ compressor, respectively. That is, we are trying to increase the compression ratio with the same level of data distortion and comparable compression/decompression performance compared with SZ as a baseline (because SZ has been confirmed as a fairly fast lossy compressor in many existing studies [22], [43]),

In our evaluation in Section VI, not only do we present the rate distortion results for many different datasets at different bit-rate ranges, but we also assess the impact of our lossy compressor on the results of decompressed-data-based post-analysis on one production-level seismic simulation research.

IV. Deeply Understanding the Pros and Cons of SZ

In this section, we first give a review of the current SZ design and then provide an in-depth analysis of a serious problem in the latest version of the SZ compressor (SZ2.1) [18]. Understanding this problem is fundamental to understanding why our new solution can significantly improve the compression ratio.

A. Review of SZ Lossy Compression Framework

SZ2.1 [18], the latest version of SZ, has been recognized as an excellent error-bounded lossy compressor based on numerous experiments with different scientific applications by different researchers [27], [28], [42].

SZ2.1 involves four stages during the compression: (1) data prediction, (2) linear-scale quantization, (3) Huffman encoding, and (4) lossless compression such as Zlib [16]. We briefly describe the four steps, and we refer readers to read our prior papers [17], [18] for technical details.

- **Step 1: data prediction**. In this step, SZ predicts each data point by its nearby data values. The prediction methods differ with various versions (from 0.1 through 2.1). For example, SZ 0.1∼1.0 [14] adopted a simple one-dimensional adaptive curve-fitting method, which selects the best predictor for each data point from among three candidates: previous-value fitting, linear-curve fitting, and quadratic-curve fitting. SZ1.4 [17] completely replaced the curve-fitting method by a multidimensional first-order Lorenzo predictor, significantly improving compression ratios by over 200% over SZ1.0. SZ2.0∼SZ2.1 further improved the prediction method by proposing a blockwise linear regression predictor that can significantly enhance compression ratios by 150%∼800% over SZ1.4, especially for cases with a high compression-ratio (i.e., when the error bound is relatively large).
- **Step 2: linear-scale quantization**. In this step, SZ computes the difference (denoted Δ) between the predicted value (calculated in Step 1) and the original value for each data point and then quantizes Δ based on the user-predefined error bound (e). The quantization bins are equal-sized and are twice as large as the error bound, such that the maximum compression errors must be controlled within the specified error bound. After this step, all

floating-point values are converted to integer numbers (i.e., quantization numbers), most of which are to be close to zero, especially when the data are fairly smooth in locality or the predefined error bound is relatively large.

- **Step 3: customized Huffman encoding**. SZ has a tailored integer-based Huffman encoding algorithm to encode the quantization numbers generated by Step 2.
- **Step 4: lossless compression**. The last step in SZ is adopting a lossless compressor with a pattern-recognized algorithm such as LZ77 [45] to further improve the compression ratios significantly.

B. Critical Features of SZ Compression Framework

First, SZ is a very flexible compression framework, in which the data prediction is the most critical step. More accurate data prediction will result in more quantization numbers being close to zero which leads to a better compression ratio during the encoding and lossless compression steps. Thus we have explored other more efficient predictors in the past two years (from version 0.1 through the latest released version 2.1, as well as a few recent prototypes [27], [28]). Accordingly, we are still focused only on the data prediction stage in this paper.

Second, SZ has to follow a necessary condition, in order to guarantee that the compression errors are always within the user-predefined error bound. For the same data point, its predicted value during the compression stage has to be exactly the same as the one predicted in the decompression stage. Otherwise, the compression errors would be accumulated easily during the decompression, causing totally uncontrolled compression errors. Thus, in the compression stage, SZ has to predict each data point by its nearby lossy decompressed values instead of the original values, which will in turn degrade the prediction accuracy (as exemplified in our prior work [17]). We proposed the linear-regression predictor in SZ2.1 [18], which can mitigate this issue to a certain extent. Such a predictor, however, has a significant drawback and may substantially inhibit the compressor from obtaining a high compression ratio in many cases. We analyze this drawback in detail in the following text.

C. Review of Linear Regression Predictor in SZ2.1

In what follows, we describe the linear regression predictor used in SZ2.1 and its serious drawback.

In SZ2.1, the whole dataset is split into equal-sized blocks (e.g., $6\times6\times6$ for a 3D dataset) and performs a linear-regression-based prediction when the data inside the block is relatively smooth or the error bound is relatively high. The basic idea is to use linear regression to construct a hyperplane in each block, such that the data inside the block can be approximated by the hyperplane with minimized min squared error (MSE), as illustrated in Fig. 1. The details can be found in our prior work [18].

D. Serious Dilemma of Linear-Regression Predictor in SZ2.1

In order to get a high compression quality (i.e., a very good rate-distortion result), the four coefficients need to be compressed based on a certain error bound, which may introduce a serious dilemma: a higher error bound used on coefficient compression will decrease the overhead of storing the coefficients (to be demonstrated in Fig. 2) but also decrease the regression accuracy of the constructed hyperplane (to be demonstrated in Fig. 3). We confirm this issue by four real-world scientific simulations (QMCPack [46], RTM [30], Hurricane [47], and NYX [48]), which are commonly used by scientists in quantum structure research, seismic imaging for oil and gas exploration, climate research, and cosmology research, respectively. More details about these applications are given in Section VI. We exemplify the results using specific fields (e.g., time step 1500 of RTM data, the W field of Hurriane, and velocity z in NYX) because of the space limits and similar results in other fields.

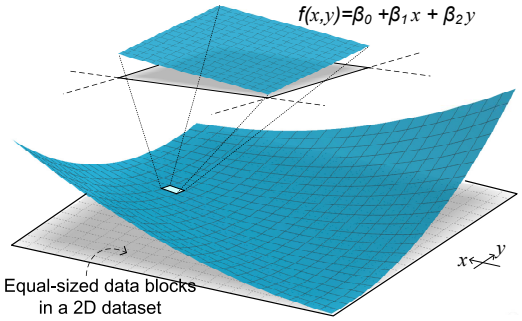
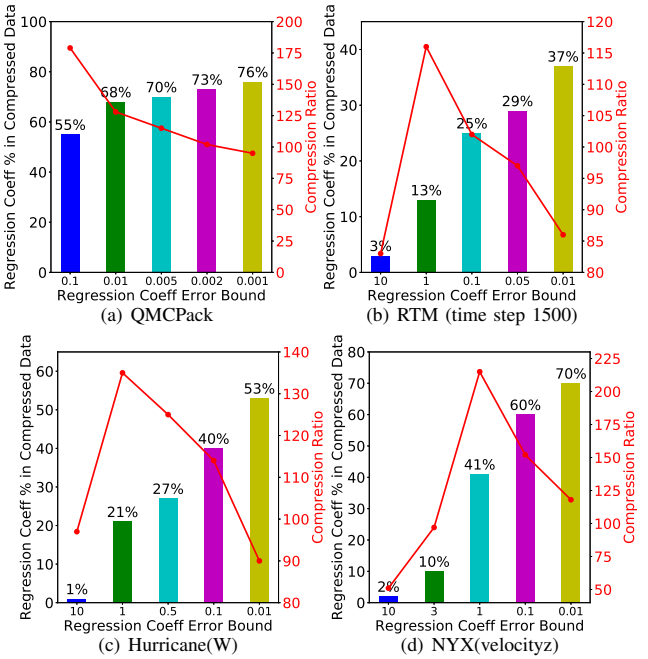


Fig. 1. Illustration of Linear-regression-based prediction (2D dataset)

Fig. 2 shows that the overhead always increases with decreasing error bounds used on the compression of coefficients. Specifically, we observe that when the error bound decreases from 0.1 to 0.01, the coefficient overhead in the compressed data increases from 55% to 68%, from 25% to 37%, from 40% to 53%, and from 60% to 70%, for the four test cases, respectively. The compression ratios (the red curve) thus degrade from 179 to 128, from 102 to 86, from 114 to 90, and from 152 to 118, respectively.

