Fast Topology-Aware Lossy Data Compression with Full Preservation of Critical Points and Local Order

Alex Fallin*, Nathaniel Gorski†, Tripti Agarwal†, Bei Wang†, Ganesh Gopalakrishnan†, Martin Burtscher*
*Texas State University, USA †University of Utah, USA
Emails: {waf13, burtscher}@txstate.edu, {gorski, beiwang}@sci.utah.edu, {tripti.agarwal, ganesh}@cs.utah.edu

***Abstract*—Many scientific instruments and simulations generate large volumes of floating-point data at high rates that require compression before storage. Typically, only lossy compression algorithms can achieve the compression ratios needed for these workloads. However, most lossy compressors merely provide pointwise error guarantees and do not preserve topological properties of the data, such as the relative ordering of neighboring values. Existing topology-preserving compressors generally preserve only selected critical points and often incur substantial computational overhead. Our Local-Order-Preserving Compressor is the first method to preserve the complete local order of the data—and therefore all critical points—while remaining orders of magnitude faster than prior topology-preserving approaches. It delivers compression ratios that exceed those of lossless compressors and produces bit-for-bit identical output across CPUs and GPUs.**

***Index Terms*—lossy data compression, critical-point preservation, local-order preservation, parallel execution**

I. Introduction

Many scientific workloads and devices generate data volumes that exceed what can be practically managed in terms of both throughput and storage capacity [22]. To address this challenge, two primary compression strategies are used: lossless and lossy. Lossless compression reproduces the original data exactly but often cannot achieve the compression ratios required by large-scale scientific workflows. Lossy compression, in contrast, can attain substantially higher compression ratios by allowing bounded deviations from the data, albeit at the cost of some information loss and imperfect reconstruction.

Topological data analysis (TDA) uses descriptors such as critical points, contour trees [5], and Morse–Smale complexes [11] to characterize, summarize, and analyze complex datasets. Because these topological descriptors depend on the *global* structure of the data, the *local* error guarantees provided by conventional scientific compressors are generally insufficient to preserve them. Consequently, a variety of compression methods and frameworks have been developed to preserve topological features in scalar fields [18, 25, 38, 45], vector fields [26, 44], and tensor fields [17].

Despite their benefits, existing topology-preserving compressors suffer from several limitations. First, they are often computationally expensive, as preserving topological information typically requires substantial additional processing, resulting in runtimes that can be orders of magnitude slower than those of conventional compressors [45]. Second, most methods preserve only a limited subset of topological features, such as the critical points in the contour tree [18, 45]. Third, although many lossy compressors provide error control, they do not always strictly enforce the specified error bound [15].

Our Local-Order-Preserving Compressor (LOPC) addresses these challenges by providing a strict error-bound guarantee, introducing a novel algorithm that preserves the full local order of the data (and therefore all critical points), and leveraging a parallelism-friendly and fast GPU implementation.

While topology, defined here as the relationships among critical points, is inherently a global property, our approach preserves the relative ordering of neighboring data values, which we refer to as *local order*. This local guarantee serves as a foundation for preserving global topological structure: it improves the fidelity of local gradient flows, reduces visual artifacts such as jagged features, and yields more accurate derived quantities.

Critical points of scalar fields (namely local minima, maxima, and saddles) are fundamental topological features determined entirely by the local order of the data. They serve as essential descriptors in visualization and form the building blocks of more complex topological structures, including contour trees and Morse–Smale complexes. Beyond their topological significance, critical points often have direct physical interpretations in a wide range of applications, including from chemistry [4], climate science [24], and physics [39].

Because local order depends only on local relationships among neighboring values, it is a natural target for preservation under lossy compression. Our method preserves the local order exactly and, consequently, preserves all critical points. To the best of our knowledge, no previous lossy compressor provides either guarantee.

Preserving more information inevitably reduces compressibility, which is why LOPC achieves lower compression ratios than some other topology-preserving compressors. Nevertheless, it remains faster even in serial execution. When parallelized, LOPC attains compression speeds that are orders of magnitude higher than those of prior topology-preserving methods while simultaneously preserving local order, all critical points, and a strict error bound—a combination of guarantees that is unique among compressors reported in the literature. Our main contributions are as follows:

- **A local-order-preserving compression algorithm**: We introduce LOPC, the first compression algorithm that guarantees complete preservation of local order while operating in conjunction with a strictly error-bounded lossy compressor.
- **Theoretical guarantees**: We prove that LOPC preserves the locations and types of all critical points, introduces no

spurious critical points, and always terminates despite the lossy nature of the underlying compression.
- **Efficient implementation**: We present the design, optimization, and parallelization strategies underlying our CPU and especially our GPU implementation of LOPC.
- **Comprehensive evaluation**: We compare LOPC to eight state-of-the-art compressors, including three topology-preserving methods, and demonstrate its unique ability to preserve all critical points and local order.

Our C++/OpenMP and CUDA implementations of LOPC are freely available on GitHub [16].

The remainder of this paper is organized as follows. Section II provides the necessary background. Section III reviews related work. Section IV presents the LOPC algorithm. Section V describes the evaluation methodology. Section VI reports and discusses the experimental results. Section VII concludes the paper with a summary.

II. Background

Piecewise-Linear Scalar Fields: LOPC operates on piecewise-linear (PL) scalar functions defined on triangular (2D) or tetrahedral (3D) meshes. We focus on this setting because critical points are well defined for PL functions on such meshes [42]. Let $\mathbb{X}$ be a triangular mesh and let $f : \mathbb{X} \to \mathbb{R}$ be a PL scalar function. The value of f is specified at each vertex v of $\mathbb{X}$ and extended to the remainder of the mesh through linear interpolation. Consider a triangular cell $\sigma \subset \mathbb{X}$ with vertices v_1, v_2, and v_3. Any point $x \in \sigma$ can be expressed uniquely in barycentric coordinates as $x = t_1v_1 + t_2v_2 + t_3v_3$, where $t_i \geq 0$ and $\sum_i t_i = 1$. The function value at x is then defined as $f(x) = \sum_i t_i f(v_i)$. An analogous definition applies to tetrahedral cells in 3D. Although our proposed approach applies to scalar fields defined on arbitrary triangular and tetrahedral meshes, the current LOPC implementation supports only regular 2D and 3D grids. Following prior work [42], these grids are converted into triangular or tetrahedral meshes through a standard subdivision procedure.

Critical Points: The critical points of a PL function $f : \mathbb{X} \to \mathbb{R}$ are defined in terms of the local neighborhood of each vertex. Let $v \in \mathbb{X}$ be a vertex. The *link* of v, denoted $Lk(v)$, consists of all simplices that do not contain v but whose union with v forms a simplex. The *lower link* of v consists of all simplices σ in $Lk(v)$ whose vertices have function values strictly less than the function value at v,

$$Lk^-(v) = \{\sigma \in Lk(v) : f(u) < f(v), \forall \text{ vertex } u \in \sigma\},$$

and the *upper link* is defined analogously as

$$Lk^+(v) = \{\sigma \in Lk(v) : f(u) > f(v), \forall \text{ vertex } u \in \sigma\}.$$

A vertex v is a local minimum if $Lk^-(v) = \emptyset$ and a local maximum if $Lk^+(v) = \emptyset$. If both $Lk^-(v)$ and $Lk^+(v)$ are nonempty and simply connected, then v is a regular (non-critical) point. Otherwise, v is a saddle point. This classification assumes that adjacent vertices have distinct function values. Throughout this work, we enforce this condition using Simulation of Simplicity [12].

III. Related Work

This section reviews the state-of-the-art floating-point compressors with which we compare LOPC. Note that all lossless compressors inherently preserve the complete topology.

A. Lossless Compressors

FPzip [30] is a CPU-based library for both lossy and lossless compression of scientific floating-point data. It exploits coherence in floating-point values to predict input samples, computes the resulting residuals, converts them to an integer representation, and applies a fast entropy encoder. This approach achieves high compression ratios while maintaining fast compression and decompression speeds.

Zstandard [6] (ZSTD) is a parallel CPU compressor based on LZ77 [23], ANS [10], and Huffman coding [19]. Unlike most of the other compressors evaluated in this work, ZSTD is a general-purpose compressor and is not specifically designed for floating-point data.

FPCompress [3] is a CPU/GPU-parallel lossless floating-point compressor designed for smooth scientific datasets. It provides separate variants optimized for compression speed and compression ratio for both single- and double-precision data. The speed-oriented variants employ delta encoding, a transformation from two's-complement to magnitude-sign representation, and leading-bit elimination. For single-precision data, the ratio-oriented variant uses the same delta encoding and magnitude-sign transformation stages, followed by bit shuffling and iterative zero-byte elimination. The double-precision ratio-oriented variant follows a similar pipeline but first applies a finite-context method (FCM) predictor. Although the FCM stage doubles the intermediate data size, this expansion improves downstream compressibility. In the results section, we compare to the GPU implementations of both the speed-optimized and ratio-optimized FPCompress variants.

