2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC) | 979-8-3503-8322-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/HIPC58850.2023.00029

SECRE: Surrogate-based Error-controlled Lossy Compression Ratio Estimation Framework

Arham Khan,* Sheng Di,† Kai Zhao,‡ Jinyang Liu,§ Kyle Chard,* Ian Foster,* Franck Cappello†¶
*University of Chicago, Chicago, IL, USA
†Argonne National Laboratory, Lemont, IL, USA
‡Florida State University, Tallahassee, FL, USA
§ University of California, Riverside, Riverside, CA, USA
¶University of Illinois Urbana-Champaign, Urbana, IL, USA
arham@uchicago.edu, sdi1@anl.gov, kzhao@cs.fsu.edu, jinyang.liu@ucr.edu, chard@uchicago.edu,
foster@anl.gov, cappello@mcs.anl.gov

***Abstract*—Error-controlled lossy compression has been effective in reducing data storage/transfer costs while preserving reconstructed data fidelity based on user-defined error bounds. State-of-the-art error-controlled lossy compressors primarily focus on error control rather than compression size, and thus, compression ratios are unknown until the compression operation is fully completed. Many use cases, however, require knowledge of compression ratios a priori, for example, pre-allocating appropriate memory for the compressed data at runtime. In this paper, we propose a novel, efficient Surrogate-based Error-controlled Lossy Compression Ratio Estimation Framework (SECRE), which includes three key features/contributions. (1) We carefully design the SECRE framework, which, in principle, can be applied to different error-bounded lossy compressors. (2) We implement a compression ratio estimation method for four state-of-the-art error-controlled lossy compressors—SZx, SZ3, ZFP, and SPERR—by devising a corresponding lightweight compression surrogate for each. (3) We evaluate the performance and accuracy of SECRE using four real-world scientific simulation datasets. Experiments show that SECRE can obtain highly accurate compression ratio estimates (e.g., ∼1% estimation errors for SZx) with low execution overhead (e.g., ∼2% estimation cost for SZx).**

***Index Terms*—error-controlled lossy compression, scientific datasets, compression ratio estimation, sampling**

I. INTRODUCTION

Scientific simulations and advanced instruments produce enormous amounts of data that are relied upon for post hoc analysis. For instance, the most widely used climate simulation package—Community Earth System Model (CESM) [1]—can produce 300+ TB of data in the first 30 ensemble simulations. Significantly reducing the size of scientific datasets can address these and other substantial issues [2] such as inadequate storage space, limited I/O or network bandwidth, and insufficient memory capacity to run simulations. Error-controlled lossy compression (or error-bounded compression) [3]–[7] has been arguably the most efficient method to resolve issues associated with big data, because not only can it obtain high compression ratios (e.g., 100X) but it can also allow users to control the data distortion based on a specified error bound.

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

Existing error-controlled lossy compressors, are typically designed towards a certain error-control mode (such as limiting point-wise error using the absolute error, relative error, or peak signal to noise ratio (PSNR) [8]), which leaves a significant gap for use in practice: users must foresee the compression ratio before compression is completed in many use cases. Here, we present three practical use cases that motivate the need to predict compression ratios: **Use-case 1:** Parallel HPC simulations (e.g. CESM, Hurricane-ISABEL [1], [9]) produce enormous volumes of data that must be stored for later use. Unfortunately, file systems (and user allocations) are limited and thus it is crucial to set the compression ratio with a strict lower bound to avoid simulation failures due to insufficient storage. **Use-case 2:** error-bounded lossy compression - where floating point error in reconstructed data is limited according to user requirements - has been explored to reduce data transfer costs on wide area networks (WANs) [10], [11] to foresee the data transfer time (to determine whether it is worthwhile to compress data), accurately predicting the compression ratio is a critical step. **Use-case 3:** In many I/O or data communication use-cases that leverage data compression techniques, user code must allocate a fixed amount of memory beforehand to hold the compressed data, which also requires an accurate estimation of compression ratio.

In this paper, we aim to develop an efficient online error-controlled compression ratio estimation framework. The estimation framework should be generic and compatible with different error-bounded lossy compression models. The fundamental idea is to sample a small portion of data from the full dataset and emulate the compression operation (via a 'surrogate function') on the sampled dataset, such that the compression ratio can be inferred accurately with little computational cost.

Such an estimation framework, in principle, can be applied to any error-controlled lossy compressors to estimate compression ratios accurately. However, we must address three grand challenges. 1) An efficient sampling method needs to be devised carefully. The sampling rate should be minimal (to minimize execution overhead), while also covering as many regions of the full dataset as possible (to maximize the

2640-0316/23/$31.00 ©2023 IEEE
DOI 10.1109/HiPC58850.2023.00029

accuracy of the prediction). Moreover, the sampling method must be compatible with the principle of the corresponding compressor's design, which will be analyzed in detail later in the paper. 2) The design of the surrogate compressor must be customized based on the specific lossy compression principles, which requires an in-depth and comprehensive investigation of the target lossy compressors, requiring significant research and development effort. 3) The surrogate functions must not only be accurate but also lightweight in order to obtain an efficient, accurate estimation.

Our key contributions are three-fold:

- We propose a generic online surrogate-based compression ratio estimation framework (SECRE) which is the first fast, generic runtime lossy compression ratio estimation framework to the best of our knowledge. Note that the SECRE framework is not a trial-and-error method (unlike FraZ/OptZConfig [12], [13]) and does not depend on the characterization of compressors' internal data features (unlike Lu et al's model for SZ1.4 [14]) or preliminary model training based on masses of existing datasets.
- We design and implement lightweight surrogates for four cutting-edge error-controlled lossy compressors (SZx [5], SZ3 [4], [15], ZFP [6] and SPERR [16]). All four customized algorithms can achieve highly accurate compression ratio estimation with little computation overhead.
- We comprehensively evaluate SECRE based on the four cutting-edge error-controlled lossy compressors, using four real-world scientific simulation datasets, to show the advantage of our solution over prior work.

The remainder of the paper is organized as follows. Section II discusses related work. Section III formulates the research problem. Section IV presents the design overview. Section V describes how we apply SECRE to different lossy compressors for obtaining fast, accurate ratio estimation. Section VI presents and analyzes the performance of SECRE on real-world datasets. Finally, Section VII summarizes our work.

II. Related Work

In this section, we discuss contemporary compression ratio estimation work in two relevant categories: image-based lossy compression and error-controlled lossy compression.

The compressibility of image data via lossy compression has been studied for decades. Rate–distortion theory [17] provides a theoretical analysis of the compression ratio that can be achieved under a lossy compression method (a.k.a., lossy source coding). Blahut and Arimoto [18], [19] proposed the Blahut–Arimoto algorithm, which is an iterative algorithm to compute the rate distortion for lossy compression. Smieja and Tabor [20] performed a theoretical analysis for estimating the entropy of the lossy compression especially in the context of image compression, also considering the permitted distortion on any symbol under an error-control family. These theoretical analyses, however, cannot be used to estimate the compression ratios for specific error-controlled lossy compressors because of their specific design principles with diverse features.