We present a slice segment of the four application datasets in Fig. 3 to illustrate that the error bounds of the coefficient compression would significantly affect the prediction accuracy of the constructed linear-regression hyperplane. For instance, when the coefficients' compression error bound is set to 0.001 for QMCPack and 0.01 for RTM (time step 1500), the constructed hyperplane (the yellow curve) can fit the real data (the red curve) well, but the fitting will be much worse with increasing error bounds. In the case with a relatively large error bound (e.g., 0.1 in QMAPack), the hyperplane will downgrade to a simple horizontal line (see blue lines in the figures), because simply using the neighbor data value is "accurate" enough for the large error-bounded compression of the coefficients. This will definitely result in large prediction errors (the difference between predicted value and raw value) significantly degrading the final compression ratios.

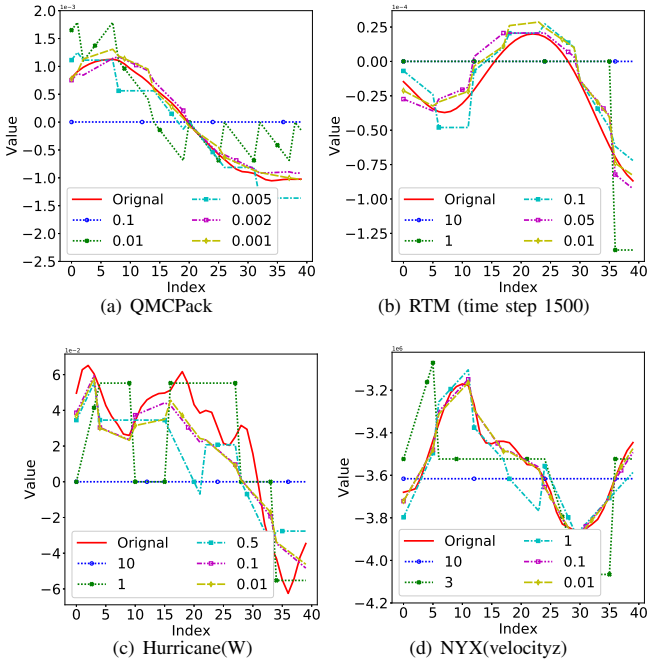
In Fig. 4, we demonstrate that the latest version of SZ (v2.1) may cause significant loss of the data visualization, especially



(a) QMCPack (b) RTM (time step 1500)

(c) Hurricane(W) (d) NYX(velocityz)

Fig. 2. Overhead of Linear Regression Coefficients



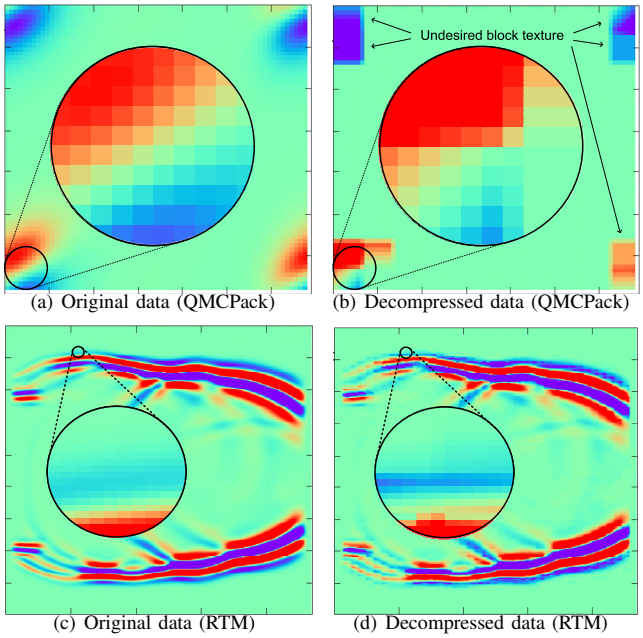
(a) QMCPack (b) RTM (time step 1500)

(c) Hurricane(W) (d) NYX(velocityz)

Fig. 3. Linear Regression Prediction Hyperplane with Different Error Bound Settings of Coefficients

when the compression ratio (CR) is relatively high (e.g., 196 and 568 for the two test cases). We observe that SZ2.1 suffers from a significant undesired block texture artifact, resulting from its blockwise linear-regression design.

To address the serious issue of the linear regression predictor, we developed a novel efficient predictor based on a



(a) Original data (QMCPack) (b) Decompressed data (QMCPack)

(c) Original data (RTM) (d) Decompressed data (RTM)

Fig. 4. Visualization of SZ Decompressed Data Based on Two Applications: (1) QMCPack – PSNR=56.2, CR=196, and (2) RTM – PSNR=50.7, CR=316

dynamic spline interpolation, such that compression quality (rate distortion) can be significantly improved for almost all application datasets, with little performance overhead.

V. Error-Bounded Lossy Compression with a Dynamic Multidimensional Spline Interpolation

We present the design overview in Fig. 5, with yellow rectangles indicating the differences between our design and the classic SZ compressor and with highlighted rectangles indicating the critical steps. The fundamental idea is to develop a **dynamic multidimensional spline interpolation**-based predictor (i.e., solution P2 shown in Fig. 5) to replace the linear-regression-based predictor such that the coefficient overhead can be completely eliminated while still keeping a fairly high prediction accuracy. Our newly designed interpolation-based predictor starts with one data point and performs interpolation and linear-scale quantization alternatively along each dimension recursively until all data points are processed. Two alternative approaches can be used to perform the interpolation in the multidimensional space. We can build a multidimensional curve to fit all the already-processed data points, or we can build multiple 1D curves to do the interpolation. We choose the latter because the former is much more expensive.

In what follows, we introduce the background of spline interpolation (Section V-A), followed by our design of dynamic multidimensional spline interpolation based predictor (Sections V-B, V-C, and V-D).

A. Introduction to Spline Interpolation

Interpolation is widely used in the field of engineering and science to construct new data points with a set of known data points. Interpolation techniques attempt to build a curve

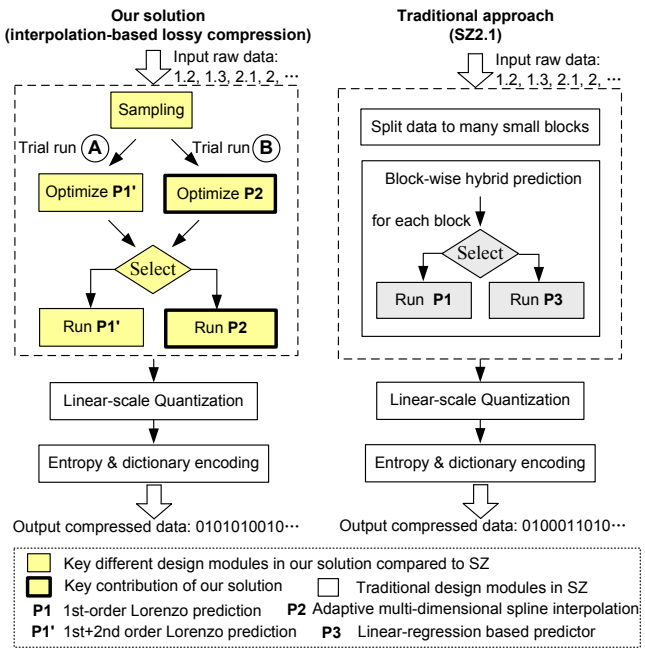


Fig. 5. Design Overview

that goes through all the known data points. It differs from regression analysis, which usually seeks a curve that most closely fits the known data points according to a specific mathematical criterion such as mean squared error. The curve generated by regression may not go through all known points.