B. Lossy Compressors Without Topology Preservation

SZ3 [27, 29, 46] is the successor to SZ2 [28], offering improved compression ratios while maintaining similar throughput. It combines Lorenzo prediction [20] with dynamic spline interpolation and applies entropy coding followed by lossless compression (e.g., Huffman coding [19] followed by GZIP [8] or ZSTD [7]) after the lossy stage. SZ3 is a CPU-only compressor and guarantees that the user-specified error bound is never exceeded. We compare to SZ3 both directly and indirectly since it serves as the base compressor in TopoA (described below). Like the SZ family of compressors, LOPC uses quantization as its lossy stage and employs a sequence of lossless transformations to improve compressibility. However, unlike LOPC, SZ compressors do not preserve the topological structure of the original dataset.

PFPL [13] is a high-throughput lossy compressor designed for both CPUs and GPUs. It begins by quantizing the input and converting the resulting bin values to magnitude-sign representation, while storing outliers inline rather than separately to maximize performance. Similar to LOPC, PFPL partitions the input into 16 kB chunks to enable efficient parallel compression. Each chunk is then processed through a sequence of

lossless transformations. First, the data is delta encoded [21] and converted to negabinary representation. Next, it undergoes bit shuffling. Finally, repeated zero-byte elimination is applied to compress the transformed data. Like SZ3 and LOPC, PFPL guarantees that the user-specified error bound is not violated.

C. Topology-Preserving Lossy Compressors

A number of topology-preserving lossy compressors have been developed for scalar field data. The earliest of these, proposed by Soler et al. [38] and referred to as *TopoQZ* throughout this paper, uses a single persistence threshold to preserve only the critical-point pairs whose persistence exceeds that threshold. The decompressed field retains all such high-persistence pairs while removing all other critical points [41]. However, the locations of the preserved critical points may change. In contrast, LOPC preserves all critical points together with their original locations. Conversely, TopoQZ preserves the persistence relationships among critical point pairs, a property that LOPC does not guarantee.

TopoSZ [45] modifies SZ 1.4 [9] to preserve the contour tree [5] of the original data after persistence simplification [41]. It losslessly stores each critical point in the contour tree. For all other points, it computes upper and lower bounds based on the contour tree. It then iteratively tightens these bounds until the contour tree of the decompressed data matches the ground truth. LOPC focuses on preserving all critical points, not just those stored in the contour tree [40, page 22, Property 2.12]. However, it does not ensure that the internal connections of the contour tree are preserved.

Gorski et al. [18] proposed a framework, referred to as *TopoA* in this paper, for augmenting existing lossy compressors with topology-preservation capabilities. Like TopoSZ, it aims to preserve contour-tree structures and follows a similar identify-and-fix strategy. However, TopoA progressively tightens the upper and lower bounds on vertex values, avoiding the need to repeatedly recompute the contour tree, which improves performance. It also introduces a more efficient quantization scheme that yields higher compression ratios. In our evaluation, we compare to the TopoA-augmented version of SZ3, as it provides the highest compression ratios among topology-preserving compressors with guaranteed error bounds.

mSZ [25] is a parallel topology-preserving compressor that focuses specifically on preserving Morse–Smale segmentations. Unlike the contour-tree and critical-point preservation approaches discussed above, Morse–Smale segmentation represents a different class of topological structures. To preserve these structures, mSZ augments both SZ3 and ZFP with an iterative identify-and-fix procedure similar to that employed by LOPC. Beyond compression, topology-preserving dimensionality reduction techniques have also been proposed (e.g., [31]), including approaches based on deep learning [32, 36].

IV. LOPC Algorithm and Implementation

This section describes LOPC's modular approach and how it preserves the critical points and local ordering.

A. Quantization

The first stage of LOPC is lossy quantization, where the error-bounded approximation of the input data is performed. LOPC supports both point-wise absolute (ABS) and point-wise normalized absolute (NOA) error bounds. For an ABS error bound of ε, each reconstructed value x must satisfy

$$|x_{\text{original}} - x_{\text{reconstructed}}| \leq \varepsilon.$$

The NOA error bound is defined as the ABS error normalized by the data range $R = x_{\max} - x_{\min}$, where $x_{\max}$ and $x_{\min}$ denote the maximum and minimum values in the dataset, respectively. NOA is commonly used in topology-preserving compression.

Using the supplied error bound ε, the quantizer maps each floating-point value to a quantization bin. This is accomplished by multiplying the value by $1/\varepsilon$ and rounding the result to the nearest integer, yielding the corresponding bin number. In conventional ABS quantization, bins have width 2ε and values are reconstructed to the bin center, guaranteeing a reconstruction error of at most ε. In LOPC, however, the bin width is reduced by a factor of two to accommodate the subsequent intra-bin adjustments used to preserve local ordering while still satisfying the prescribed error bound.

In addition to the quantization bins, LOPC introduces a set of *subbins* that allow reconstructed values to be shifted within their assigned bin without violating the error bound. For example, with an ABS error bound of 0.1, all values in the interval $[0.95, 1.05)$ are mapped to bin 10. Without subbins, every value in this interval would be reconstructed to 1.0. With subbins, values can instead be reconstructed to carefully selected locations within the interval $[0.95, 1.05)$, while remaining within the prescribed error bound. This additional degree of freedom enables LOPC to preserve the locations and types of critical points, as described next.

B. Preserving Critical Points and Local Order

Rather than explicitly preserving critical points, LOPC iteratively corrects all values that violate the less-than relationship (i.e., local order) with neighboring values assigned to the same quantization bin. Restricting corrections to values within the same bin is sufficient because the relative ordering of values mapped to different bins is automatically preserved by the *monotonic increasing* property of the quantization function. Notably, preserving local order is a stronger condition than preserving critical points as it guarantees not only their existence but also the preservation of their exact types and locations. LOPC achieves this through an iterative procedure. Algorithm 1 presents the overall workflow, while Algorithm 2 details the operations performed during a single iteration of the local-order preservation step.

Initially, all subbin values are set to zero and the original data is quantized into bins. LOPC then computes a set of flags for each point p. For every neighbor of p, the flags record whether the neighbor belongs to the same bin (i.e., has the same bin value). If so, they additionally record whether the neighbor's original value is less than that of p. A deterministic tiebreaker ensures that any pair of neighboring points is

Algorithm 1 Main LOPC Operation

```
1: for each point p do                                          ▷ parallel loop
2:     p_bin ← ABS or NOA quantized bin number of p_val;
3:     p_subbin ← 0;
4:     p_flags ← 0;
5: for each point p do                                          ▷ parallel loop
6:     for each neighbor n of p do
7:         p_flags ← p_flags ∪ (n_bin = p_bin);
8:         p_flags ← p_flags ∪ (n < p);                        ▷ with tie breaker
9: worklist1 ← all input points;
10: while worklist1 is not empty do
11:     worklist2 ← empty;
12:     Algorithm 2;
13:     swap(worklist1, worklist2);
```

Algorithm 2 Parallel Subbin Computation (one iteration)

```
1: for each point p in worklist1 do                             ▷ parallel loop
2:     n_max ← 0;
3:     for each same-bin neighbor n of p do                     ▷ using p_flags
4:         if n should be less than p then                      ▷ using p_flags
5:             tie ← (n_idx > p_idx);                           ▷ tie breaker: 0 or 1
6:             val ← atomicRead(n_subbin);
7:             n_max ← max(n_max, val + tie);
8:     if atomicMax(p_subbin, n_max) < n_max then
9:         worklist2 ← worklist2 ∪ {p's greater same-bin neighbors};   ▷ using p_flags
```

assigned a strict ordering, so that one is always considered greater than the other, never equal. These flags encode the ground-truth local ordering that must be preserved in the reconstructed data.

LOPC then enters an iterative refinement process. In each iteration, every point p examines its same-bin neighbors n that are required to be less than p, as determined by the flags. If all such neighbors already satisfy this relationship, no action is taken. Otherwise, the subbin value of p is increased to match the largest subbin value among the violating neighbors, or to exceed it by one when the tiebreaker favors the neighbor. Because subbin values are only increased and never decreased, each update makes progress toward satisfying the required local ordering. The process repeats until an iteration produces no changes, at which point it terminates.

Upon termination, the subbin assignments are the smallest values that satisfy all required local-order constraints, ensuring that the reconstructed data preserve the same less-than relationships as the original data. In practice, most subbin values remain small integers close to zero, which is beneficial for compression.

As illustrated in Algorithm 1, preserving the critical points and local order requires several auxiliary data structures, resulting in storage overhead. If n is the number of floating-point values in the input and w the targeted word size in bytes (i.e., 4 or 8), then the LOPC encoder requires up to $2nw + 14n + 8$ bytes of auxiliary memory. The first term reflects the bins and subbins. The second term accounts for the two work lists, the timestamp array, and the flag array, all of which are not dependent on the word size. The final term is for the two worklist-size variables.

C. Compression

The above procedure converts each input value into two components: a bin number and a subbin number, effectively doubling the size of the original dataset. The result is two arrays that encode distinct types of information. The first array contains the quantized bin numbers, which are sufficient to reconstruct the original values within the specified error bound. The second array contains the subbin numbers, which encode the information needed to preserve the original local ordering.