Recently, there emerged a few studies on the estimation of compression ratios of error-controlled lossy compressors for scientific datasets. Lu et al. [14] proposed a sampling-based compression ratio estimation model for ZFP [6] and SZ1.4 [21]. There are several key differences between their model and our framework. (1) Their model focused on only absolute error bound, while we investigate several error-control modes. (2) Their model treats the dataset as a 1D array (e.g., perform a 1D random block sampling), which may cause large estimation errors for multi-dimensional datasets (as shown later in this paper). (3) For SZ1.4, they use a Gaussian distribution to approximate the quantization bins according to their offline analysis, then estimate Huffman tree size and encoding size. Their method generates estimates of the quantization bin histogram by compressing a naively-aggregated sampled dataset. Since the naive sampling procedure may cause unexpected discontinuity on the edges of the sampled blocks, the prediction would be inconsistent between the sampled dataset and the original full dataset, leading to large estimation errors. Wang et al. [22] proposed to extrapolate the compression ratios for SZ across error bounds from 10^{-9} to other error bounds, and proposed a ratio estimation method for ZFP by analyzing the weighted average of *BitsPerBitplane* for each block. However, their solution still suffers from very large estimation errors (up to 100% in percent estimation error) as shown in their evaluation results. Jin et al. [23] proposed a quality estimation model (including estimating compression ratio) for prediction-based compressors. Their solution, however, is tailored for the prediction-based compression model and also suffers from higher estimation errors in some datasets than our solution (to be demonstrated later). Liu et al. [24] present a quality estimation method in their proposed compressor, which constrains the estimation on small sampled data blocks.

III. Problem Formulation

We formulate the research problem as follows. Suppose we are given a scientific dataset (denoted by *D*), which contains N data points d_1, d_2, $\cdots$, d_N. We are also given a set of conditions by the user regarding the compression, including the user-specified lossy compressor (e.g., SZ3, SZx, ZFP), user-specified error-control mode (e.g., absolute error bound, relative error bound, peak signal-to-noise ratio (PSNR), and precision), and user-specified error bound (e) which specifies the acceptable point-wise reconstruction error. We denote the compressed dataset as $\hat{D}$ and the decompressed data points (i.e. the data as reconstructed after compression) as $\hat{d}_i$, where i=1,2,3,$\cdots$,N.

We consider the following error-control methods:

- Absolute error bound (ABS): ABS is perhaps the simplest and most widely used error control method. With ABS, the absolute difference between a datapoint, d_i, and its reconstructed counterpart, $\hat{d}_i$, must always respect a maximum threshold e:
$$|d_i - \hat{d}_i| \leq e \tag{1}$$
- Relative error bound (REL): REL (denoted as ϵ in our paper) is used by many applications/use cases in practice.

A relative error bound of $\epsilon \in [0, 1]$ denotes the absolute error bound as a percentage of the data range, that is it corresponds to the absolute error bound e:

$$e = \epsilon \cdot (\max(D) - \min(D)) \tag{2}$$

where $\max(D)$ and $\min(D)$ refer to the largest value and smallest value in the dataset, respectively.

- Peak signal-to-noise ratio (PSNR): PSNR is commonly used by the visualization community to evaluate the quality of the reconstructed data. It is defined:

$$PSNR(D, D') = -20 \log_{10} \frac{RMSE(D,D')}{\max(D)-\min(D)} \tag{3}$$

where *RMSE*(D,D') refers to the root mean squared error (RMSE) between the original dataset and the decompressed dataset. In general, the higher the PSNR, the higher the precision. SZ-series compressors (such as SZ3) support fixed-PSNR mode [25] based on a derived formula that can map the required PSNR to an absolute error bound.

- Precision: Precision is essentially the number of bits to be preserved during the compression, so its value is an integer. The higher the precision the better the quality of the reconstructed data. Precision control is an error-control method offered in ZFP compressor [6] where floats are truncated at the mantissa to compress blocks of data.

Our research objective is to develop a generic, efficient, online framework that can be used to estimate the compression ratio for different error-bounded lossy compressors with diverse error-control modes based on different given datasets. The compression ratio (denoted as R) is defined as the ratio of the original dataset's size to the compressed data size:

$$R = \frac{N \cdot s}{C} \tag{4}$$

where s refers to the number of bytes used per data point in the original raw dataset (e.g., s = 4 for the single-precision floating-point dataset), and C refers to the number of bytes in the compressed dataset. We distinguish between the true compression ratio on the full dataset and our estimated compression ratio obtained using our surrogates as follows: we denote the measured compression ratio using the true compressor and original dataset as R_m and our surrogate-estimated compression ratio on the sampled dataset as R_s. Then, our research target can be written as the following formula:

$$\begin{aligned} &\min \tfrac{|R_m - R_s|}{R_m} \\ &\textit{given a dataset } D, \\ &\qquad \textit{a lossy compressor } Z, \\ &\qquad \textit{an error-control mode } \Gamma. \end{aligned} \tag{5}$$

where Z refers to an error-controlled lossy compressor such as SZ3, SZx, ZFP, and SPERR, and Γ is one of the aforementioned error-control modes (e.g., absolute error bound).

Note that the compression ratio estimation framework would run with an embarrassing parallel mode in nature, meaning that the scalability is not a concern at all. This is because an error-bounded lossy compressor is often running on each rank/processor to compress local data individually in a parallel simulation. In fact, many practical experiments about scientific lossy data compression were conducted in an embarrassing parallel mode. In these works [24], [26], [27], for example, each MPI rank handles the local data compression individually, followed by a parallel data writing to the global file system. Another typical example is data compression used in HDF5 [28], in which each data chunk would be compressed individually by the corresponding H5Z filter [29]. As such, in this paper, we mainly focus on how to efficiently estimate the compression ratio for a given local dataset, as depicted in Formula (5).

IV. Design Overview of SECRE

As shown in Figure 1, the SECRE framework is composed of two key steps: sampling and surrogate. The sampling stage applies a particular sampling method to sample a small portion of the data points from the full dataset. The surrogate stage aims to simulate the compression process on the sampled dataset. We use the simulation to infer the compression ratio given a user-specified error-control mode and error-bound value.

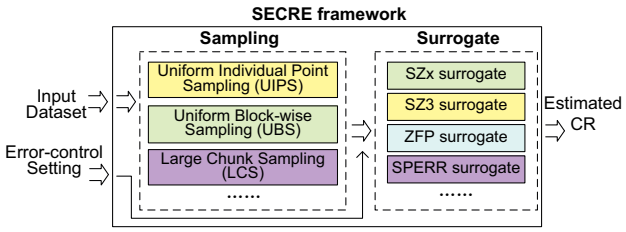


Fig. 1. Design Overview of SECRE Framework

The design of the SECRE framework may appear simple; however, accurately estimating compression ratios is challenging because 1) it is non-trivial to obtain a representative sample given the different data processing methods used by compressors and 2) it is non-trivial to develop accurate surrogate functions for each compressor. We propose the following rules to obtain accurate compression ratio estimates, which will be detailed in the following text.