The most popular interpolation methods can be categorized into three types: piecewise constant interpolation, polynomial interpolation, and spline interpolation. Piecewise constant interpolation always uses the nearest known data points to estimate the new data point, so it has a simple implementation and fast speed. However, its ability to estimate complex curves is limited because it does not consider the surrounding data points. Polynomial interpolation is designed to find a polynomial with the lowest possible degree that passes through all the known data points. If the number of known data points is large, the polynomial may suffer highly inaccurate oscillation between the data points. This issue is well known as *Runge's phenomenon* and could be mitigated by spline interpolation. Spline interpolation uses piecewise polynomials to define the estimation curve. If the degree of the polynomials is 1, the spline interpolation turns to **linear interpolation**. If the degree of polynomials is 3, it is known as **cubic spline interpolation**. Cubic spline polynomials have different restrictions. In this paper, we use not-a-knot restriction for cubic spline interpolation.

B. Spline Interpolation Designed for Scientific Data

In this sub-section, we introduce a basic interpolation method and derive closed-form formulas with the optimal coefficients, which is a fundamental work to the development of our following multi-dimensional interpolation predictor.

We propose a light-weight cubic interpolation based prediction method for each unknown data point by only using its four surrounding data values, to address the drawbacks of the conventional interpolation methods. The accuracy of polynomial interpolation could be affected significantly by Runge's phenomenon when interpolating across multiple regions with different locality features. Cubic spline interpolation can prevent large oscillation, but it has high computational cost as it needs to solve a huge linear system. To avoid high computation cost, we precompute a closed-form interpolation formula based on four specific neighbor data points (e.g., using the data points $i-3$, $i-1$, $i+1$ and $i+3$ to predict data point i as shown in Figure 6). In what follows, we mainly use a 1D example to illustrate how we derive the interpolation formula, but the formula can be extended to multidimensional cases easily.

Lemma 1: Denote the dataset as $d = (d_1, d_2, ..., d_n)$ with n as the total number of elements. The prediction values are denoted as p=$(p_1, p_2, ..., p_n)$. We consider all elements in odd-index positions as preknown and use them to predict the elements in even-index positions. The prediction formulas of linear and cubic spline interpolation are shown in Table I.

TABLE I
SPLINE ESTIMATIONS

Spline method	Prediction Value p_i
Linear spline	$p_i = \frac{1}{2}d_{i-1} + \frac{1}{2}d_{i+1}$
Cubic spline	$p_i = -\frac{1}{16}d_{i-3} + \frac{9}{16}d_{i-1} + \frac{9}{16}d_{i+1} - \frac{1}{16}d_{i+3}$

Proof: The linear formula is easy to derive, so we prove only the cubic spline formulas as follows. In our designed cubic spline interpolation, the known data points d_{i-3},d_{i-1},d_{i+1} and d_{i+3} are used to predict the data point p_i. Three spline curves correspond to the known data points:

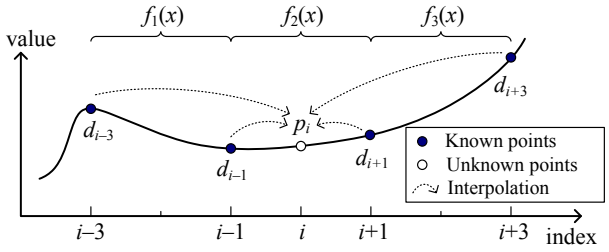


Fig. 6. Illustration of Cubic Spline Interpolation

$$\begin{aligned} f_1(x) &= a_1(x-(i-3))^3+b_1(x-(i-3))^2+c_1(x-(i-3))+\delta_1 \\ f_2(x) &= a_2(x-(i-1))^3+b_2(x-(i-1))^2+c_2(x-(i-1))+\delta_2 \\ f_3(x) &= a_3(x-(i+1))^3+b_3(x-(i+1))^2+c_3(x-(i+1))+\delta_3 \end{aligned} \quad (3)$$

The scope of f_1, f_2, and f_3 is $[i-3,i-1]$, $[i-1,i+1]$, and $[i+1,i+3]$ (as shown in Fig. 6). The spline curves should pass through the known data points, so we have

$$\begin{aligned} &f_1(i-3) = d_{i-3};\ f_1(i-1) = d_{i-1} \\ &f_2(i-1) = d_{i-1};\ f_2(i+1) = d_{i+1} \\ &f_3(i+1) = d_{i+1};\ f_3(i+3) = d_{i+3} \end{aligned} \quad (4)$$

The first derivatives of $f_1(x)$ is $f_1^{'}(x) = 3a_1(x-(i-3))^2 + 2b_1(x-(i-3))+c_1$. The second derivative is $f_1^{''}(x) = 6a_1(x-(i-3))+2b_1$. The third derivative is $f_1^{'''}(x) = 6a_1$. Derivatives of f_2 and f_3 are similar with f_1.

To have a smooth curve, we should let the adjacent spline functions have the same first derivatives and the same second derivatives on the joint data points.

$$f_1^{'}(i-1) = f_2^{'}(i-1);\ f_2^{'}(i+1) = f_3^{'}(i+1)$$
$$f_1^{''}(i-1) = f_2^{''}(i-1);\ f_2^{''}(i+1) = f_3^{''}(i+1) \quad (5)$$

The not-a-knot restriction requires the third derivative of f to be equal on locations $i-1$ and $i+1$.

$$f_1^{'''}(i-1) = f_2^{'''}(i-1);\ f_2^{'''}(i+1) = f_3^{'''}(i+1) \quad (6)$$

Using the system of Equations (4), (5), and (6), we can derive

$$\begin{aligned} a_2 &= -\tfrac{1}{48}d_{i-3} + \tfrac{1}{16}d_{i-1} - \tfrac{1}{16}d_{i+1} + \tfrac{1}{48}d_{i+3} \\ b_2 &= \tfrac{1}{8}d_{i-3} - \tfrac{1}{4}d_{i-1} + \tfrac{1}{8}d_{i+1} \\ c_2 &= -\tfrac{1}{6}d_{i-3} - \tfrac{1}{4}d_{i-1} + \tfrac{1}{2}d_{i+1} - \tfrac{1}{12}d_{i+3} \\ \delta_2 &= d_{i-1}. \end{aligned} \quad (7)$$

Thus the prediction value of pi will be

$$p_i = f_2(i) = -\tfrac{1}{16}d_{i-3} + \tfrac{9}{16}d_{i-1} + \tfrac{9}{16}d_{i+1} - \tfrac{1}{16}d_{i+3}. \quad (8)$$

Equation (8) is the cubic formula in Table I. ■

We discuss why we adopt only four known data points in our interpolation instead of six or more data points. If we use six data points d_{i-5}, d_{i-3}, d_{i-1}, d_{i+1}, d_{i+3}, and d_{i+5} to predict p_i, the formula by not-a-knot spline turns out to be

$$p_i = \tfrac{d_{i-5}}{80} - \tfrac{d_{i-3}}{10} + \tfrac{47}{80}d_{i-1} + \tfrac{47}{80}d_{i+1} - \tfrac{d_{i+3}}{10} + \tfrac{d_{i+5}}{80}. \quad (9)$$

Compared with Equation (8), Equation (9) involves two additional data points d_{i-5} and d_{i+5}, but the weight of the two data points is only $1/80$, which means a very limited effect on the prediction. Moreover, it has 50% higher computation cost compared with Equation (8). Hence, we choose to use four data points for prediction, as shown in Table I.