The information content of these two arrays depends strongly on the chosen error bound. When the error bound is small, neighboring values are likely to be assigned to different bins. In this case, the bin array retains most of the information from the original dataset, while the subbin array contains predominantly zeros because few local-order corrections are required. Conversely, when the error bound is large, neighboring values are more likely to be assigned to the same bin. The bin array then carries relatively little information, whereas the subbin array captures much of the local-order structure of the original data.

To boost the compression ratio, LOPC compresses the bin and subbin arrays separately using algorithms tailored to their respective characteristics. Bin numbers are computed using SLEEK's quantizer [1], which guarantees compliance with the prescribed error bound without requiring special treatment of outliers. The resulting bin array is then compressed using the lossless stage of PFPL [13, 14]. Specifically, PFPL computes a negabinary difference sequence, performs bit transposition, and then eliminates zero bytes as described below. Both SLEEK and PFPL are efficient and effective and provide compatible implementations for CPUs and GPUs.

To create efficient CPU- and GPU-parallel lossless compressors and decompressors for the subbin data, we used the LC framework [2]. For 32-bit subbins (single-precision data), LC generates the three-stage algorithm BIT_4 RZE_4 RZE_1. For 64-bit subbins (double-precision data), it generates the three-stage algorithm BIT_8 RZE_8 RZE_1. In both cases, the first two stages operate on words matching the data type size, while the final stage operates at byte granularity. The BIT stages, illustrated in Figure 1, perform a bit transposition (or bit shuffle). Specifically, they group together the first bit of every value, then the second bit of every value, and so on. The RZE (Repeated Zero Elimination) stages, outlined in Figure 2, generate a bitmap in which each bit corresponds to an input word and indicates whether that word is zero. All zero-valued words are then removed. The compressed output consists of the remaining nonzero words and the bitmap. The bitmap is subsequently compressed using a similar algorithm that identifies repeated words rather than zero-valued words. Additional details on these stages are available elsewhere [3].

Adding further stages to these compression pipelines reduces throughput with little gain in compression ratio. Since the lossless compression component is modular and independent of the critical-point preservation mechanism, alternative

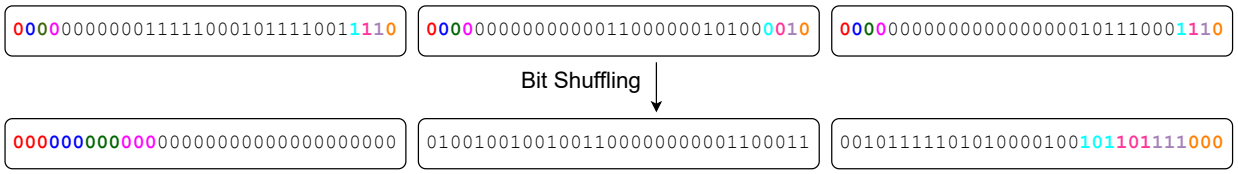

Fig. 1. Example of the bit-shuffling lossless stage. As the input size increases, contiguous sequences of bits with the same color become longer.

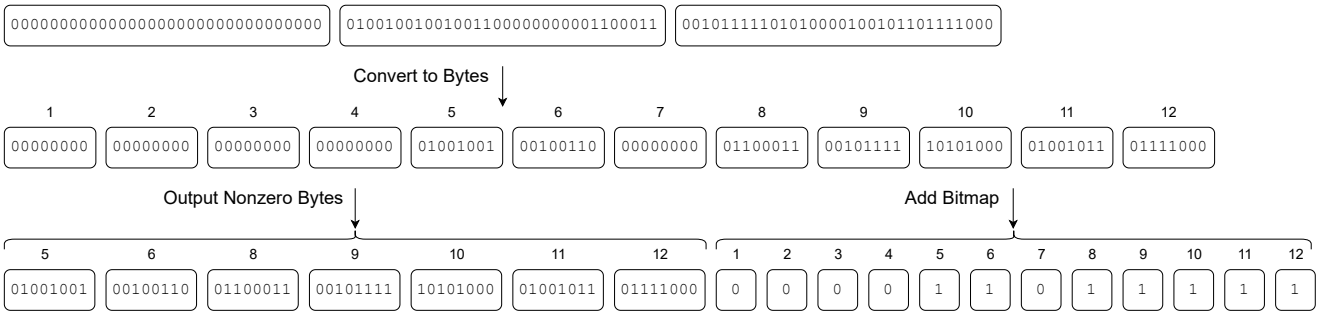

Fig. 2. Example of the zero-byte elimination lossless stage. Further compression of the bitmap is not shown.

compressors can be used. However, such alternatives may yield lower throughput or compression ratios because the LC framework optimized the selected pipelines to the characteristics of the bin and subbin values.

D. Parallelization and Optimization

LOPC leverages the existing parallel CPU and GPU implementations from PFPL and SLEEK to compute, compress, and decompress the bin information. For the subbin data, it employs CPU- and GPU-parallel lossless compressors and decompressors generated by the LC framework.

Subbin decoding is embarrassingly parallel because each bin/subbin pair can be processed independently to reconstruct its corresponding floating-point value. Subbin encoding is more involved. The flag computation (Section IV-B) and subbin initialization are both embarrassingly parallel. The iterative phase that raises subbin values is parallelized by executing each iteration within a separate barrier interval and updating subbins via atomicMax operations, resulting in a lock-free implementation. On the GPU, each point is assigned to a separate thread. On the CPU, points are distributed to threads using a blocked assignment strategy.

Because most points require little or no adjustment, many subbins converge to their final values within the first few iterations. Consequently, processing every point in later iterations is often inefficient. To address this issue, LOPC maintains a worklist containing only the greater neighbors of points whose subbin values were increased in the current iteration. As a result, only potentially affected points are processed in the next iteration. LOPC uses two worklists: one for reading and one for writing. At the end of each iteration, it resets the size of the old worklist and swaps the pointers and size variables of the two worklists. New entries are added using an atomicAdd operation, which both increments the worklist size and returns a unique insertion position. To prevent duplicate entries, each point stores the most recent iteration in which it was added to the worklist; this value is updated atomically whenever a point is inserted.

E. Correctness and Termination

This subsection explains how the LOPC algorithm preserves local ordering, including all critical points, while guaranteeing termination and error boundedness. Since our quantization is an increasing function, the local order is always correct between neighbors that are quantized to distinct bin numbers. Hence, in the rest of the explanation, we only consider same-bin neighbors.

Initially, all subbin numbers are zero. In each iteration, the algorithm examines, for every point, all neighbors that belong to the same bin and whose value should be lower. If a point fails the less-than relationship with any of these neighbors, its subbin number is raised according to one of two rules. (1) The subbin number is raised to match that of the highest violating neighbor if that neighbor's index is lower. (2) The subbin number is raised one higher than that of the highest violating neighbor if that neighbor's index is higher. These two rules ensure that the local ordering among neighboring points gradually becomes consistent (even in the presence of ties) with the local order of the original data.

The algorithm terminates when no violation remains, i.e., all local relationships are satisfied, thereby fully preserving the local order. Since preserving local order implies preserving local maxima, minima, and saddle points, this in turn guarantees that all critical points and their types are preserved as well.

LOPC is guaranteed to terminate because the update process is non-decreasing and bounded. In each iteration, if a violation occurs, at least one subbin number increases. This monotonicity ensures progress and prevents oscillations (i.e., livelock). Moreover, the highest number a subbin can reach is finite. To see why, it helps to consider the connected component (CC) of same-bin values to which a point belongs. (1) Each such CC must contain at least one local minimum (due to the tie

breaker), and the subbin number of a local minimum is never raised because it has no lower same-bin neighbors. (2) No point ever raises its subbin number to more than one higher than its highest same-bin neighbor. (3) The targeted less-than relationships are necessarily acyclic since they stem from the original input data. Together, these properties limit the possible range of subbin numbers in a CC with n points to between 0 and $n-1$. Hence, after a finite number of updates, all violations disappear and the process terminates.

In the worst case, when all n points of a CC form an increasing chain, the algorithm converges after $O(n^2)$ iterations since we can create the values 0 through $n-1$ with $n \times (n-1)/2 = O(n^2)$ individual increments. This worst-case behavior is a tradeoff of the simple approach that LOPC employs. Improving it is a potential avenue for future work.

The only remaining concern is whether the subbin range provides enough resolution to represent all necessary ordering distinctions while remaining within the quantization bin (i.e., no overflow), thereby guaranteeing error boundedness. This is also guaranteed because the maximum number of subbin levels required to preserve local order within any CC is directly tied to the number of distinct floating-point values present in this CC in the original data. Note that decompression maps each value within its quantization bin such that subbin 0 decodes to the lowest representable value within the bin, subbin 1 to the next lowest, and so on. Thus, the algorithm guarantees the local order without running out of subbin range and without violating the user-provided error bound.

V. Experimental Methodology

We compare LOPC to the compressors described in Section III on the two systems listed in Table I. We ran the GPU codes on System 1 and the CPU codes on System 2.