- *The sampling method must be compatible with the method used by the lossy compressor to process the data.* For example, SZ3 scans and processes the data compression point by point via a multi-level topology, so the corresponding sampling method used by our sampling stage should follow the same topology in order to emulate the compression as closely as possible. Similarly, ZFP performs a multi-dimensional 4^k-block-wise compression, such that we must correspondingly perform a multi-dimensional 4^k-block-wise sampling, where k refers to the number of dimensions.
- *The surrogate method aims to estimate the number of bytes in the compressed dataset by processing the sampled data. The surrogate methods must be both lightweight (in order to produce estimates quickly) and accurate.* In order to produce accurate estimates, it is

important that either the surrogate method closely mimics how the compression is done or that we can derive an algorithm that can essentially replace the compression operations.

V. Compression Ratio Estimation for Different Error-controlled Lossy Compressors

We describe how we implement the compression ratio estimation method for different state-of-the-art lossy compressors following the SECRE framework. Specifically, each subsection focuses on a specific error-controlled lossy compressor. We first describe the best-fit sampling method and then present our designed surrogate function for each compressor.

A. Applying SECRE for SZx

Here, we describe how we develop the compression ratio (CR) estimation method for SZx. In what follows, we first review SZx's compression pipeline and key features, which are the foundations of the design of our estimation method.

In Figure 2, the top subfigure shows the compression pipeline of SZx. SZx splits the whole dataset into many small 1D fixed-length blocks and processes each block individually. For each block, SZx computes a median value (i.e., the mean of minimum value and maximum value in the block), based on which all the blocks can be split into two categories: either constant blocks (i.e., all the data points d_i in the block can be approximated by the median value (denoted as μ) safely according to user-specified error bound ε) or non-constant blocks. Each constant block can be represented by its median value, while the non-constant blocks need to be compressed by IEEE 754 binary format analysis, which involves XOR leading zero byte analysis (to remove redundant bytes), bit-truncation (to remove unnecessary bits based on error bound), and right shift operation to avoid expensive bit-truncation operation. We refer readers to the SZx paper [5] for more details.

The SECRE-based SZx compression ratio estimation is illustrated in the bottom subfigure of Figure 2. The highlighted blue boxes represent our designed surrogate subroutines. As mentioned previously, the basic idea is to emulate the compression procedure over the sampled data with lightweight surrogates, so that we can infer compressed data size.

As presented in the figure, our algorithm performs a uniform block-wise sampling with the same block size as consistent with the SZx compression setting. Specifically, the block size is set to 128 by default in SZx, so we set it to 128 in our implementation. Then, we check each sampled block and calculate the total number of constant blocks, and estimate the compressed data size for non-constant blocks. Finally, the algorithm output is the compression ratio based on Formula (6), which will be derived in detail in the following text.

Corollary 1. *The compression ratio (CR) of SZx can be accurately estimated by the following formula.*

$$R' = \frac{Bs}{M/J+H+(1+(\frac{1}{4}+P)\cdot B-Q)\cdot(1-\lambda)} \tag{6}$$

where s is the number of bytes used per data point in the original raw dataset, B is the block size, M is metadata cost

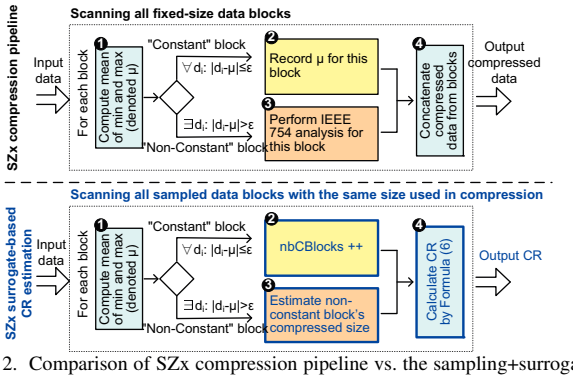


Fig. 2. Comparison of SZx compression pipeline vs. the sampling+surrogate-based CR estimation method (blue boxes indicate the surrogate routines)

of the whole dataset, J is the number of sampled blocks, H is per-block-overhead, P denotes the average number of requested bytes per block, Q is the average number of xor_lead_bytes, and λ denotes the percentage of constant blocks.

Proof. According to the definition of compression ratio as shown in Formula (4), SZx's compression ratio R' can be written as Formula (7).

$$R' = \frac{JBs}{C} \tag{7}$$

where J refers to the number of blocks sampled from the full dataset (i.e., J=$\frac{N}{BK}$ where K is the sampling stride) and C denotes the compressed data size in bytes.

We present how to calculate C as follows. SZx's compressed data is composed of three portions:

- **Metadata (denoted by M)**: It includes version number (3 bytes), random_access_mode marker (1 byte), size of block (1 byte), and the number of constant blocks represented by size_t type (8 bytes). So, M=13.
- **Indispensable storage overhead per block (denoted by H)**: It includes storing the median value (s bytes) and a bit-wise status to represent whether the block is constant or not (1 bit). So, H = s + 1/8.
- **The number of bytes generated by non-constant blocks**. It includes three parts. The first part records the required number of bits calculated based on the user-specified error bound and the value range of the block (1 byte), which can be thought of as the metadata of the block. The second part is a 2bit-per-point state_array to store the number of XOR_lead_zero bytes (2 bits per point, i.e., 1/4 byte per point) for each data point. The third part includes the necessary/significant bits used to represent each data point.

Hence, the compressed data size (in bytes) for the non-constant blocks can be written as:

$$C = M + JH + (1 + \tfrac{1}{4}B + PB - Q)(1-\lambda)J \tag{8}$$

where P denotes the average number of requested bytes per block and Q means the average number of *xor_lead_bytes*.

Combining Equation (7) and Equation (8) leads to Formula (6). □

In terms of the Formula (6), we can estimate the compression ratio for SZx efficiently. We present the pseudocode in Algorithm 1. Line 1 is to sample the data. Line 2-5 is initializing the variables for the compression ratio estimation. Line 8 is counting the number of constant blocks in the sampled data set. Line 10-25 calculates the compressed size for non-constant blocks. The algorithm finally calculates the overall estimated compression ratio in line 26.