In addition, we note that the linear spline interpolation may exhibit better prediction accuracy than the cubic spline does when setting a relatively large error bound (as shown in Table II). The reason is that our interpolation method relies on the reconstructed data values generated after a linear-scale quantization step, so that the reconstructed data is lossy to a certain extent. When the error bound is relatively large, the loss of these reconstructed data would degrade the prediction accuracy, and the more data points used in the interpolation, the higher the impact on the accuracy. Since linear spline adopts fewer data points, it could be superior to cubic spline especially when the error bound is relatively large. This possibility motivated us to design a dynamic method selecting the better interpolation type (linear or cubic) in practice.

TABLE II
COMPARISON OF SPLINE METHODS PREDICTION ERROR

Dataset	$\epsilon = 1E-2$		$\epsilon = 1E-4$	
	Linear Spline	Cubic Spline	Linear Spline	Cubic Spline
RTM (time step 1500)	**1.20E-4**	1.27E-4	2.0E-5	**8.3E-6**
Miranda (velocityz)	**0.0026**	0.0025	0.0061	**0.0020**
QMCPACK	**0.05**	0.06	0.008	**0.004**
SCALE (QS)	**0.076**	0.078	**0.040**	0.041
NYX (velocityz)	**123486**	134820	22453	**19978**
Hurricane (W)	**0.04**	0.05	0.023	**0.022**

C. Multilevel Multidimensional Spline Interpolation

The previous derivation works in the 1D case with 50% of preknown data points, based on which we predict the other 50%. In this section, we extend this interpolation method to support data prediction on the entire multidimensional dataset.

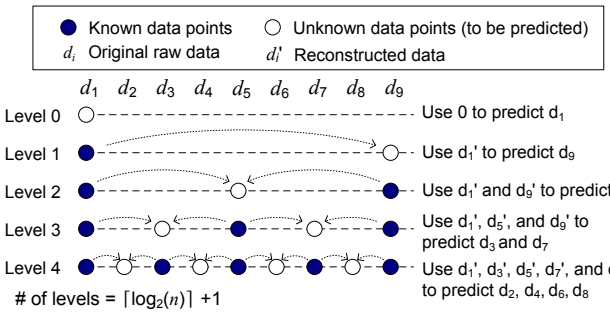


Fig. 7. Illustration of Multilevel Linear Spline Interpolation

We use Fig. 7 to demonstrate the multilevel solution with linear interpolation; cubic interpolation has the same multilevel design. Suppose the dataset has n elements in one dimension. The number of levels required to cover all elements in this dimension is $l = 1 + \lceil log_2 n \rceil$. At level 0, we use 0 to predict d_1, followed by the error-bounded linear-scale quantization. We perform a series of interpolations from level 1 to level $l-1$ recursively, as shown in Fig. 7. At each level, we use error-bounded linear-scale quantization to process the predicted value such that the corresponding reconstructed data must be within the error bound. We denote the reconstructed data as $d_1', d_2', ..d_n'$, as shown in the figure.

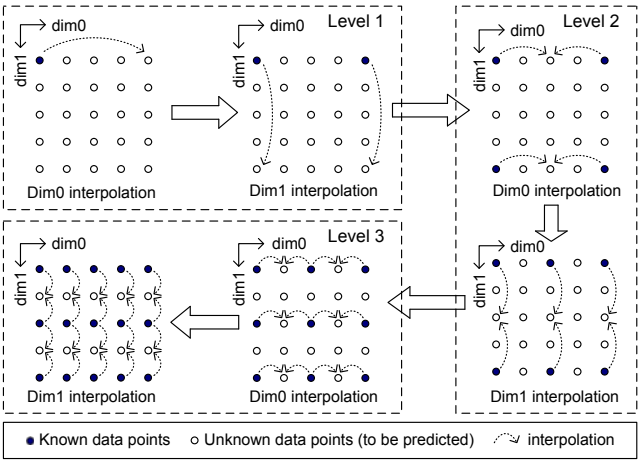


Fig. 8. Illustration of Multidimensional Linear Spline Interpolation

Such a multilevel interpolation is applied on a multidimensional dataset, illustrated in Fig. 8 with a 2D dataset as an example. We perform interpolation separately along all dimensions at each level, with a fixed sequence of dimensions. A 2D dataset, for example, has two possible sequences: $dim_0 \rightarrow dim_1$ and $dim_1 \rightarrow dim_0$. A 3D dataset has 6 possible sequences. In our solution, we propose to check only two sequences, $dim_0 \rightarrow dim_1 \rightarrow dim_2$ (sequence 1) and $dim_2 \rightarrow dim_1 \rightarrow dim_0$ (sequence 2) instead of all 6 possible combinations. On the one hand, the last interpolation dimension involves about 50% of the data points (much more than other dimensions), so which dimension to be put in the end of the sequence determines the overall prediction accuracy. On the other hand, we note that either the highest or lowest dimension in scientific datasets tends to be smoother than other

dimensions without loss of generality, as confirmed by the first three columns of Table III (with all 6 applications), which presents the autocorrelation (AC) of each dimension (higher AC means smoother data). Accordingly, putting either dim0 or dim2 in the end of the sequence at each level will get lower overall prediction errors, as validated in Table III. Hence, we also develop a dynamic strategy to select the best-fit sequence of dimensions from among the two candidates, as detailed in the next subsection.

TABLE III
AUTOCORRELATION AND PREDICTION ERROR OF CUBIC SPLINE INTERPOLATION WITH DIFFERENT SEQUENCES OF DIMENSION SETTINGS, ϵ=1E−3

Dataset	Autocorrelation (Lag=4)			Prediction Error		
	dim2	dim1	dim0	0→1→2	0→2→1	2→1→0
RTM (time step 1500)	**0.88**	0.58	0.45	**2.17E-5**	2.32E-5	2.51E-5
Miranda (velocityz)	0.84	0.82	**0.96**	0.004	0.004	**0.003**
QMCPACK	**0.83**	0.83	0.75	**0.010**	0.010	0.013
SCALE (QS)	**0.987**	0.986	0.872	**0.0447**	0.0448	0.10
NYX (velocityz)	0.9818	0.99	**0.99**	31668	29903	**28975**
Hurricane (W)	0.19	0.027	**0.86**	0.024	0.025	**0.016**

D. Dynamic Optimization Strategies

In this section we propose a dynamic design with two adaptive strategies: (1) automatically optimizing the spline interpolation predictor (Trial run Ⓑ in Fig. 5) by selecting the best-fit interpolation type (either linear or cubic) and optimizing the sequence of interpolation dimensions and (2) automatically selecting the better predictor between the Lorenzo-based predictor (Trial run Ⓐ in Fig. 5) and the interpolation predictor.