We compiled the CPU codes using the build processes supplied by their respective authors. When not specified, we used the "-O3 -march=native" flags. Unless automatically determined, the thread count was set to the number of CPU cores as hyperthreading usually does not help. We compiled the GPU codes using "-O3 -arch=sm_89" for the RTX 4090.

For all compressors, we measured the execution time of the compression and decompression functions, excluding reading the input file, verifying the results, and transferring data to and from the device. We ran each experiment 9 times and collected the compression ratio, median compression throughput, and median decompression throughput as well as the critical-point false positives, false negatives, and false types. For all compressors, we used NOA error bounds of 1E-2 and 1E-4. For TopoA augmented SZ3, we ran two experiments, one where the persistence threshold ϵ is $1.5\times$ and another where it is $0.5\times$ the NOA error bound. These settings reflect a normal level of persistence and an over-preserving level, respectively. If a compressor runs for more than an hour (wall-clock time), we report 'TO' (timeout). If a compressor crashes, fails, or runs out of memory, we report 'DNF' (did not finish).

We used the 2 single- and 6 double-precision datasets listed in Table II as inputs for the compressors. The Earthquake dataset is from the TeraShake 2 earthquake simulation [33, 34].

TABLE I
Systems used for experiments

	System 1	System 2
CPU	Threadripper 2950X	Threadripper 3970X
Base clock	3.5 GHz	3.7 GHz
Sockets	1	1
Cores per socket	16	32
Threads per core	2	2
Main memory	64 GB	256 GB
GPU	RTX 4090	N/A
Compute capability	8.9	N/A
Base clock	2.2 GHz	N/A
Boost clock	2.5 GHz	N/A
SMs	128	N/A
CUDA cores per SM	128	N/A
Main memory	24 GB HBM2e	N/A
Operating system	Fedora 37	Ubuntu 24.04.1 LTS
`g++` version	12.2.1	13.3.0
`nvcc` version	12.0	N/A
GPU driver	525.85	N/A

TABLE II
Information about the used inputs

Name	Description	Format	Dimensions	Size (MB)
Isabel	Weather Sim.	Single	$90 \times 500 \times 500$	90
Tangaroa	Weather Sim.	Single	$300 \times 180 \times 120$	26
Earthquake	TeraShake 2 Sim.	Double	$375 \times 188 \times 50$	28
Ionization	Ionization Sim.	Double	$310 \times 128 \times 128$	41
Miranda	Hydrodynamics	Double	$384 \times 384 \times 256$	302
S3D	Weather Sim.	Double	$500 \times 500 \times 500$	1000
SCALE-LETKF	Weather Sim.	Double	$1200 \times 1200 \times 98$	1129
QMCPACK	Quantum MC	Double	$69 \times 69 \times 115$	4

The Ionization dataset is timestep 125 from cluster 2 of an ionization front simulation [43]. The Tangaroa dataset is the wind velocity field from a simulation of the Tangaroa research vessel [35]. The remaining inputs are sourced from the SDRBench repository [37, 47], which hosts real-world scientific datasets from various domains for compression evaluation. The table lists the input name, a short description, the data type, input dimensions, and file size. We chose these inputs because they are commonly used as benchmarks in topology work [18, 45] and represent the kind of scientific data that may have important topological information to preserve.

VI. Results

In this section, we evaluate the performance of the compressors discussed in Section III on the inputs described in Section V. We first discuss the preservation of critical points and local ordering for the tested compressors. Next, we compare the compression ratios achieved by each compressor using NOA error bounds. Then, we analyze the compression and decompression speed. Next, we study the effect of error bounding on the compression ratio and speed. Finally, we evaluate the reconstruction quality yielded by the tested compressors.

A. Critical-Point and Local-Order Preservation

Table III shows the quality at which the critical points are preserved. The key strength of LOPC is immediately obvious: it preserves all critical points and local ordering whereas none of the other compressors do, not even the topology-preserving compressors. While TopoA and TopoSZ preserve the critical

TABLE III
COUNT OF FALSE POSITIVES, FALSE NEGATIVES, AND FALSE TYPES ('M' = MILLION, 'K' = THOUSAND).

	LOPC	TopoA SZ3 ϵ = 1.5x EB	TopoA SZ3 ϵ = 0.5x EB	TopoSZ	TopoQZ	SZ3	PFPL
Error bound = 1E-2							
Isabel	0/0/0	686K/16K/2K	734K/13K/3K	2M/16K/3K	461K/20K/639	28K/22K/493	430K/22K/314
Tangaroa	0/0/0	380K/2K/565	514K/2K/500	DNF	57K/4K/236	15K/5K/711	35K/7K/421
Earthquake	0/0/0	218K/95K/8K	180K/83K/8K	448K/95K/11K	22K/100K/1K	25K/110K/2K	10K/112K/398
Ionization	0/0/0	95K/12K/1K	110K/10K/1K	DNF	13K/15K/753	23K/18K/2K	16K/20K/1K
Miranda	0/0/0	1M/12M/214K	TO	644K/13M/255	566/13M/3	81K/13M/9K	2K/13M/2
S3D	0/0/0	7M/23K/5K	TO	133K/50K/218	2M/26K/2K	83K/40K/4K	2M/46K/2K
SCALE	0/0/0	4M/216K/36K	TO	8/315K/0	456K/258K/5K	268K/303K/6K	2M/310K/2K
QMCPACK	0/0/0	22K/168/34	15K/163/33	85K/313/78	11K/167/36	2K/575/16	6K/584/5
Error bound = 1E-4							
Isabel	0/0/0	3K/2K/162	3K/1K/132	TO	5K/4K/103	8K/5K/557	14K/8K/305
Tangaroa	0/0/0	12K/135/9	14K/128/8	70K/203/22	8K/292/15	7K/356/30	17K/540/41
Earthquake	0/0/0	183K/36K/9K	TO	199K/43K/11K	50K/41K/4K	196K/52K/13K	78K/60K/6K
Ionization	0/0/0	87K/4K/1K	118K/4K/1K	458K/4K/1K	10K/7K/814	53K/6K/1K	10K/8K/864
Miranda	0/0/0	624K/13M/74K	TO	1M/12M/779K	7/13M/0	215K/13M/22K	25/13M/0
S3D	0/0/0	27K/2K/204	TO	TO	53K/5K/295	7K/5K/409	252K/10K/715
SCALE	0/0/0	0/0/0	TO	TO	84K/129K/957	219K/155K/9K	210K/187K/3K
QMCPACK	0/0/0	122/84/16	120/84/16	268/138/25	316/119/34	135/159/25	1K/341/35

points on the contour tree, and TopoQZ preserves the critical point pairs, our results show the generally high number of false positives, false negatives, and false types that the other topology-preserving compressors introduce.

Comparing the other topology-preserving compressors to the non-topology-preserving lossy compressors makes it clear that a large portion of the critical points are missed when only preserving the contour tree or critical point pairs. LOPC, which is specialized for exactly this purpose, avoids introducing any false positives, false negatives, or false types by preserving the full local ordering.

The results in the table further demonstrate that it is not straightforward for other topology-preserving compressors to preserve more critical points by lowering the persistence threshold. In many cases, doing so actually introduces more erroneous critical points in a given category. Additionally, lowering the persistence threshold introduces significant additional runtime, leading to timeouts. LOPC avoids these issues without the need for a tunable parameter.

In Figure 3, we visually compare the critical-point preservation of LOPC to SZ3 and TopoA augmenting SZ3 on the QMCPACK dataset. The figure shows that SZ3 and TopoA introduce numerous false critical points whereas LOPC perfectly preserves all critical points.

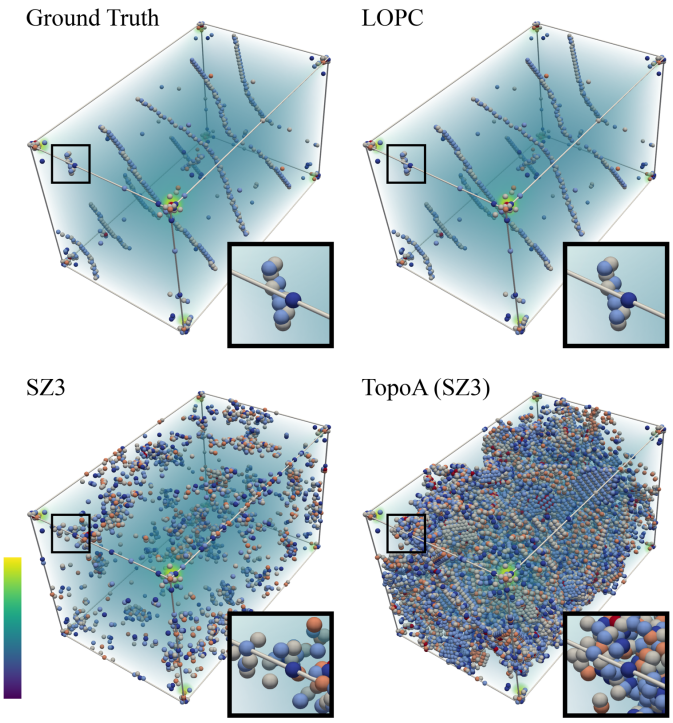

Fig. 3. Visualization of the QMCPACK dataset along with its critical points for the ground truth as well as after compression with LOPC (ours), SZ3, and TopoA augmenting SZ3. We also provide a select zoomed-in view. Both compressors used an error bound of 1e-2 and TopoA used $\varepsilon = 0.04$. Dark blue: minima. Light blue: 1-saddles. Gray: 2-saddles. Orange: Maxima. Red: Degenerate critical points.