Algorithm 1 COMPRESSION RATIO ESTIMATION FOR SZX

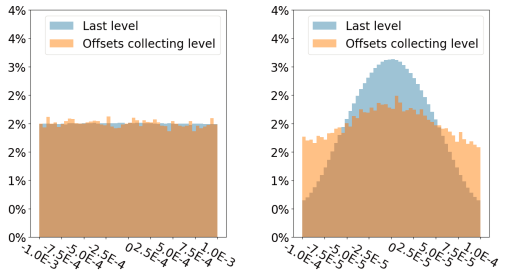
Input: dataset D, user-specified error bound ε
Output: estimated compression ratio R'

1: Block-wise Sampling with stride K; /*K=20 in our implementation*/
2: nbConstantBlocks $\leftarrow$ 0; /*Initialize # constant blocks*/
3: sumReqByteCount $\leftarrow$ 0;
4: metadata $\leftarrow 13B/N$; /*B: blocksize, N: # data points in D*/
5: blockCost $\leftarrow$ s + 1/8; /*e.g., for single-prec float, blockCost=33/8*/
6: **for** each sampled block **do**
7: **if** (radius $\leq \varepsilon$) **then**
8: nbConstantBlocks ++; /*Calculate total # of constant blocks*/
9: **else**
10: Calculate the medianValue and radius for the block;
11: Compute reqBitCount based on radius and medianValue;
12: $\alpha \leftarrow$ reqBitCount/8; /*α: # of requested bytes*/
13: resiBitCount$\leftarrow$reqBitCount%8; /*resiBitCount: # of residual bits*/
14: rightShiftBits $\leftarrow$ 8 - resiBitCount;
15: Adjust α based on rightShiftBits; /*α++ if rightShiftBits$\neq$0*/
16: sumReqByteCount += α;/*Calculate total # of requested bytes*/
17: **for** each data point d_i in the block **do**
18: Calculate β based on d_i XOR d_{i-1}; /*β: # of lead_zero bytes*/
19: Adjust β according to α; /*β must be $\leq \alpha$*/
20: sumXORLead += β; /*Calculate total # of xor_lead_zero bytes*/
21: **end for**
22: **end if**
23: **end for**
24: avgReqNbBytes = sumReqByteCount/nbNonConstantBlocks;
25: avgXORLead = sumXORLead/nbNonConstantBlocks;
26: Calculate R' based on Formula (6);

B. *Applying* SECRE *for SZ3*

SZ3 is a prediction-based lossy compressor that has distinct design principles compared with bit-manipulation-based compressors (such as SZx) and transformation-based compressors (such as ZFP and SPERR). In this subsection, we first introduce the design of SZ3, and then discuss how we estimate the compression ratio for SZ3.

SZ3 is designed based on a typical prediction-based error-bounded lossy compression framework, which has been used by many other compressors such as SZ1.4 [21], SZ2 [27], and FPZIP [30]. SZ3 exhibits much better compression quality than other prediction-based compressors according to prior studies [4], [15], so we focus on SZ3 as the best example of prediction-based compression in this paper. Basically, SZ3 uses the interpolation method (both linear and cubic) to approximate data points. Estimation errors are introduced in the course of a customized linear-scale quantization converting floating-point numbers to integers, which is lossless-compressed by Huffman coding and Zstd [31]. Interpolation proceeds by progressively covering all data points, initially using very large strides to cover the dataset and gradually converging to using each data point's immediate neighbors to estimate that point's value. Each stride value is associated with an "interpolation level" and the processing at each level concludes when the stride increments our current index over the current dimension out of bounds. By performing this algorithm over each dimension, we progressively cover every point in the dataset. Using a set of fixed strides eliminates the need to store location information for the reconstruction procedure, thus saving additional space.



(a) Hurricane (field = U) (b) Miranda (field = velocity-tyx)

Fig. 3. We can see that emulating interpolation with a high stride value can accurately estimate the error distribution for SZ3 as if we had run full compression. We show the distributions obtained by our surrogate and the true distribution for the Hurricane-ISABEL and Miranda datasets.

SECRE is deeply customized with regard to the SZ3 architecture in order to estimate the compression ratio of SZ3 accurately and efficiently. The estimation contains three steps:

- *Step 1 (Zero-cost preprocessing)*. SECRE first locates the data points that belong to the final level of SZ3, since the final level serves as the best representation of the whole dataset. In SZ3, the distance between neighbor data points processed in each level is inversely proportional to the level order. Before the final level, data points with all even indexes are already processed. The remaining data points which count for the majority of the data (75% for 2D datasets, and 87.5% for 3D datasets) will be processed at the final level. As a result, the final level is the ideal location to estimate compression errors via our surrogate to avoid expensive computation.
- *Step 2 (Sampling)*. Sampling is needed to reduce the data volume at the final level, further reducing runtime. Uniform Individual Point Sampling (UIPS, as shown in Figure 1) is applied in this step to trim the data points uniformly in each dimension. The drop-out rate of UIPS is configurable in SECRE and the default value is 5 which means in each dimension, 5 data points will be reduced to 1 (In a 3D case, the dataset will be reduced to 0.8% of its original size). The default value is chosen to balance the sampled size and sampled accuracy.
- *Step 3 (Surrogate)*. In the last step, interpolation is performed to approximate the sampled data. In SZ3, each data point after interpolation is considered a known data point at the following levels, and its value is adjusted as the way it is decompressed, in order to supply the same value to the following levels during compression and decompression. In comparison, SECRE only handles a tiny subset of the final level, such that most of the values are not adjusted. In order to have similar interpolation

results as SZ3, SECRE brings offsets to the sampled data to mimic the adjustment. It is observed in Figure 3 that the distribution of offsets differs between datasets due to diverse data characteristics. As a consequence, SECRE will first collect offsets by executing interpolation on a small level that handles much less data than the last level, and then apply offsets (by sampling uniformly from the gathered distribution) to the sampled data before the interpolation to simulate reconstruction error. Our surrogate currently estimates the compression ratio until the Huffman Encoding step of SZ3, omitting the lossless ZSTD step due to the complexity of modeling ZSTD's behavior - our evaluation similarly compares only the output of the Huffman Encoding step with [23].

C. Applying SECRE for ZFP

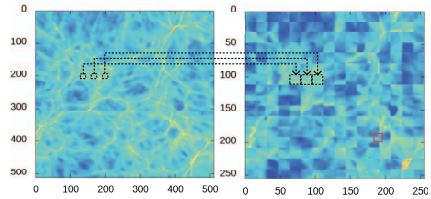
We first review the working principle of ZFP [6], and then describe how we estimate the compression ratio for ZFP.

The basic idea of ZFP is to split the whole dataset into fixed-size blocks (4^k, where k is the number of dimensions), and then perform a set of compression operations on each block independently. The compression operations include four key steps for each block: (1) align the values to have the same exponent; (2) convert the values to a fixed-point representation; (3) apply an orthogonal block transform to make the data easier to encode; (4) embedded coding to significantly reduce the data size.