We use a uniform sampling method to determine the best interpolation settings for the input dataset. There are two settings to optimize for the multidimensional interpolation predictor: the interpolation type and the dimension sequence. We adopt a uniform sampling method with only 3% total data points to select the better interpolation type with the higher compression ratio.

We note that the spline interpolation predictor does not work as effectively as the multilayer Lorenzo predictor [17], [28] on the relatively nonsmooth dataset, especially when the user's error bound is relatively small (as shown in Table IV). As a result, our final solution is selecting the better predictor from our spline interpolation method and Lorenzo method.

TABLE IV
PREDICTION ERROR OF MULTIDIMENSIONAL SPLINE INTERPOLATION PREDICTOR (S), REGRESSION PREDICTOR (R), AND LORENZO PREDICTOR (L)

Dataset	$\epsilon = 1E-2$			$\epsilon = 1E-7$		
	S	R	L	S	R	L
RTM (time step 1500)	**1.2E-4**	1.3E-4	2.0E-4	6.9E-6	1.0E-4	**1.8E-7**
Miranda (velocityz)	**0.02**	0.03	0.05	0.001	0.02	**6E-5**
QMCPACK	**0.05**	0.06	0.13	0.004	0.03	**6E-4**
SCALE (QS)	**0.07**	0.16	0.11	0.04	0.15	**0.01**
NYX (velocityz)	**121436**	132071	410083	**15237**	51963	16965
Hurricane (W)	**0.04**	0.05	0.06	0.01	0.04	**0.004**

VI. EXPERIMENTAL EVALUATION

In this section we present the experimental setup and discuss the evaluation results and our analysis.

A. Experimental Setup

1) Execution Environment: We perform the experiments on the Argonne Bebop supercomputer. Each node in Bebop is driven by two Intel Xeon E5-2695 v4 processors with 128 GB of DRAM.

2) Applications: We perform the evaluation using six real-world scientific applications from different domains:

- QMCPack: An open source ab initio quantum Monte Carlo package for the electronic structure of atoms, molecules, and solids [46].
- RTM: Reverse time migration code for seismic imaging in areas with complex geological structures [30].
- NYX: An adaptive mesh, cosmological hydrodynamics simulation code.
- Hurricane: A simulation of a hurricane from the National Center for Atmospheric Research in the United States.
- Scale-LETKF: Local Ensemble Transform Kalman Filter (LETKF) data assimilation package for the SCALE-RM weather model.
- Miranda: A radiation hydrodynamics code designed for large-eddy simulation of multicomponent flows with turbulent mixing.

Detailed information about the datasets (all using single precision) is presented in Table V. Some data fields are transformed to their logarithmic domain before compression for better visualization, as suggested by domain scientists.

TABLE V
BASIC INFORMATION ABOUT APPLICATION DATASETS

App.	# files	Dimensions	Total Size	Domain
RTM	3600	449×449×235	635GB	Seismic Wave
Miranda	7	256×384×384	1GB	Turbulence
QMCPACK	1	288×115×69×69	612MB	Quantum Structure
Scale-LETKF	13	98×1200×1200	6.4GB	Climate
NYX	6	512×512×512	3.1GB	Cosmology
Hurricane	48×13	100×500×500	58GB	Weather

3) State-of-the-Art Lossy Compressors in Our Evaluation: In our experiment we compare our new compressor with five other state-of-the-art error-bounded lossy compressors (SZ2.1 [18], ZFP0.5.5 [15], SZ(Hybrid) [27], SZ(SP+PO)[1] [28] and MGARDx [43]), which have been recognized as the best in class (validated by different researchers [18], [21], [25]).

4) State-of-the-Art Lossless Compressors in Our Evaluation: We also evaluate six lossless compressors including Google Brotli [49], Google Snappy [35], Facebook Zstandard [37], LZMA [50], Zlib [16], and Fpzip [51] in our experiment as a comparison with lossy compressors. Brotli, Snappy and Zstandard (Zstd) are depolyed in many industrial data management systems. LZMA is the default compression method of 7-Zip. Zlib [16] is one of the most widely used compressor in operating systems. Fpzip [51] is a compressor targeted at floating-point data.

5) Evaluation Metrics: We evaluate the six lossy compressors based on five critical metrics, as described below.

- Compression ratio (CR) based on the same error bound: The descriptions of CR and absolute error bound are de-

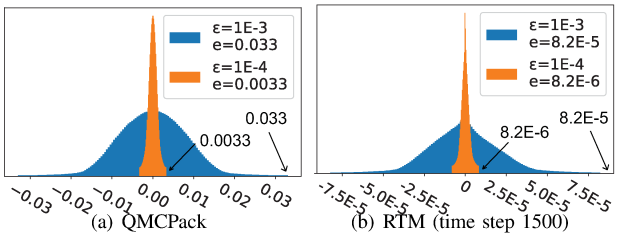
[1]SZ(SP+PO) represents the SZ compression model with 2nd-order prediction (SP) and parameter optimization (PO), suffering 1X slower compression.

fined in Section III. Without loss of generality, we adopt value-range-based error bound (denoted as ϵ), which takes the same effect with absolute error bound (denoted e) because $e = \epsilon(\max(D) - \min(D))$.

- Compression speed and decompression speed: $\frac{original\ size}{compression\ time}$ (MB/s) and $\frac{reconstructed\ size}{decompression\ time}$ (MB/s).
- *Rate-distortion*: The detailed description is in Section III.
- Visualization with the same CR: Compare the visual quality of the reconstructed data based on the same CR.
- Precision of final execution results of RTM data with lossy compression.

B. Evaluation Results and Analysis

First, we verified the maximum compression errors for all six lossy compressors based on all the application datasets with different error bounds. Experiments confirm that they all respect the error bound constraint very well. Fig. 9 shows the distribution of compression errors of our solution on two error bounds (ϵ=1E-3 and ϵ=1E-4, in other words, e=0.033&0.0033 for QMCPACK and e=8.2E-5&8.2E-6 for RTM). We can clearly see that the compression errors are 100% within the absolute error bound (e) for all data points.



(a) QMCPack (b) RTM (time step 1500)

Fig. 9. Compression Error Distribution of Our Solution

Table VI presents the compression ratios of the six lossy compressors based on the six real-world applications with the same error bounds. We can clearly observe that our solution always exhibits the highest compression ratio in all cases. In particular, the compression ratio of our solution is higher than other compressors by 20%∼460% in most cases. For example, when setting the error bound to 1E-3 for compressing RTM data, the second-best compressor ((SZ(SP+PO)) gets a compression ratio of 114.4, while our compressing ratio reaches up to 397.6 (with a 247.5% improvement). The key reason our solution can get a significantly higher compression ratio is twofold: (1) we significantly improve the prediction accuracy by a dynamic spline interpolation, and (2) some other compressors such as ZFP and MGARDx suffer from the precision-overpreservation issue (i.e., the actual maximum errors are smaller than the required error bound, as verified by prior works [14], [17], [43].

Table VII compares the compression/decompression speed among all six lossy compressors for all six applications. It clearly shows that our solution exhibits compression performance similar to that of SZ2.1 and MGARDx, and its decompression performance is also comparable to that of SZ2.1 and is about 30% higher than that of MGARDx.