B. Compression Ratios

The compression ratios of the evaluated compressors across the two tested error bounds are shown in the first section of Tables IV, V, VI, and VII. Due to its CPU/GPU parity, LOPC delivers the same compression ratio on both devices.

Compared to the other compressors that maintain local order information, namely the lossless compressors (FPZip, ZSTD, and FPCompress), LOPC compresses more in all but one case. On the Ionization input with an error bound of 1E-4, LOPC's compression ratio is a little lower than that of ZSTD. In all other cases, its compression ratio is higher, on average by a factor of 3.7 and, in one case, by over a factor of 8.7.

In general, LOPC yields lower compression ratios than the other topology preserving compressors and significantly lower compression ratios than the lossy non-topology preserving compressors. This is expected for the following three reasons.

First, because LOPC preserves all local ordering, it often preserves many relationships that the other topology preserving compressors do not (see Section VI-A). Of course, it preserves even more relationships than the lossy non-topology-preserving compressors. This large amount of extra information is the main reason for the lower compression ratio. For this reason, LOPC is only recommended when the preservation of local order or critical points is important. If it is not, a non-topology-preserving compressor would be a better choice.

Second, by storing bins and subbins separately, the information density of each part of the compressed file changes with

TABLE IV
COMPARISON WITH TOPOLOGY-PRESERVING COMPRESSORS FOR THE 1E-2 NOA ERROR BOUND

	LOPC			TopoA SZ3			
	Ser	OMP	CUDA	ϵ = 1.5x EB	ϵ = 0.5x EB	TopoSZ	TopoQZ
Compression Ratio							
Isabel	5.47	5.47	5.47	**64.40**	28.43	27.50	4.67
Tangaroa	4.39	4.39	4.39	**28.36**	20.91	DNF	3.33
Earthquake	8.79	8.79	8.79	**85.44**	25.09	65.85	7.08
Ionization	10.19	10.19	10.19	**93.45**	62.69	DNF	5.78
Miranda	10.05	10.05	10.05	**239.09**	TO	92.01	9.14
S3D	9.48	9.48	9.48	35.90	TO	**11,336.58**	5.82
SCALE	10.74	10.74	10.74	37.16	TO	**401,765.12**	5.82
QMCPACK	9.07	9.07	9.07	**102.04**	82.48	12.01	9.08
Geomean	8.18	8.18	8.18	68.32	37.80	**457.04**	6.04
Compression Throughput (MB/s)							
Isabel	9	17	**750**	1	1	0	14
Tangaroa	1	1	**104**	1	1	DNF	14
Earthquake	7	13	**940**	2	1	1	21
Ionization	3	4	**313**	1	1	DNF	11
Miranda	5	8	**414**	0	TO	1	8
S3D	TO	9	**408**	2	TO	2	24
SCALE	TO	TO	34	1	TO	2	**34**
QMCPACK	12	16	**730**	1	0	0	13
Geomean	5	8	**314**	1	1	0	16
Decompression Throughput (MB/s)							
Isabel	310	2,601	**28,754**	13	13	281	4
Tangaroa	316	2,592	**19,059**	11	11	DNF	4
Earthquake	441	1,896	**28,200**	15	14	564	8
Ionization	339	3,628	**33,305**	13	13	DNF	9
Miranda	422	4,314	**30,199**	10	TO	702	8
S3D	TO	3,448	**123,457**	14	TO	446	5
SCALE	TO	TO	**112,896**	14	TO	649	5
QMCPACK	429	4,380	**10,950**	13	13	438	9
Geomean	372	3,142	**35,229**	13	13	492	6

TABLE V
COMPARISON WITH TOPOLOGY-PRESERVING COMPRESSORS FOR THE 1E-4 NOA ERROR BOUND

	LOPC			TopoA SZ3			
	Ser	OMP	CUDA	ϵ = 1.5x EB	ϵ = 0.5x EB	TopoSZ	TopoQZ
Compression Ratio							
Isabel	4.60	4.60	4.60	**11.92**	11.51	TO	2.65
Tangaroa	4.13	4.13	4.13	**15.49**	15.33	10.52	2.35
Earthquake	7.73	7.73	7.73	**14.51**	TO	6.82	4.85
Ionization	8.36	8.36	8.36	**33.31**	30.78	13.78	4.46
Miranda	8.81	8.81	8.81	**47.85**	TO	24.63	4.56
S3D	9.11	9.11	9.11	**51.49**	TO	TO	4.56
SCALE	**9.53**	**9.53**	**9.53**	2.53	TO	TO	3.98
QMCPACK	8.23	8.23	8.23	**39.60**	39.59	11.07	4.66
Geomean	7.26	7.26	7.26	19.63	**21.53**	12.19	3.89
Compression Throughput (MB/s)							
Isabel	79	8,182	**8,411**	1	1	TO	12
Tangaroa	46	199	**2,592**	1	1	0	13
Earthquake	83	470	**7,050**	2	TO	0	21
Ionization	48	102	**4,063**	2	1	0	11
Miranda	7	13	**643**	1	TO	1	8
S3D	8	568	**21,044**	3	TO	TO	22
SCALE	TO	3	**179**	1	TO	TO	31
QMCPACK	146	438	**4,380**	2	2	1	13
Geomean	38	173	**3,003**	1	1	0	15
Decompression Throughput (MB/s)							
Isabel	307	2,894	**31,142**	13	13	TO	2
Tangaroa	324	2,057	**27,284**	11	11	259	4
Earthquake	470	1,986	**21,793**	15	TO	403	6
Ionization	406	4,515	**29,921**	12	12	508	8
Miranda	425	3,355	**33,780**	10	TO	559	7
S3D	383	3,448	**50,618**	15	TO	TO	4
SCALE	TO	3,421	**112,896**	8	TO	TO	3
QMCPACK	438	4,380	**10,950**	13	12	438	8
Geomean	389	3,132	**32,253**	12	12	419	5

the user-requested error bound. For example, if the user selects a strict error bound, most of the topological information ends up in the quantization bins, making them more challenging to compress well, while the subbins contain almost no information. The reverse is true for a loose error bound because the topological information must now be stored in the subbins. This makes it difficult for a single compression algorithm to be effective in all cases due to the greatly changing information density. We discuss this relationship more in Section VI-D.

Third, LOPC is designed to work well on GPUs. As a consequence, its compression algorithm cannot exploit some of the effective but hard-to-parallelize transformations that the CPU-only compressors include. This relationship is evident when comparing the CPU/GPU-compatible compressors PFPL and LOPC to the CPU-only compressors SZ3 and TopoA SZ3. Whereas the CPU-only compression ratios are higher relative to those of the cross-compatible compressors, their throughput is much lower as shown in Section VI-C.

C. Compression and Decompression Speed

The speeds at which the evaluated compressors compress and decompress our inputs are listed in the middle and last section of Tables IV, V, VI, and VII, respectively. The speed of the topology-preserving compressors is very input dependent. For example, even for LOPC, the serial and OpenMP versions time out on the SCALE input. Additionally, for the other topology-preserving compressors, the persistence threshold ϵ has a large impact on the runtime. TopoA augmented SZ3 with a persistence threshold less than the error bound times out on 3 of the 8 inputs for the 1E-2 error bound and on 4 of the 8 inputs for the 1E-4 error bound. These results highlight the need for fast (parallelized) topology preserving compressors like LOPC. We further discuss the effect of the error bound and persistence threshold on performance in Section VI-D.

On the SCALE input at an error bound of 1E-4, only LOPC and TopoA with a large persistence threshold are able to finish within an hour. In this case, however, TopoA takes 22 minutes whereas LOPC only takes 6.3 seconds on the GPU, making it 209.9 times faster than TopoA. Further, TopoA is only able to compress this input using the aforementioned large persistence threshold, which causes fewer critical points to be preserved on the contour tree, while LOPC is able to preserve all critical points. Serially, LOPC performs better than the other topology preserving compressors as well. This is likely due to the other compressors building a new contour tree in each iteration, a step that LOPC is able to forgo due to its entirely different approach for preserving the topology.

The CPU throughputs of LOPC are much lower than the GPU throughputs. This is not only due to the CPU/GPU performance difference but also due to our serial and OpenMP implementations being straightforward backports to make them compatible with the CUDA code. Hence, they are not particularly optimized for CPUs and only yield a small speedup when run on all cores. The GPU implementation is the main contribution of our work and addresses the problem of CPU codes sometimes being too slow to be practical in this domain. We mainly created the CPU code so that the data can be compressed and decompressed on both types of devices.