In the following, we describe our compression ratio estimation method for ZFP. Since each block is processed independently from one another in ZFP, our approach involves two steps: (1) perform a uniform multi-dimensional block-wise sampling where each sampled block has the size of 4^k, which is exactly consistent with the block size used in ZFP. Such a design complies with the first rule we proposed in Section IV, which is crucial to accurately estimate the compression ratio for ZFP. (2) We assemble the sampled blocks to construct a synthetic dataset, on which we perform an essential ZFP compression[1], which will output the estimated compression ratio. Although the assembled dataset may lose smoothness at the edge of the blocks (as illustrated in Figure 4 using Nyx cosmology simulation dataset), this would not affect compression ratio estimation because ZFP's compression is applied on the 4^k-blocks individually, as described above.

It is worth noting that Lu et al. [14] also proposed the use of block-wise sampling to estimate the compression ratio (CR) for ZFP, however, our surrogate-based CR estimation method advances prior work in the following facets. (1) Lu et al.'s CR estimation applies 1D block-wise sampling, which may suffer from large CR estimation errors for multi-dimensional data compression. In comparison, our sampling method is consistent with the multi-dimensional compression of ZFP, so it can maintain the high accuracy of CR estimation, to be demonstrated later. (2) Lu et al.'s CR estimation method supports only absolute error bound, while our SECRE-based method supports different error control modes such as absolute error bound and precision mode, to be shown later.

[1]The essential ZFP compression here means that we perform ZFP's compression on the dataset but ignore the final compressed data aggregation step because our goal is to obtain the estimated compressed data size (or compression ratio).



(a) Original raw data (b) sampled data

Fig. 4. Full dataset vs. block-wise sampled dataset for Nyx [32]

D. Applying SECRE for SPERR

SPERR [16] is a wavelet-transform-based error-bounded lossy compressor that achieves outstanding rate-distortion on a variety of data inputs [33] with relatively low compression speed. The compression pipeline of SPERR is illustrated in the left sub-figure of Figure 5. SPERR first transforms the original input data into wavelet coefficients with CDF9/7 [34] wavelet transform and then applies the lossy SPECK [35] encoding on those coefficients to acquire a lossy compressed representation. Additionally, an inverse wavelet transform has to be performed on the SPECK lossy encoded coefficients, in order to detect and correct out-bounded compression errors, so that the user-specified point-wise error bound can be respected. The out-bounded errors are quantized and the bins for error corrections are stored together with the SPECK-encoded bitstream.

To estimate the SPERR compression ratio on certain input data, SECRE leverages a tiny-scale compression test on the whole input data. As demonstrated in the right sub-figure of Figure 5, Our SECRE-based estimation first uniformly samples several data blocks from the full dataset based on fixed block size and certain sampling percentage/rate (e.g. 1% data are sampled), then performs a surrogate version of SPERR compression on those blocks. In the end, our algorithm calculates the overall estimated compression ratio by aggregating the compression ratio outputted by each sampled block.

VI. PERFORMANCE EVALUATION

We evaluate SECRE on four real-world scientific datasets. We evaluate our solution with respect to (1) the estimation error of the predicted compression ratio computed as the percent error of the estimated compression ratio with respect to the true compression ratio and (2) the run-time overhead of SECRE (including sampling time) compared to the overall compression time. We show that SECRE achieves state-of-the-art estimation performance while outperforming comparable models in terms of run-time efficiency.

A. Experimental Setup

1) System Environment: We conduct our experiments on the Bebop supercomputer at Argonne National Laboratory (ANL), evaluating our performance on a single node equipped

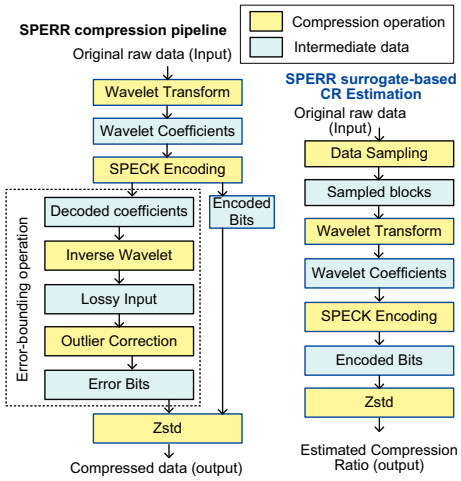


Fig. 5. Comparison of SPERR compression pipeline vs. our approach that relies on sampling and surrogate compression ration estimation

with an Intel Xeon Phi 7230 processor with 64 CPU cores, 96GB DDR4, and 16GB MCDRAM.

2) Compressors used in our evaluation: We evaluate our approach with four cutting-edge error-bounded lossy compressors: SZ3, ZFP, SZx, and SPERR. Each of them features a particular advantage in different use cases. For example, SZ3 and ZFP are two fast error-bounded lossy compressors that can achieve high compression ratios for many scientific datasets. SZx is an ultra-fast error-bounded lossy compressor (generally 4-5X as fast as SZ3/ZFP), with compromised compression ratios to a certain extent. SPERR is an emerging error-bounded compressor taking advantage of the global wavelet transform on the whole dataset, which can achieve extremely high compression ratios in some cases but suffers from slow compression. SZ1 and SZ2 are excluded because their compression quality is always inferior to SZ3 according to prior studies [4], [24].

3) Evaluation Datasets: Our evaluation includes four widely used real-world scientific datasets spanning several domains (such as cosmology and climate research).

- Miranda [36]: A snapshot of the 3D hydrodynamics data obtained from running a large turbulence simulation, which involves 7 fields in total, including pressure, density, etc.
- NYX [32]: 3D mesh datasets generated by a cosmological hydrodynamics simulation based on adaptive mesh. Each snapshot contains 6 fields including dark matter density, baryon density, temperature, velocity x, velocity y, and velocity z.
- Hurricane ISABEL [9]: 48 snapshot datasets each containing 13 fields, each of which is a 3D mesh dataset (100x500x500). This dataset was produced by a weather simulation of Hurricane Isabel.
- CESM-ATM [1]: climate simulation datasets generated by a 1/4 degree high-resolution Community Atmosphere Model (CAM) simulation.

We summarize the four scientific datasets in Table I.

TABLE I
DATASETS USED IN EXPERIMENTS

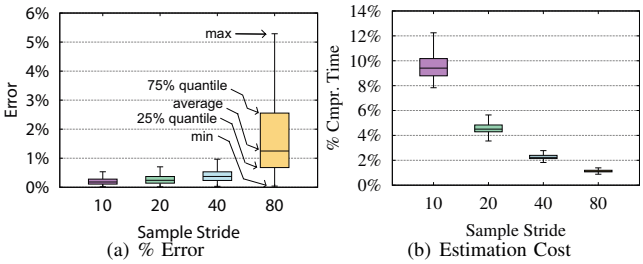
App.	# fields	Dimensions	Total Size	Domain
Miranda	7	256×384×384	1GB	Turbulence
NYX	6	512×512×512	3.1GB	Cosmology
CESM-ATM	77	1800×3600	1.9GB	Climate
Hurricane	48x13	100×500×500	58.1GB	Weather

4) Baselines: In the evaluation, we compare our SECRE with two related works.