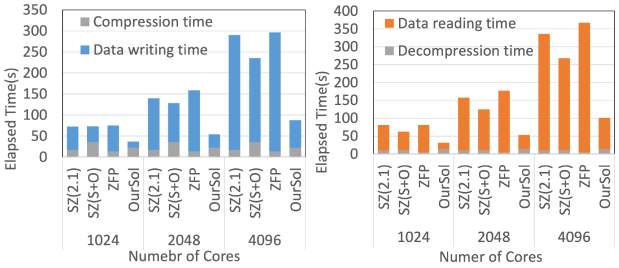
Data smoothness and error bound settings are two key factors that affect the compression ratio and speed. In the SZ framework, datasets with better local smoothness or with larger error bound settings will result in smaller quantization data value range and more close-to-zero quantized data values. In general, this will get higher compression ratios and speed because such quantized data turns much easier to be compressed by the succeeding encoding steps. This analysis also applies to other lossy compressors such as ZFP and MGARDx that utilize the coding stage.

TABLE VI
COMPRESSION RATIO COMPARISON BASED ON THE SAME ERROR BOUND

Dataset	ϵ	SZ 2.1	SZ (Hybrid)	SZ (SP+PO)	ZFP	MGARDx	OurSol	OurSol Improve %
RTM	1E-2	271.7	195.7	358.1	111.0	229.7	**1997.5**	457%
	1E-3	109.8	101.4	114.4	59.3	78.1	**397.6**	247%
	1E-4	57.3	44.4	63.0	34.9	38.3	**116.3**	84%
Miranda	1E-2	125.6	130.4	188.4	46.6	113.7	**582.1**	209%
	1E-3	59.9	55.4	58.4	25.6	38.0	**160.7**	168%
	1E-4	30.6	23.4	33.9	14.5	20.0	**47.1**	39%
QMCPack	1E-2	196.2	144.8	174.5	39.4	159.8	**675.5**	244%
	1E-3	51.1	53.4	68.0	21.2	47.1	**204.3**	200%
	1E-4	18.9	24.9	23.6	10.4	14.9	**63.7**	155%
SCALE	1E-2	84.3	94.2	108.2	14.5	52.8	**157.0**	45%
	1E-3	26.6	27.1	31.8	7.8	20.2	**40.5**	27%
	1E-4	14.0	13.2	14.1	4.6	10.4	**14.9**	5%
NYX	1E-2	43.6	33.2	48.7	12.0	24.7	**59.4**	22%
	1E-3	16.8	16.3	17.4	6.0	11.2	**21.1**	21%
	1E-4	7.6	8.0	8.1	3.7	5.5	**9.1**	12%
Hurricane	1E-2	49.4	44.6	65.4	11.3	28.1	**69.3**	6%
	1E-3	17.6	17.9	19.8	6.7	12.7	**22.5**	14%
	1E-4	9.8	10.1	10.5	4.3	7.4	**10.8**	3%

TABLE VII
COMPRESSION/DECOMPRESSION SPEEDS (MB/S) WITH ϵ=1E-3

Type	Dataset	SZ 2.1	SZ (Hybrid)	SZ (SP+PO)	ZFP	MGARDx	OurSol
Compression	RTM	207	76	97	549	128	149
	Miranda	125	73	91	201	140	128
	QMCPack	146	63	78	158	136	133
	SCALE	145	59	75	101	122	128
	NYX	123	81	86	131	117	110
	Hurricane	115	63	78	115	122	131
Decompression	RTM	385	299	298	984	173	276
	Miranda	285	221	206	531	177	232
	QMCPack	327	232	282	367	168	241
	SCALE	271	184	192	295	164	215
	NYX	222	172	215	244	145	136
	Hurricane	222	186	200	257	163	193



(a) Data dumping performance (b) Data loading performance

Fig. 10. Parallel Performance Evaluation of QMCPack Simulation (SP(S+O) stands for SP(SP+PO))

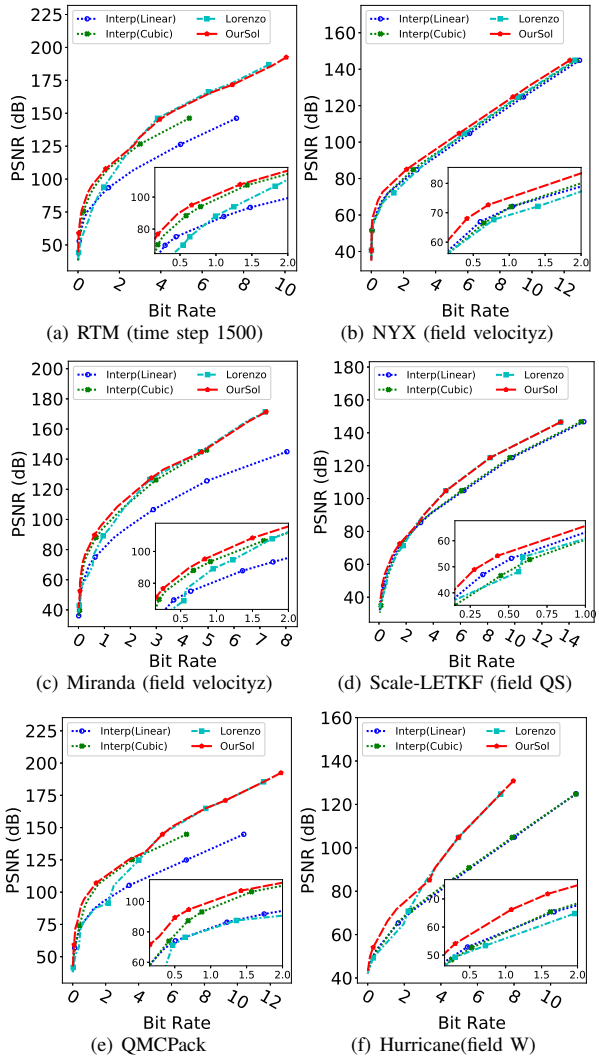
We evaluate the data dumping and loading performance of the QMCPack simulation when using lossy compressors to demonstrate the performance impact of lossy compressors on scientific simulations. SZ2.1, SZ(SP+PO), ZFP, and our solution are assessed under the same level of data distortion (PSNR fixed to 70). The evaluation uses up to 4096 cores and each core processes 3.4GB of data. Fig. 10 shows that our solution leads to the highest data dumping and loading performance. In the scale of 4096 cores, QMCPACK simulation needs more than 3 hours to dump the data to disk without the help of lossy

compressors. Our solution reduces the elapsed time to less than 100 seconds and it is 1.7X faster than the second-best one.

Table VIII demonstrates the compression ratios of the six lossless compressors. It confirms our statement in Section II that lossless compressors have limited compression ratios on scientific datasets. Lossy compressors, on the other hand, can achieve much higher compression ratio as shown in Table VI.