Naturally, LOPC is slower than the non-topology-preserving compressors, though its CUDA version is on par with some of the serial compressors. Compared to other GPU-based compressors like PFPL and FPCompress that do not need to preserve topology, LOPC can be slower by over a factor of 100 (e.g., on the S3D input). This is expected because LOPC performs most of its topology preservation work during compression and explains why its decompression performance is much closer to the non-topology-preserving compressors. Given the significant effect that topology preservation has on compression throughput, other compressors should be used if

TABLE VI
COMPARISON WITH NON-TOPOLOGY-PRESERVING COMPRESSORS FOR THE 1E-2 NOA ERROR BOUND

	LOPC							FPCompress	
	Ser	OMP	CUDA	SZ3	PFPL	FPZip	ZSTD	Speed	Ratio
Compression Ratio									
Isabel	5.47	5.47	5.47	**1,258.13**	33.56	2.42	1.16	1.52	1.73
Tangaroa	4.39	4.39	4.39	**285.80**	34.55	3.85	1.43	1.76	2.12
Earthquake	8.79	8.79	8.79	**1,336.56**	102.00	1.31	1.18	1.19	1.19
Ionization	10.19	10.19	10.19	**422.57**	66.92	2.26	9.50	1.47	2.89
Miranda	10.05	10.05	10.05	**1,350.33**	94.05	2.06	2.07	1.84	1.88
S3D	9.48	9.48	9.48	**3,097.37**	63.69	1.68	1.09	1.33	1.33
SCALE	10.74	10.74	10.74	**1,334.26**	120.57	1.48	2.76	1.35	1.36
QMCPACK	9.07	9.07	9.07	**853.99**	57.97	1.53	1.61	1.33	1.43
Geomean	8.18	8.18	8.18	**995.92**	65.32	1.96	1.92	1.46	1.67
Compression Throughput (MB/s)									
Isabel	9	17	750	268	229,479	96	4	**234,375**	194,805
Tangaroa	1	1	104	240	**207,480**	130	3	75,349	63,066
Earthquake	7	13	940	424	**166,903**	130	5	76,011	11,515
Ionization	3	4	313	469	**171,798**	183	1	103,390	12,209
Miranda	5	8	414	460	223,927	169	2	**434,518**	9,798
S3D	TO	9	408	456	241,723	164	2	**469,043**	9,859
SCALE	TO	TO	34	488	241,089	139	1	**473,557**	13,157
QMCPACK	12	16	730	327	**52,124**	137	6	12,959	5,566
Geomean	5	8	314	379	**176,190**	141	3	142,869	18,234
Decompression Throughput (MB/s)									
Isabel	310	2,601	28,754	644	**314,809**	60	526	201,794	143,770
Tangaroa	316	2,592	19,059	607	**278,159**	68	298	72,000	62,760
Earthquake	441	1,896	28,200	992	**340,120**	63	513	85,455	126,457
Ionization	339	3,628	33,305	1,034	**351,053**	81	726	117,775	32,821
Miranda	422	4,314	30,199	979	**429,901**	72	651	378,433	184,703
S3D	TO	3,448	123,457	979	392,667	68	792	**458,505**	210,084
SCALE	TO	TO	112,896	1,009	430,160	66	495	**464,975**	211,455
QMCPACK	429	4,380	10,950	693	**164,518**	17	17	14,504	54,752
Geomean	372	3,142	35,229	849	**325,142**	57	354	142,613	106,719

TABLE VII
COMPARISON WITH NON-TOPOLOGY-PRESERVING COMPRESSORS FOR THE 1E-4 NOA ERROR BOUND

	LOPC							FPCompress	
	Ser	OMP	CUDA	SZ3	PFPL	FPZip	ZSTD	Speed	Ratio
Compression Ratio									
Isabel	4.60	4.60	4.60	**23.06**	6.90	2.42	1.16	1.52	1.73
Tangaroa	4.13	4.13	4.13	**24.49**	7.32	3.85	1.43	1.76	2.12
Earthquake	7.73	7.73	7.73	**33.99**	13.97	1.31	1.18	1.19	1.19
Ionization	8.36	8.36	8.36	**54.59**	16.29	2.26	9.50	1.47	2.89
Miranda	8.81	8.81	8.81	**81.92**	30.34	2.06	2.07	1.84	1.88
S3D	9.11	9.11	9.11	**127.02**	13.91	1.68	1.09	1.33	1.33
SCALE	9.53	9.53	9.53	**30.03**	20.93	1.48	2.76	1.35	1.36
QMCPACK	8.23	8.23	8.23	**80.89**	12.58	1.53	1.61	1.33	1.43
Geomean	7.26	7.26	7.26	**47.63**	13.75	1.96	1.92	1.46	1.67
Compression Throughput (MB/s)									
Isabel	79	8,182	8,411	195	225,361	96	4	**234,375**	194,805
Tangaroa	46	199	2,592	206	**205,793**	130	3	75,349	63,066
Earthquake	83	470	7,050	329	**162,953**	130	5	76,011	11,515
Ionization	48	102	4,063	384	**170,300**	183	1	103,390	12,209
Miranda	7	13	643	434	223,080	169	2	**434,518**	9,798
S3D	8	568	21,044	407	229,240	164	2	**469,043**	9,859
SCALE	TO	3	179	376	244,565	139	1	**473,557**	13,157
QMCPACK	146	438	4,380	251	**49,738**	137	6	12,959	5,566
Geomean	38	173	3,003	310	**172,953**	141	3	142,869	18,234
Decompression Throughput (MB/s)									
Isabel	307	2,894	31,142	416	**304,679**	60	526	201,794	143,770
Tangaroa	324	2,057	27,284	446	**275,136**	68	298	72,000	62,760
Earthquake	470	1,986	21,793	652	**335,459**	63	513	85,455	126,457
Ionization	406	4,515	29,921	772	**348,070**	81	726	117,775	32,821
Miranda	425	3,355	33,780	835	**426,173**	72	651	378,433	184,703
S3D	383	3,448	50,618	864	384,781	68	792	**458,505**	210,084
SCALE	TO	3,421	112,896	608	402,804	66	495	**464,975**	211,455
QMCPACK	438	4,380	10,950	602	**164,518**	17	17	14,504	54,752
Geomean	389	3,132	32,253	630	**318,677**	57	354	142,613	106,719

topology is not of interest to the downstream data processing.

D. Effects of Error Bound on LOPC Compression

The user-requested error bound affects all tested lossy compressors in terms of both compression ratio and compression/decompression speed. In this section, we discuss how each compressor behaves when the error bound is changed. Lossy compressors that do not preserve topology tend to produce significantly lower compression ratios for smaller error bounds but only incur minor slowdowns [13]. For these compressors, the computation is not directly affected by the error bound, and the slowdown stems from the increased amount of data that needs to be written due to the lower compressibility with tighter error bounds.

For the topology-preserving compressors, the compression speed is largely determined by how much correction needs to be performed. Figure 4 illustrates this for LOPC on 7 error bounds. For the loosest tested error bound, where almost all information is lost, LOPC must perform a large amount of correction and, thus, has the highest runtime. In contrast, the tightest tested error bound yields the lowest runtime because LOPC is "correcting" data that already has many of the local-order relationships intact.

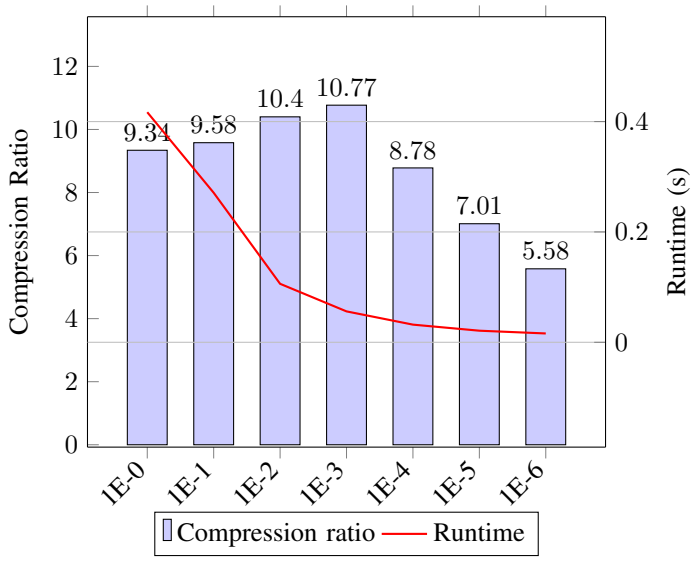

Fig. 4. Geometric-mean compression ratio and compression runtime of LOPC on 7 NOA error bounds.

Similar runtime behavior is observed in Tables IV and V for the compressors that preserve the contour tree within a persistence threshold. These compressors have an additional parameter for the persistence threshold, which also affects how much correction must be performed. In particular, more correction must be done and, therefore, more time is taken when the persistence threshold is smaller than the error bound.