- Lu et al.'s model [14]: Lu et al. proposed a compression ratio estimation method for SZ1.4 and ZFP. As for SZ1.4, since their method is strictly dependent on SZ1.4's design principle, it cannot be extended to other versions of the SZ compressor such as SZ2 and SZ3 directly. As such, we compare our approach with the ZFP compressor. Since their released code [37] is missing the compression ratio estimation function for ZFP, we implemented the estimation method according to their paper [14]: specifically, their method samples the data points in a block of four points and then performs ZFP compression. We also implemented a version to support the precision mode for the ZFP compressor based on their method.
- Jin et al.'s model [23]: Jin et al. developed an analytical framework to estimate the compression ratio for the SZ compression model. In their code, they studied how to predict compression ratios based on three predictors: Lorenzo [38], regression [27], and spline interpolation [4], which are adopted by SZ1.4, SZ2, and SZ3, respectively. In our evaluation, we compare our solution with Jin et al's method based on SZ3, because as mentioned previously, SZ3 exhibits similar or better compression quality over SZ1.4 and SZ2 according to prior studies [4].

B. Evaluation on SZx

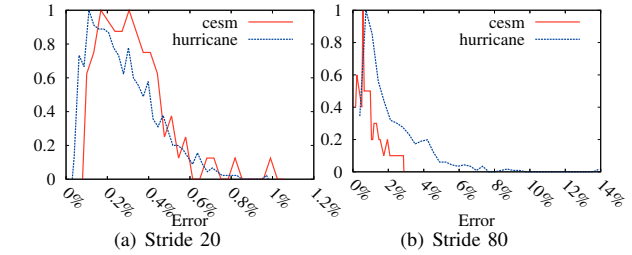
Figure 6 (a) and (b) shows the compression ratio estimation errors (i.e., the percentage of estimation error over the measured compression ratios with the same error bound setting) and the time cost of the estimation (i.e., the percentage of the estimation time compared with the compression time), respectively, based on different sampling strides. Each candle stick shows the max error, 75% quantile error, average error, 25% quantile error, and min error. Figure 6 (a) shows that the estimation errors increase with the sampling stride, which is reasonable because a larger stride means lower portions of data sampled for the analysis and thus lower accuracy in estimation. In comparison, Figure 6 (b) demonstrates that the estimation cost decreases with the sampling stride. We can clearly see that even when adopting a very large sampling stride such as 40 (sampling rate would be 1/41≈2.43% at this stride), the estimation error is still fairly low (less than 1% in the worst case). In this situation, the corresponding estimation cost is also extremely low (∼2%) compared with the compression time. As such, the sampling stride = 40 seems to offer the best tradeoff for SZx.



(a) % Error (b) Estimation Cost

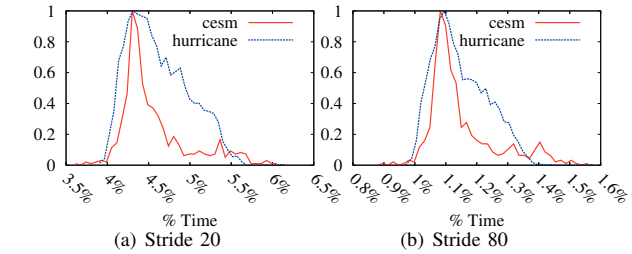
Fig. 6. SECRE Performance for SZx

We also provide the normalized distribution (i.e., probability density function (PDF)) of the estimation error and estimation cost in Figure 7 and Figure 8, respectively, using two datasets (CESM-ATM and Hurricane) each with dozens or hundreds of data fields (files). We do not perform the distribution analysis for Miranda and NYX because there are too few fields for each of them (Miranda has 7 fields and NYX has 6 fields). Combining Figure 7 and 8, we can clearly see that the estimation errors and estimation cost form a trade-off for SZx. Specifically, the estimation error is very low (about 0.2-0.4%) while suffering from relatively high estimation cost when the stride is set to 20 (i.e., 5% data are sampled), compared with the stride of 80, and vice versa.



(a) Stride 20 (b) Stride 80

Fig. 7. SECRE % Error for SZx



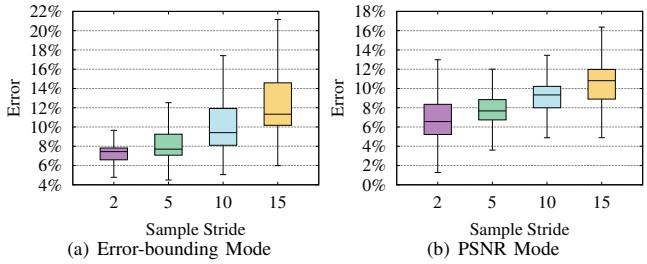
(a) Stride 20 (b) Stride 80

Fig. 8. SECRE Estimation Cost for SZx

The key reason for the high estimation accuracy and low estimation cost with SECRE is twofold. On one hand, the uniform sampling method can significantly reduce the time complexity because of considerably fewer data points involved in the estimation. On the other hand, we carefully design the surrogate method by leveraging a derived estimation formula with only the necessary operations in estimating the compression ratios.

C. Evaluation on SZ3

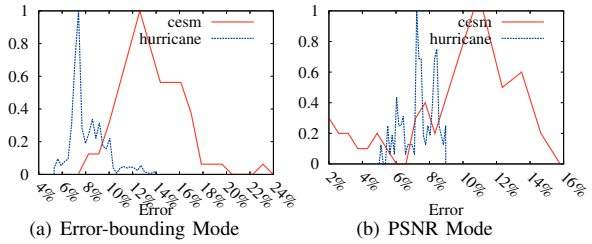
We present the evaluation results of the estimation error and cost on SZ3 in Figure 9 based on two compression modes: error-bounding mode and peak signal-to-noise ratio (PSNR) mode. It is clear that the estimation error increases slightly with the sampling stride because a larger stride means less sampled data to be used. It is worth noting that the average estimation error stays at a low level—around 10% even when adopting a relatively high sampling stride such as 15 (corresponding to a sampling rate of 1/16≈6%)—indicating highly accurate estimation of compression ratios for both compression modes.



(a) Error-bounding Mode (b) PSNR Mode

Fig. 9. SECRE Estimation Error for SZ3 using various sampling strides in (a) error-bounding mode and (b) PSNR mode

Figure 10 shows the normalized distribution (PDF) of estimation error (%) calculated based on all fields/files of the CESM-ATM and Hurricane datasets, under the the sampling stride of 5. We can see that the Hurricane dataset has lower estimation errors than CESM-ATM does in general. This is due to the fact that the 3D Hurricane dataset projects a much lower sampling rate than the 2D CESM-ATM dataset does when they are using the same sampling stride. In fact, their sampling rates are both quite small (0.8% for Hurricane and 3% for CESM-ATM) because our method samples the data only at the lowest interpolation level along different dimensions (see Section V-B for details).