TABLE VIII
COMPRESSION RATIO COMPARISON OF LOSSLESS COMPRESSORS

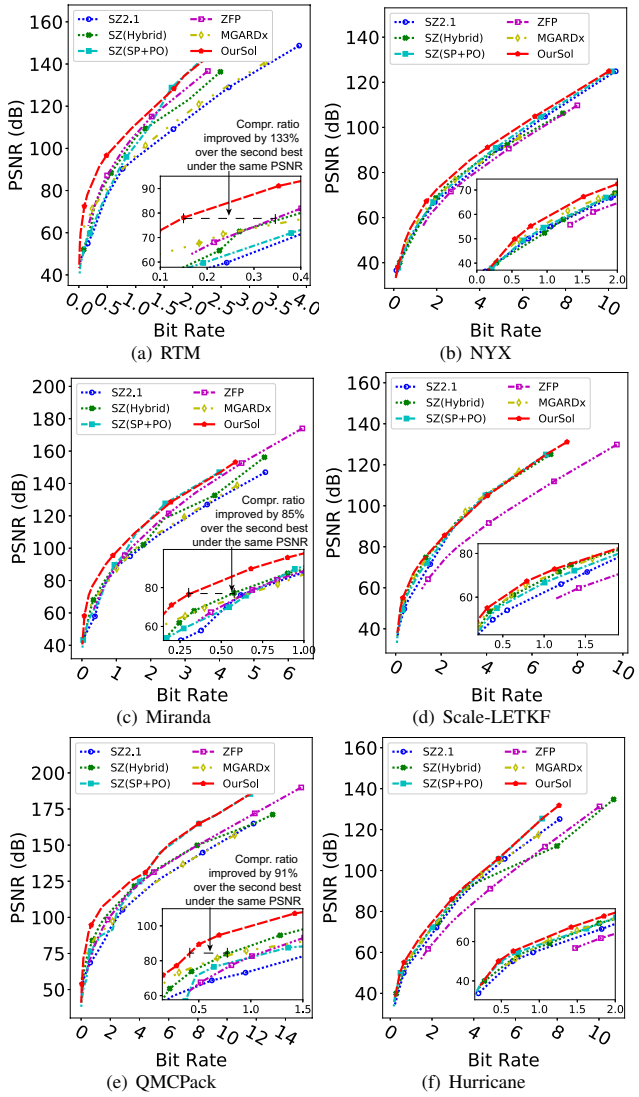
Dataset	Brotli	Zstd	Snappy	Fpzip	Zlib	LZMA
RTM	2.04	2.02	1.87	2.62	2.04	2.18
Miranda	1.21	1.21	1.11	1.86	1.21	1.30
QMCPack	1.19	1.19	1.01	1.75	1.20	1.51
SCALE	1.45	1.39	1.17	2.60	1.39	1.80
NYX	1.19	1.11	1.00	1.37	1.11	1.25
Hurricane	1.52	1.49	1.26	2.28	1.49	1.78



(a) RTM (time step 1500) (b) NYX (field velocityz)
(c) Miranda (field velocityz) (d) Scale-LETKF (field QS)
(e) QMCPack (f) Hurricane(field W)

Fig. 11. Our Solution Compared with Interpolation and Lorenzo

As discussed in Section V-D, we designed a dynamic strategy to optimize the compression quality throughout the entire bit-rate range. Fig. 11 demonstrates that the dynamic strategy has a critical effect in the compression quality improvement. For instance, as shown in Fig. 11 (a), our solution always exhibits the best compression quality when the bit rate is lower than 2.5 because it adopts a dynamic interpolation method with optimized dimension sequences on a multilevel interpolation, whereas both linear interpolation and tricubic interpolation (shown in the figure) use a fixed sequence. ($z \rightarrow y \rightarrow x$). On the other hand, Fig. 11 (a) shows that our solution also keeps the best rate-distortion level when the bit rate is higher than 2.5, a result that is attributed to our accurate predictor selection algorithm (selecting a better predictor between interpolation and Lorenzo at runtime).



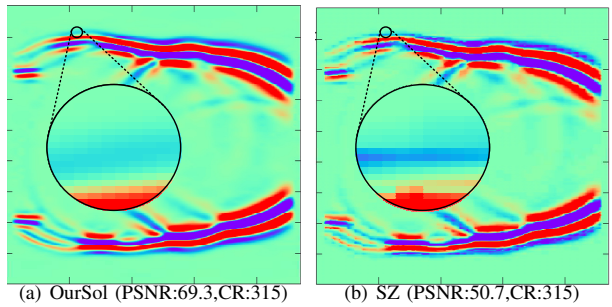
(a) RTM (b) NYX
(c) Miranda (d) Scale-LETKF
(e) QMCPack (f) Hurricane

Fig. 12. Overall Evaluation (Lower Bit Rate / Higher PSNR → Better Quality)

Fig. 12 presents the overall compression quality (i.e., rate distortion). One can see that our solution is the best in class from among all the related works for all six applications. In particular, with the same data-distortion level (PSNR), the compressed data size under our solution is about 50% of the compressed data size under the second-best compressor

in most of the cases for RTM, Miranda, and QMCPack.

We demonstrate the visual quality of the decompressed data of four error-bounded lossy compressors in Fig. 13, using one slice image (slice 340) in the RTM dataset. The original visualization is shown in Fig. 4. The figure clearly shows that our solution keeps an excellent visual quality in the decompressed data with a compression ratio even up to 315. In contrast, other compressors suffer from prominent degradation in visual quality to different extents with the same compression ratios. In particular, SZ and ZFP suffer from undesired blockwise texture artifacts.



(a) OurSol (PSNR:69.3,CR:315) (b) SZ (PSNR:50.7,CR:315)

(c) ZFP (PSNR:51.7,CR:258) (d) MGARDx (PSNR:62.5,CR:310)

Fig. 13. Visualization of Decompressed Snapshot Data (RTM)

We show in Fig. 14 and Fig. 15 that the final RTM image for a single shot is not degraded at all using our lossy compressor with very high compression ratios (about 2∼4× higher than that of other compressors). We use value-range-based error bound 1.25E-3 in our solution for each time step. The RTM application requires propagating waves generated by a source signal, in a given subsurface model. At the beginning of the propagation the compression ratios are very high (10k+) when the waves are close to the source locations. Over time, the waves are propagating further in the model, resulting in more complex images and compression ratios dropping to about 70. The overall compression ratio is 274 because the compression ratio at most time steps can reach 300+ (e.g., CR=315 at time step 1500 as shown in Fig. 13 (a)). In this simulation we used one shot to generate the final image in Fig. 15. One can see a very good preservation of amplitudes and main structures of the lossy-compression-based final result, which is acceptable for post-analysis as confirmed by the seismic researchers. Our lossy compressor dramatically decreases the size of the RTM snapshots while not increasing the computation time compared with SZ 2.1. This can significantly lower the I/O throughput requirements and enable either faster turnaround or higher-fidelity simulations for production-level seismic imaging.

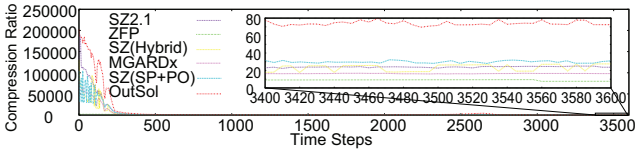
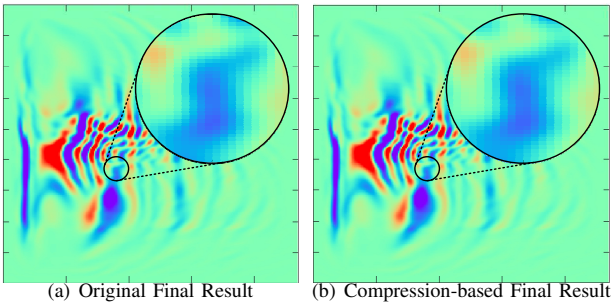


Fig. 14. Compression Ratio of RTM Data for Different Time Steps (with Value-Range-Based Error Bound 1.25E-3)



(a) Original Final Result (b) Compression-based Final Result

Fig. 15. Visualization of RTM Image for One Shot

VII. Conclusion and Future Work

In this paper we present a novel error-bounded lossy compressor based on the SZ framework. We develop a dynamic spline interpolation approach with adaptive optimization strategies. We thoroughly evaluate the compression quality and performance of our solution compared with that of five other lossy compressors on six real-world scientific simulations. The key findings are summarized below.