Figure 4 includes LOPC's compression ratio. The relationship between the error bound and the compression ratio is not as straightforward as the runtime relationship. The geometric-mean compression ratio is 9.3 at an error bound of 1, increases to a maximum of 10.8 at an error bound of 1E-3, and then decreases to 5.6 at the lowest error bound of 1E-6. This behavior is due to the dual-compression design of LOPC. Since LOPC processes the bins and subbins separately, there is an optimal error bound where the information is most evenly split between the bin and subbin values. TopoA behaves similarly, where the best compressing base compressor does not always yield the highest final compression ratio [18].

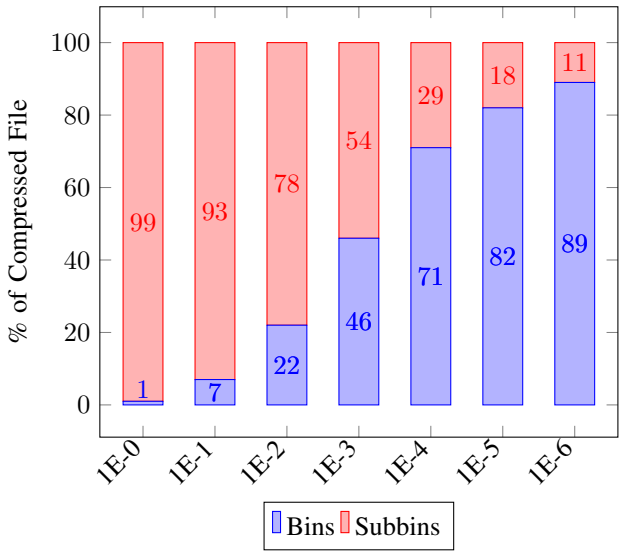

Fig. 5. Average portion of the compressed file that is bin data and subbin data for LOPC on 7 NOA error bounds.

To help explain these compression ratio results, Figure 5 shows the average fraction of the compressed file taken up by the compressed bin data and the compressed subbin data across the 7 tested error bounds. As discussed, at the loosest error bound of 1, the main data (i.e., the bin information) is almost entirely lost. Hence, the bin data is extremely compressible as it contains almost no information. In contrast, the subbins contain the corrections for the main data to maintain local order. For this reason, the subbins are much less compressible and make up 99% of the compressed file. However, the overall compression ratio is still high because the subbins only contain small numbers that are relatively easy to compress.

As the error bound decreases, the bin data starts to take up a larger portion of the compressed file. The highest observed compression ratio is seen at an error bound of 1E-3, where the bins and subbins take up an approximately equal portion of the compressed file. As discussed in Section IV, we used the LC framework to generate good compression algorithms for the bins and subbins. We targeted this search around the commonly used 1E-3 error bound, which contributes to the compression ratio peaking for LOPC at 1E-3.

The worst compression ratios are seen at the tightest error bounds, where the bins take up most of the compressed file. This is because at these error bounds, most of the information is stored in the bins. In fact, we are approaching lossless compression with these tight error bounds, meaning most of the original floating-point information is actually retained.

In summary, all lossy compressors are affected by changes in the user-provided parameters. The non-topology-preserving

lossy compressors, which tend to be memory bound, are mainly affected in compression ratio, with only a slight slowdown at lower error bounds. In contrast, the more compute-bound topology-preserving compressors actually run faster at tighter error bounds, even while producing lower compression ratios, due to the decreased amount of topology correction that must be performed.

E. Quality of Reconstructed Data

Tables VIII and IX show the quality of the reconstructed data yielded by each tested compressor. We show peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM). Both are higher-is-better metrics that are commonly used when evaluating reconstructed lossy data. SSIM measures visual quality and can highlight differences between compressors even when they use realistic error bounds, where direct visualization would not show appreciable differences to a human observer.

Overall, TopoA augmented SZ3 produces the highest-quality reconstructed data, followed by LOPC. This is explained by the differences in how TopoA fixes errors compared to LOPC. When TopoA finds a topology problem, it tightens the error bound, bringing that value closer to the original, which necessarily moves the values closer to lossless encoding. This is in contrast to LOPC, which fixes problems with local ordering by monotonically increasing subbin values. This means that, while the local order is preserved, the individual values may be shifted further from their original value than traditional quantization. For this reason, relaxing the local order preservation would not necessarily improve these metrics. To obtain higher PSNR or SSIM values with LOPC, the error bound needs to be tightened, which mimics the preservation that TopoA performs. Even with this consideration, LOPC consistently produces higher quality reconstructions than the other topology-preserving compressors while exceeding their speed by a large margin.

VII. Summary and Conclusions

This paper describes LOPC, a local-order-preserving data compression algorithm. It is the first lossy compressor that fully preserves the local order (and, by extension, all critical points) of scalar fields. Moreover, it strictly guarantees the user-specified point-wise error bound.

We evaluated LOPC, 3 state-of-the-art topology-preserving lossy compressors, and 5 other leading compressors on 2 single- and 6 double-precision inputs. LOPC is the fastest topology-preserving compressor in all cases. At the maximum, it compresses 152,000 times faster than TopoSZ. On average, is 345 times faster than TopoA-augmented SZ3 at an error bound of 1E-4 and a persistence threshold of 1.5E-3. This speed is important for scientific simulations and instruments that produce data at high rates, where the prior state of the art in topology-preserving lossy compression is too slow to be effectively utilized.

LOPC guarantees not only the error bound, which many lossy compressors do not, but also that local ordering and all critical points are preserved. None of the prior topology-preserving compressors can do this without setting the persistence threshold to 0, which slows them down even more.

We hope that LOPC helps enable topology preservation in science where it was previously prohibitively slow to do so.

Acknowledgments

This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Research (ASCR), under Awards DE-SC0022223 and DE-SC0021015. Additional support was provided by the U.S. National Science Foundation (NSF) under Awards CCF-2403380, CCF-2403379, CCF-2346394, CCF-2217154, and OAC-2313124.

References

[1] Anju Mongandampulath Akathoott, Andrew Rodriguez, and Martin Burtscher. SLEEK: Compressing Memory Copies for Floating-Point Data on GPUs. In *2026 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 874–887, May 2026.

[2] Noushin Azami, Alex Fallin, Brandon Burtchell, Andrew Rodriguez, Benila Jerald, Yiqian Liu, and Martin Burtscher. LC Git Repository. https://github.com/burtscher/LC-framework, 2025. Accessed: 2025-01-07.

[3] Noushin Azami, Alex Fallin, and Martin Burtscher. Efficient Lossless Compression of Scientific Floating-Point Data on CPUs and GPUs. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, ASPLOS '25, pages 395–409, New York, NY, USA, 2025. Association for Computing Machinery.

[4] Harsh Bhatia, Attila G Gyulassy, Vincenzo Lordi, John E Pask, Valerio Pascucci, and Peer-Timo Bremer. Topoms: Comprehensive topological exploration for molecular and condensed-matter systems. *Journal of computational chemistry*, 39(16):936–952, 2018.

[5] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

[6] Yann Collet. Zstandard Git Repository. https://github.com/facebook/zstd, 2016. Accessed: 2025-01-07.

[7] Yann Collet and Murray Kucherawy. Zstandard Compression and the 'application/zstd' Media Type. RFC 8878, February 2021.

[8] L. Peter Deutsch. GZIP file format specification version 4.3. RFC 1952, May 1996.

[9] Sheng Di and Franck Cappello. Fast Error-Bounded Lossy HPC Data Compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 730–739, Los Alamitos, CA, USA, may 2016. IEEE Computer Society.

[10] Jarek Duda, Khalid Tahboub, Neeraj J. Gadgil, and Edward J. Delp. The use of asymmetric numeral systems as an accurate replacement for Huffman coding. In *2015 Picture Coding Symposium (PCS)*, pages 65–69, 2015.