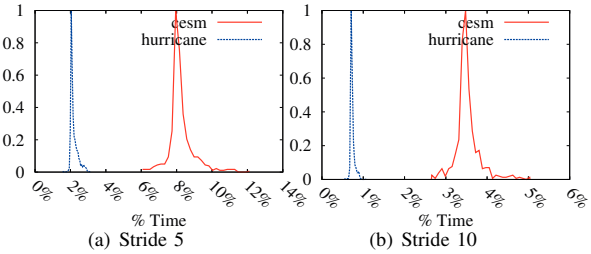


(a) Error-bounding Mode (b) PSNR Mode

Fig. 10. SECRE % Error for SZ3 using sample stride 5

Fig. 11 shows estimation cost based on two sampling strides (5 and 10). We can clearly observe very low estimation costs (about 1-2% for Hurricane and 3-8% for CESM on average).

We also compare SECRE with Jin et al.'s [23] state-of-the-art estimation method in terms of both estimation error (Table II) and estimation time (Table III). We see that SECRE has 14-50% lower average estimation errors for most of the datasets (Miranda, NYX, and Hurricane except for CESM) than Jin



(a) Stride 5 (b) Stride 10

Fig. 11. SECRE Estimation Cost for SZ3

et al.'s method, with a comparable estimation time cost. The higher accuracy in our estimation is because of our efficient sampling method and surrogate for SZ3.

TABLE II
COMPARISON OF % ERROR FOR SZ3 ESTIMATION (STRIDE=5)

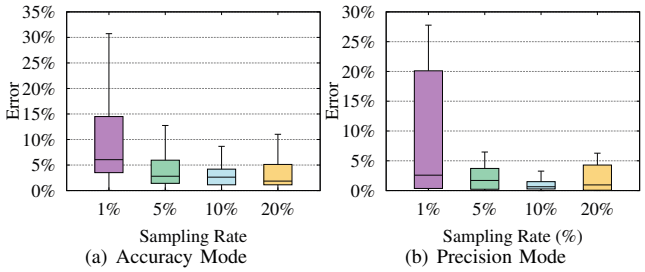
Dataset	SECRE			Jin et al. [23]		
	Min	Avg	Max	Min	Avg	Max
Miranda	4.63%	6.21%	8.44%	12.19%	12.44%	13.04%
NYX	4.49%	6.34%	7.66%	4.18%	7.38%	9.52%
CESM	5.62%	13.23%	31.85%	7.27%	10.97%	20.04%
Hurricane	5.03%	8.03%	13.90%	8.06%	12.56%	17.21%

TABLE III
COMPARISON OF ESTIMATION COST (%) FOR SZ3 ESTIMATION

Dataset	SECRE			Jin et al. [23]		
	Min	Avg	Max	Min	Avg	Max
Miranda	1.48%	1.90%	2.17%	1.31%	1.61%	1.89%
NYX	1.58%	1.71%	2.05%	1.26%	1.37%	1.66%
CESM	5.87%	8.13%	12.11%	6.41%	9.11%	13.80%
Hurricane	1.51%	2.16%	3.12%	1.31%	1.85%	2.66%

D. Evaluation on ZFP

Figure 12 shows the relative error of compression ratio estimation for ZFP based on different sampling rates by aggregating the evaluation results from all four application datasets. Using only 5% of sampled data can achieve fairly high estimation accuracy (with an estimation error of only 2% on average).



(a) Accuracy Mode (b) Precision Mode

Fig. 12. SECRE Average % Estimation Error for ZFP using various sampling rates in (a) accuracy mode and (b) precision mode

We present the min, average, and max of the estimation error and estimation cost of SECRE versus Lu et al.'s method [14] in Table IV and Table V, respectively. The evaluation is conducted using ZFP's precision mode. We see that our estimation method is very accurate in compression ratio estimation (with an average estimation error of less than 1% for Miranda, NYX, and CESM and less than 4% for Hurricane). In comparison, Lu et al.'s method suffers from large estimation errors (¿ 150% on average) for all datasets. The key reason is that the sampling method used in Lu et al.'s method selects a block of four points in only one dimension, which is suitable for 1D compression but unsuitable for multi-dimensional compression in ZFP. In our solution, we adopt a multi-dimensional block-wise sampling, which perfectly matches the multi-dimensional compression in ZFP.

TABLE IV
COMPARISON OF % ERROR FOR ZFP ESTIMATION USING A 5% SAMPLING RATE

Dataset	SECRE			Lu et al. [14]		
	Min	Avg	Max	Min	Avg	Max
Miranda	1.17%	2.50%	3.96%	149.52%	368.93%	594.35%
NYX	0.03%	0.10%	0.18%	149.37%	276.84%	1077.30%
CESM	0.15%	1.47%	5.22%	111.85%	174.68%	253.69%
Hurricane	0.10%	5.81%	37.47%	202.48%	323.95%	423.41%

Table V shows the compression ratio estimation cost of the two solutions. Our solution has a relatively low cost (10-20% in general) compared with the compression time because of the high efficiency in data sampling and surrogate function. Lu et al.'s method [14], however, suffers from much higher cost, mainly because this method performs 1D sampling followed by a 1D compression, which projects more expensive orthogonal transform operations compared with the multi-dimensional transforms adopted by our SECRE method.

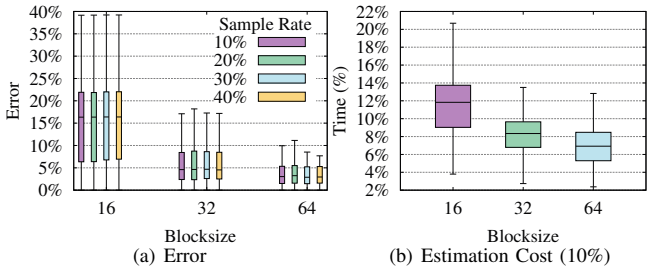
TABLE V
COMPARISON OF ESTIMATION COST (%) FOR ZFP USING A 5% SAMPLING RATE

Dataset	SECRE			Lu et al. [14]		
	Min	Avg	Max	Min	Avg	Max
Miranda	12.26%	23.67%	35.93%	149.52%	368.93%	594.35%
NYX	12.26%	18.65%	59.36%	149.37%	276.84%	1077.30%
CESM	8.24%	10.42%	13.61%	111.85%	174.68%	253.69%
Hurricane	14.61%	20.87%	26.80%	202.48%	323.95%	423.41%

E. Evaluation on SPERR

Our SECRE solution is the first attempt to estimate the compression ratio for SPERR because SPERR is an emerging error-bounded compressor. Therefore, we have no related works to compare regarding SPERR. We evaluate the estimation error and cost for our SECRE method with various block sizes. As mentioned previously, our algorithm samples the dataset based on relatively large block sizes because SPERR's wavelet transform step would be applied to the whole dataset. According to Figure 13 (a), we can observe that the estimation error decreases with the block size, yet it is not affected prominently by the sampling rate for these block sizes. In fact, we also tested the block size of 128, which may lead to fairly large estimation errors (even up to 100× in some datasets). Figure 13 (b) clearly demonstrates that a block size of 64 has the lowest time cost. As such, we conclude that the block size 64 turns out to be the configuration leading to the best estimation performance.