- Our analysis shows that the linear regression predictor has a significant problem because its coefficient overhead is non-negligible (25%∼70% in compressed data).
- Our dynamic spline interpolation solution can improve the compression ratio by 457%, 244%, and 209% compared with the second-best compressor on RTM, QMCPACK, and Miranda datasets, respectively.
- Our solution has high compression/decompression performance comparable to that of SZ2.1. Its compression speed is 28%∼100% faster than other SZ-based methods such as SZ(Hybrid) and SZ(SP+PO).
- Our solution keeps an extremely high visual quality in the decompressed data, whereas other lossy compressors suffer from prominent degradation in visualization with the same compression ratios.

In the future work, we plan to improve the compression quality by exploring more effective prediction models and improve the performance by optimizing the code implementation.

VIII. Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant No. 1619253 and No. 2003709. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

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

[2] S. Habib *et al.*, "HACC: extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

[3] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *SIGMOD Rec.*, vol. 34, no. 4, p. 34–41, Dec. 2005.

[4] Y. Cheng and F. Rusu, "Parallel in-situ data processing with speculative loading," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, p. 1287–1298.

[5] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou, "The researcher's guide to the data deluge: Querying a scientific database in just a few seconds," *Proc. VLDB Endow.*, vol. 4, no. 12, p. 1474–1477, Aug. 2011.

[6] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki, "Nodb: Efficient query execution on raw data files," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12, 2012, p. 241–252.

[7] S. Li *et al.*, "Towards end-to-end SDC detection for HPC applications equipped with lossy compression," in *2020 IEEE International Conference on Cluster Computing*, 2020, pp. 326–336.

[8] Globus. [Online]. Available: https://www.globus.org/

[9] R. Rew and G. Davis, "NetCDF: an interface for scientific data access," *IEEE Computer Graphics and Applications*, vol. 10, pp. 76–82, 1990.

[10] HDF5. [Online]. Available: http://www.hdfgroup.org/HDF5

[11] A. Ailamaki, V. Kantere, and D. Dash, "Managing scientific data," *Commun. ACM*, vol. 53, no. 6, p. 68–78, Jun. 2010.

[12] S. Cohen *et al.*, "Scientific formats for object-relational database systems: A study of suitability and performance," *SIGMOD Rec.*, vol. 35, no. 2, p. 10–15, Jun. 2006.

[13] The HDF Group. (2017) H5Z: Filter and Compression Interface. https://support.hdfgroup.org/HDF5/doc1.8/RM/RM_H5Z.html. Online.

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

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

[16] Zlib, http://www.zlib.net/, online.

[17] 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*, 2017, pp. 1129–1139.

[18] X. Liang *et al.*, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," *2018 IEEE International Conference on Big Data (Big Data)*, pp. 438–447, 2018.

[19] S. Li *et al.*, "SDC resilient error-bounded lossy compressor," https://arxiv.org/abs/2010.03144, 2020, online.

[20] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '15, 2015, pp. 914–922.

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

[22] X. Liang *et al.*, "Improving performance of data dumping with lossy compression for scientific simulation," in *2019 IEEE International Conference on Cluster Computing*, 2019, pp. 1–11.

[23] N. Kukreja, J. H. uuckelheim, M. Louboutin, J. Washbourne, P. H. Kelly, and G. J. Gorman, "Lossy checkpoint compression in full waveform inversion," https://arxiv.org/pdf/2009.12623.pdf, 2020, online.

[24] J. Tian *et al.*, "Revisiting huffman coding: Toward extreme performance on modern gpu architectures," https://arxiv.org/abs/2010.10039, 2020, online.

[25] T. Lu *et al.*, "Understanding and modeling lossy compression schemes on HPC scientific data," in *2018 IEEE International Parallel and Distributed Processing Symposium*, 2018, pp. 348–357.

[26] J. Tian *et al.*, "CuSZ: An efficient gpu-based error-bounded lossy compression framework for scientific data," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '20, 2020, p. 3–15.

[27] X. Liang *et al.*, "Significantly improving lossy compression quality based on an optimized hybrid prediction model," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–26.

[28] K. Zhao *et al.*, "Significantly improving lossy compression for HPC datasets with second-order prediction and parameter optimization," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '20, 2020, pp. 89–100.

[29] J. Tian *et al.*, "Wavesz: A hardware-algorithm co-design of efficient lossy compression for scientific data," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2020, pp. 74–88.

[30] S. Kayum *et al.*, "GeoDRIVE - a high performance computing flexible platform for seismic applications," *First Break*, vol. 38, no. 2, pp. 97–100, 2020.

[31] A. M. Gok *et al.*, "PaSTRI: A novel data compression algorithm for two-electron integrals in quantum chemistry," in *IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 1–11.

[32] T. Pelkonen *et al.*, "Gorilla: A fast, scalable, in-memory time series database," *Proc. VLDB Endow.*, vol. 8, no. 12, p. 1816–1827, Aug. 2015.

[33] X. Yu *et al.*, "Two-level data compression using machine learning in time series database," in *36th IEEE International Conference on Data Engineering*, 2020, pp. 1333–1344.

[34] P. Cudre-Mauroux *et al.*, "A demonstration of SciDB: A science-oriented DBMS," *Proc. VLDB Endow.*, vol. 2, no. 2, p. 1534–1537, Aug. 2009.

[35] Snappy, https://google.github.io/snappy, 2018, online.

[36] InfluxDB, https://github.com/influxdata/influxdb, 2015, online.

[37] Zstandard, https://github.com/facebook/zstd/releases, online.

[38] K. Zhao *et al.*, "SDRBench: Scientific data reduction benchmark for lossy compressors," https://arxiv.org/abs/2101.03201, 2021, online.

[39] D. Taubman and M. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer, 2013.

[40] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "TTHRESH: Tensor compression for multidimensional visual data," *IEEE Transactions on Visualization & Computer Graphics*, vol. 26, no. 09, pp. 2891–2903, sep 2020.

[41] S. Chandak, K. Tatwawadi, C. Wen, L. Wang, J. Aparicio Ojea, and T. Weissman, "LFZip: Lossy compression of multivariate floating-point time series data via improved prediction," in *2020 Data Compression Conference (DCC)*, 2020, pp. 342–351.

[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 *et al.*, "MGARD+: Optimizing multilevel methods for error-bounded scientific data reduction," https://arxiv.org/abs/2010.05872, 2020, online.

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

[45] 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.

[46] J. Kim *et al.*, "QMCPACK: an open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids," *Journal of Physics: Condensed Matter*, vol. 30, no. 19, p. 195901, 2018.

[47] Hurricane ISABEL simulation data, https://www.earthsystemgrid.org/dataset/isabeldata.html, 2016, online.

[48] NYX simulation, https://amrex-astro.github.io/Nyx, 2019, online.

[49] J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka, and L. Vandevenne, "Brotli: A general-purpose data compressor," *ACM Trans. Inf. Syst.*, vol. 37, no. 1, Dec. 2018.

[50] Igor Pavlov, https://www.7-zip.org, online.

[51] 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.