[11] Herbert Edelsbrunner, John Harer, and Afra Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proceedings of the seventeenth annual*

TABLE VIII
COMPARISON OF PSNR AND SSIM FOR THE 1E-2 NOA ERROR BOUND

	LOPC	TopoA SZ3 ϵ = 1.5x EB	TopoA SZ3 ϵ = 0.5x EB	TopoSZ	TopoQZ	SZ3	PFPL
PSNR							
Isabel	52.9	55.2	**57.3**	50.4	53.0	50.8	44.8
Tangaroa	53.0	56.7	**58.4**	DNF	57.7	53.0	45.0
Earthquake	53.3	59.3	**63.0**	47.8	50.0	53.1	47.4
Ionization	51.4	54.7	**54.7**	DNF	50.0	53.9	49.2
Miranda	**59.9**	46.3	TO	41.1	49.1	54.7	52.0
S3D	56.0	62.6	TO	11.7	**64.4**	54.2	44.7
SCALE	**56.3**	49.1	TO	8.0	51.2	51.2	46.5
QMCPACK	53.1	55.3	**60.6**	49.6	51.5	47.6	44.5
Geomean	54.4	54.7	**58.7**	27.8	53.1	52.3	46.7
SSIM							
Isabel	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.97
Tangaroa	**1.00**	**1.00**	**1.00**	DNF	**1.00**	**1.00**	0.97
Earthquake	0.92	0.98	**0.99**	0.75	0.82	0.94	0.71
Ionization	**1.00**	**1.00**	**1.00**	DNF	**1.00**	**1.00**	1.00
Miranda	**0.31**	0.26	TO	0.29	0.30	0.27	0.31
S3D	0.69	0.95	TO	0.01	0.94	**0.98**	0.44
SCALE	0.62	**0.68**	TO	0.00	0.62	0.67	0.59
QMCPACK	0.99	1.00	**1.00**	0.98	0.99	0.97	0.95
Geomean	0.77	0.80	**1.00**	0.14	0.78	0.80	0.69

TABLE IX
COMPARISON OF PSNR AND SSIM FOR THE 1E-4 NOA ERROR BOUND

	LOPC	TopoA SZ3 ϵ = 1.5x EB	TopoA SZ3 ϵ = 0.5x EB	TopoSZ	TopoQZ	SZ3	PFPL
PSNR							
Isabel	**95.1**	94.6	94.4	TO	91.5	86.4	84.8
Tangaroa	95.1	**96.7**	96.6	90.4	90.9	87.8	84.8
Earthquake	**95.1**	94.5	TO	90.4	91.8	85.4	84.7
Ionization	95.7	97.0	**97.8**	88.7	90.5	89.4	86.4
Miranda	**95.6**	94.0	TO	83.2	89.3	89.6	90.0
S3D	92.2	**93.4**	TO	TO	91.2	89.4	84.8
SCALE	91.1	inf	TO	TO	**96.0**	86.7	84.4
QMCPACK	95.1	99.3	**99.3**	90.8	90.9	88.9	84.8
Geomean	94.3	95.6	**97.0**	88.7	91.5	87.9	85.5
SSIM							
Isabel	**1.00**	**1.00**	**1.00**	TO	**1.00**	**1.00**	**1.00**
Tangaroa	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99
Earthquake	**1.00**	**1.00**	TO	**1.00**	**1.00**	**1.00**	1.00
Ionization	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Miranda	**0.34**	0.30	TO	0.30	**0.34**	0.30	0.33
S3D	**1.00**	**1.00**	TO	TO	**1.00**	**1.00**	1.00
SCALE	0.81	**1.00**	TO	TO	0.86	0.85	0.76
QMCPACK	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00
Geomean	0.85	0.86	**1.00**	0.78	0.86	0.84	0.84

symposium on Computational geometry, pages 70–79, 2001.

[12] Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics (tog)*, 9(1):66–104, 1990.

[13] Alex Fallin, Noushin Azami, Sheng Di, Franck Cappello, and Martin Burtscher. Fast and Effective Lossy Compression on GPUs and CPUs with Guaranteed Error Bounds. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 874–887, June 2025.

[14] Alex Fallin, Noushin Azami, Sheng Di, Franck Cappello, and Martin Burtscher. PFPL Git Repository. https://github.com/burtscher/PFPL, 2025. Accessed: 2025-02-10.

[15] Alex Fallin and Martin Burtscher. Lessons learned on the path to guaranteeing the error bound in lossy quantizers. *arXiv preprint arXiv:2407.15037*, 2024.

[16] Alex Fallin, Nathaniel Gorski, Tripti Agarwal, Bei Wang, Ganesh Gopalakrishnan, and Martin Burtscher. LOPC Git Repository. https://github.com/burtscher/LOPC, 2026. Accessed: 2026-06-02.

[17] Nathaniel Gorski, Xin Liang, Hanqi Guo, and Bei Wang. Tfz: Topology-preserving compression of 2d symmetric and asymmetric second-order tensor fields. *arXiv preprint arXiv:2508.09235*, 2025.

[18] Nathaniel Gorski, Xin Liang, Hanqi Guo, Lin Yan, and Bei Wang. A general framework for augmenting lossy compressors with topological guarantees. *IEEE*

Transactions on Visualization and Computer Graphics, 2025.

[19] David A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[20] Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Comput. Graph. Forum*, 22:343–348, 09 2003.

[21] F. Jager. Delta Modulation — A Method of PCM Transmission Using the One Unit Code. *Philips Res. Repts.*, 7, 01 1952.

[22] J. E. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. M. Arblaster, S. C. Bates, G. Danabasoglu, J. Edwards, M. Holland, P. Kushner, J.-F. Lamarque, D. Lawrence, K. Lindsay, A. Middleton, E. Munoz, R. Neale, K. Oleson, L. Polvani, and M. Vertenstein. "The Community Earth System Model (CESM) Large Ensemble Project: A Community Resource for Studying Climate Change in the Presence of Internal Climate Variability". *Bulletin of the American Meteorological Society*, 96(8):1333–1349, 2015.

[23] Abraham Lempel and Jacob Ziv. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.

[24] Mingzhe Li, Dwaipayan Chatterjee, Franziska Glassmeier, Fabian Senf, and Bei Wang. Tracking low-level cloud systems with topology. *arXiv preprint arXiv:2505.10850*, 2025.

[25] Yuxiao Li, Xin Liang, Bei Wang, Yongfeng Qiu, Lin Yan, and Hanqi Guo. Msz: An efficient parallel algorithm for correcting morse-smale segmentations in error-bounded lossy compressors. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[26] Xin Liang, Sheng Di, Franck Cappello, Mukund Raj, Chunhui Liu, Kenji Ono, Zizhong Chen, Tom Peterka, and Hanqi Guo. Toward feature-preserving vector field compression. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5434–5450, 2022.

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

[28] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, 2018.

[29] Xin Liang, Kai Zhao, Sheng Di, Sihuan Li, Robert Underwood, Ali M. Gok, Jiannan Tian, Junjing Deng, Jon C. Calhoun, Dingwen Tao, Zizhong Chen, and Franck Cappello. SZ3: A Modular Framework for Composing Prediction-Based Error-Bounded Lossy Compressors. *IEEE Transactions on Big Data*, 9(2):485–498, 2023.

[30] Peter Lindstrom and Martin Isenburg. Fast and efficient compression of floating-point data. *IEEE transactions on visualization and computer graphics*, 12(5):1245–1250, 2006.

[31] Zixiang Luo, Chenyu Xu, Zhen Zhang, and Wenfei Jin. A topology-preserving dimensionality reduction method for single-cell RNA-seq data using graph autoencoder. *Scientific Reports*, 11:20028, 10 2021.

[32] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. In *International conference on machine learning*, pages 7045–7054. PMLR, 2020.

[33] K. B. Olsen, S. M. Day, J. B. Minster, Y. Cui, A. Chourasia, D. Okaya, P. Maechling, and T. Jordan. Terashake2: Spontaneous rupture simulations of mw 7.7 earthquakes on the southern san andreas fault. *Bulletin of the Seismological Society of America*, 98(3):1162–1185, 06 2008.

[34] Mathieu Pont, Jules Vidal, Julie Delon, and Julien Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):291–301, 2022.

[35] Stéphane Popinet, Murray Smith, and Craig Stevens. Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. *Journal of Atmospheric and Oceanic Technology*, 21(10):1575–1589, 2004.

[36] Chethan Krishnamurthy Ramanaik, Anna Willmann, Juan-Esteban Suarez Cardona, Pia Hanfeld, Nico Hoffmann, and Michael Hecht. Ensuring Topological Data-Structure Preservation under Autoencoder Compression Due to Latent Space Regularization in Gauss–Legendre Nodes. *Axioms*, 13(8):535, 2024.

[37] SDRBench Inputs, https://sdrbench.github.io/, 2023.

[38] Maxime Soler, Mélanie Plainchault, Bruno Conche, and Julien Tierny. Topologically controlled lossy compression. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 46–55. IEEE, 2018.

[39] Thierry Sousbie. The persistent cosmic web and its filamentary structure–i. theory and implementation. *Monthly notices of the royal astronomical society*, 414(1):350–383, 2011.

[40] Julien Tierny. *Topological Data Analysis for Scientific Visualization*. Mathematics and Visualization. Springer Cham, 2018.

[41] Julien Tierny and Valerio Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE transactions on visualization and computer graphics*, 18(12):2005–2013, 2012.

[42] Jules Vidal, Pierre Guillou, and Julien Tierny. A progressive approach to scalar field topology. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):2833–2850, 2021.

[43] Daniel Whalen and Michael L. Norman. Ionization front instabilities in primordial h ii regions. *The Astrophysical Journal*, 673(2):664, feb 2008.

[44] Mingze Xia, Bei Wang, Yuxiao Li, Pu Jiao, Xin Liang, and Hanqi Guo. Tspsz: An efficient parallel

error-bounded lossy compressor for topological skeleton preservation. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 3682–3695. IEEE Computer Society, 2025.

[45] Lin Yan, Xin Liang, Hanqi Guo, and Bei Wang. Toposz: Preserving topology in error-bounded lossy compression. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1302–1312, 2023.

[46] Kai Zhao, Sheng Di, Maxim Dmitriev, Thierry-Laurent D. Tonellot, Zizhong Chen, and Franck Cappello. Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1643–1654, 2021.

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