More detailed results can be found in Table VI. It is clearly observed that the average estimation error stays very low: around 2.75%-5.23% for three datasets – Miranda, CESM-ATM, and Hurricane, while the NYX dataset manifests a much larger average estimation error (20.11%). The key reason is



(a) Error (b) Estimation Cost (10%)

Fig. 13. SECRE Performance for SPERR

analyzed as follows. Note that we are using a block size of 64 to perform the estimation. For the three datasets (Miranda, CESM-ATM, and Hurricane) that exhibit high estimation accuracy, their dimension sizes (i.e., the number of elements along each dimension) are relatively large compared with 64. In comparison, the Hurricane dataset (100×500×500) has an irregular dimension size with respect to the block size of 64, so the sampled data points will not be as uniformly dispersed as expected if the block size is set to 64.

TABLE VI
SECRE EST. ERROR FOR SPERR (SAMPLING RATE = 10%, BS=64)

Dataset	Min	Avg	Max
Miranda	2.07%	2.75%	3.63%
NYX	5.21%	20.11%	29.16%
CESM-ATM	0.0%	5.23%	31.89%
Hurricane	0.0%	3.45%	7.04%

VII. Conclusion and Future Work

We proposed SECRE—a novel, generic, lightweight compression ratio estimation framework, based on sampling and surrogate estimators. We implemented compression ratio estimation methods based on SECRE for four state-of-the-art lossy compressors and evaluated our approach by comparing it with two other related works using four real-world datasets - demonstrating that our solution can estimate compression ratios very accurately (e.g., about 1% in estimation error for SZx) with very low estimation costs (e.g., only 2% cost compared with compression time for SZx).

In the future, we plan to develop the estimation methods based on the SECRE framework for many other lossy compressors (such as FPZIP [30], TTHRESH [39] and FAZ [33]).

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, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709 and OAC-2104023. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne) and on Theta and JLSE (operated by Argonne Leadership Computing Facility).

References

[1] J. E. Kay and 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] F. Cappello, S. Di, S. Li, X. Liang, G. M. Ali, D. Tao, C. Yoon Hong, X.-c. Wu, Y. Alexeev, and T. F. Chong, "Use cases of lossy compression for floating-point data in scientific datasets," *International Journal of HPC Applications (IJHPCA)*, vol. 33, pp. 1201–1220, 2019.

[3] 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.

[4] 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)*, 2021, pp. 1643–1654.

[5] X. Yu, S. Di, K. Zhao, jiannan Tian, D. Tao, X. Liang, and F. Cappello, "Szx: an ultra-fast error-bounded lossy compressor for scientific datasets," *arXiv preprint arXiv:2201.13020*, 2022.

[6] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[7] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen, and F. Cappello, "Exploring autoencoder-based error-bounded compression for scientific data," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2021, pp. 294–306.

[8] D. Tao, S. Di, H. Guo, Z. Chen, and F. Cappello, "Z-checker: A framework for assessing lossy compression of scientific data," *The International Journal of High Performance Computing Applications*, vol. 33, no. 2, pp. 285–303, 2019.

[9] Hurricane ISABEL simulation data, http://vis.computer.org/vis2004contest/data.html, 2004, online.

[10] Y. Liu, S. Di, K. Chard, I. Foster, and F. Cappello, "Optimizing scientific data transfer on globus with error-bounded lossy compression," in *43rd IEEE International Conference on Distributed Computing Systems (IEEE ICDCS)*. IEEE, 2023.

[11] S. Li, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Resilient error-bounded lossy compressor for data transfer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21, 2021.

[12] R. Underwood, S. Di, J. C. Calhoun, and F. Cappello, "Fraz: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data," in *IPDPS*. IEEE, 2020, pp. 567–577.

[13] R. Underwood, J. C. Calhoun, S. Di, A. Apon, and F. Cappello, "Optzconfig: Efficient parallel optimization of lossy compression configuration," *IEEE Transactions on Parallel and Distributed Systems*, 2022.

[14] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu *et al.*, "Understanding and modeling lossy compression schemes on hpc scientific data," in *IEEE IPDPS*. IEEE, 2018, pp. 348–357.

[15] X. Liang *et al.*, "SZ3: A modular framework for composing prediction-based error-bounded lossy compressors," https://arxiv.org/abs/2111.02925, 2021, online.

[16] "Sperr," https://github.com/NCAR/SPERR.

[17] L. Davisson, "Rate distortion theory: A mathematical basis for data compression," *IEEE Transactions on Communications*, vol. 20, no. 6, pp. 1202–1202, 1972.

[18] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

[19] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, 1972.

[20] M. Śmieja and J. Tabor, "Entropy approximation in lossy source coding problem," *Entropy*, vol. 17, no. 5, pp. 3400–3418, 2015.

[21] 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.
[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 TPDS*, vol. 31, no. 7, pp. 1621–1635, 2020.
[23] S. Jin, S. Di, J. Tian, S. Byna, D. Tao, and F. Cappello, "Improving prediction-based lossy compression dramatically via ratio-quality modeling," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 2494–2507.
[24] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Dynamic quality metric oriented error bounded lossy compression for scientific datasets," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2022, pp. 892–906.
[25] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Fixed-psnr lossy compression for scientific data," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 314–318.
[26] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 179–189.
[27] 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 *Conference on Big Data*. IEEE, 2018.
[28] HDF5. [Online]. Available: http://www.hdfgroup.org/HDF5
[29] The HDF Group. (2017) H5Z: Filter and Compression Interface. https://support.hdfgroup.org/HDF5/doc1.8/RM/RM_H5Z.html. Online.
[30] P. G. Lindstrom *et al.*, "Fpzip," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2017.
[31] Y. Collet, "Zstandard – real-time data compression algorithm," *http://facebook.github.io/zstd/*, 2015.
[32] NYX simulation, https://amrex-astro.github.io/Nyx, 2019, online.
[33] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "FAZ: A flexible auto-tuned modular error-bounded compression framework for scientific data," in *Proceedings of the nternational Conference on Supercomputing (ICS '23)*, 2023.
[34] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on pure and applied mathematics*, vol. 45, no. 5, pp. 485–560, 1992.
[35] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *IEEE transactions on circuits and systems for video technology*, vol. 14, no. 11, pp. 1219–1235, 2004.
[36] "Miranda application," https://wci.llnl.gov/simulation/computer-codes/miranda.
[37] T. Lu, "Lossy compression study code for paper "understanding and modeling lossy compression schemes on hpc scientific data" published in ipdps2018," https://github.com/taovcu/LossyCompressStudy.
[38] 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.
[39] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "Tthresh: Tensor compression for multidimensional visual data," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 9, pp. 2891–2903, 2019.