2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) | 978-1-6654-1016-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/HIPC53243.2021.00036

Optimizing Multi-Range based Error-Bounded Lossy Compression for Scientific Datasets

Yuanjian Liu*, Sheng Di†, Kai Zhao‡, Sian Jin§, Cheng Wang†,
Kyle Chard*, Dingwen Tao§, Ian Foster*, Franck Cappello†
* University of Chicago, Chicago, IL, USA
† Argonne National Laboratory, Lemont, IL, USA
‡ University of California, Riverside, CA, USA
§ Washington State University, Washington, USA

***Abstract*—Vast volumes of scientific data cannot be stored and transferred efficiently because of limited I/O bandwidth, network bandwidth, and storage capacity. Error-bounded lossy compression can be an effective method for resolving these big data issues, since not only can it significantly reduce the data size but it can also control the data distortion based on user-defined error bounds. In practice, many scientific applications have specific data fidelity requirements across different value ranges/intervals of the dataset for the lossy compression, in order to guarantee that the reconstructed data are valid for post hoc analysis. Existing state-of-the-art error-bounded lossy compressors, however, do not support multi-range based error-bounds in the lossy compression, leaving a critical gap that hampers their effective use in practice. In this work, we address this issue by proposing a multi-range based error-bounded lossy compressor based on the state-of-the-art SZ lossy compressor. Our approach allows users to set different error bounds in different value ranges for a compressoin task. We evaluate our approach on several real-world datasets and show that it can obtain a higher visual quality or data fidelity on reconstructed data with the same or even higher compression ratios achieved by SZ.**

I. Introduction

Error-bounded lossy compressors [1–7] are a promising solution to reducing scientific data volumes while also addressing user data fidelity requirements. For example, SZ [2, 8], ZFP [1], and MGARD [9] allow users to set an absolute error bound (i.e., a threshold) when performing lossy compression such that the difference between the original data and reconstructed data is bounded by that threshold. Climate scientists have verified that the reconstructed data generated by error-bounded lossy compressors are acceptable for post hoc analysis [10–12].

Existing error-bounded lossy compressors have a significant limitation: none are able to set specific error bounds in different value ranges. According to environmental scientists, for example, different values in a dataset may have different significance to the post hoc analysis, such that they may wish to set different error bounds for different value ranges in the dataset. Specifically, in order to trace a hurricane's moving trajectory over the sea, only the data points whose water surface values are greater than a threshold (e.g., 1 meter) are interesting for particular environmental science analyses. Cosmological simulations [13] present another typical example, in which scientists often have a specific quantity of interest (e.g., dark matter halo cell information) for their post hoc analysis. According to the dark matter halo analysis algorithm, the construction of dark matter halos is primarily determined by the data value in a specific value range: $[81, 83]$. In order to preserve the features (e.g., count and location) of the dark matter halos, the values in this range should have higher precision than the values in other ranges,

We present here a novel compressor based on the SZ error-bounded lossy compression framework,[1] which allows users to set different error bounds in various value ranges, such that the reconstructed data can meet users' required quality better than the traditional global error-bounded lossy compression.

We summarize our contributions as follows.

- We propose a multi-range error-bounded lossy compression method based on the classic SZ error-bounded lossy compression model, which, to the best of our knowledge, is the first such attempt.
- We optimize the compression quality and performance by investigating three different situations where the prediction values are located compared with their raw values during the quantization stage.
- Our experiments demonstrate that our method can obtain a better visual quality or data fidelity in the lossy-reconstructed data for different applications, with the same compression ratios compared with the global error-bounded compressor. We also show that our method exhibits good scalability in compression time compared with the parallel file system's I/O cost.

The rest of the paper is organized as follows. In Section II we discuss related work. In Section III we propose several algorithms to preserve the user-required prerequisites, and optimize compression quality and performance for different cases. In Section IV we present our evaluation results. In Section V we summarize our findings and conclude with a vision of future work.

[1]We adopt the SZ compression framework as it provides leading error-bounded lossy compression quality as verified by several studies of different scientific datasets [2, 8, 14, 15].

2640-0316/21/$31.00 ©2021 IEEE
DOI 10.1109/HiPC53243.2021.00036

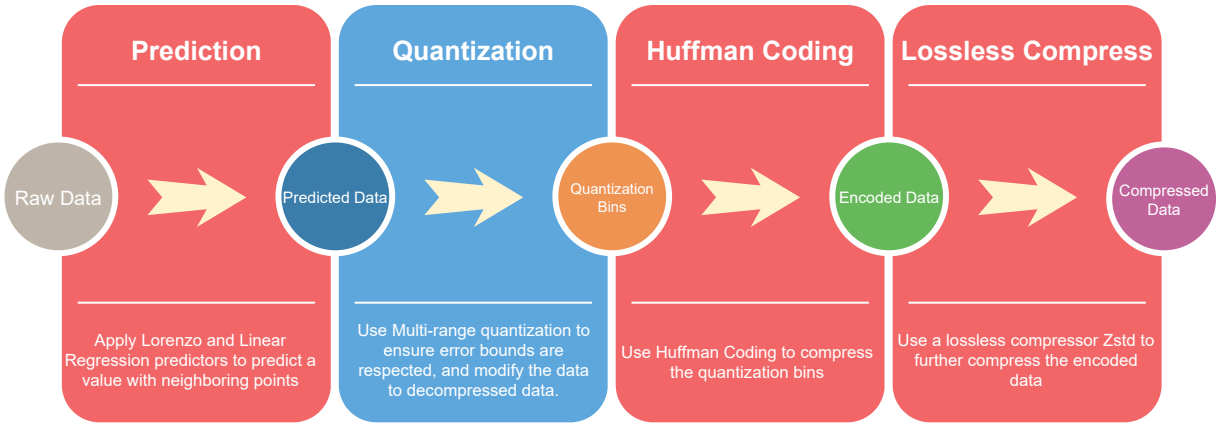


Fig. 1: **Stages of multi-range error-bounded lossy compression**: (1) The framework follows the procedure of SZ; (2) Multiple error bounds are handled in the quantization stage. Our method uses the same quantization array to accommodate different error bounds and therefore does not introduce overheads in the quantization stage but only changes the distribution of quantization values.

II. Related Work

Data compressors can be split into two classes: lossless compression [16–19] and lossy compression [1, 2, 8, 20, 21]. The former introduces no data loss during compression, but it suffers from low compression ratios (generally ∼2:1 [22, 23]). The latter can achieve very high compression ratios [1, 2, 8, 15], but potential data loss may distort analysis results. To address this issue, researchers have studied error-bounded lossy compressors for scientific data.

To satisfy user demands on a specific quality of interest, researchers have recently studied how to respect specific metrics. Existing compressors such as SZ [2, 8, 15], ZFP [1, 24], and MGARD [9] support absolute error bounds. Other error-bounding approaches have also been exploited to adapt to diverse user requirements. For instance, SZ supports pointwise relative error bound [25] while Digit Rounding [26], Bit Grooming [21], zfp [1], and FPZIP [20] support a kind of precision mode that allows users to specify the number of bits to be truncated in the end of the mantissa, in order to control the data distortion at different levels.

However, none of the existing error-bounded lossy compressors allow users to treat different values differently and hence impose a significant impediment to the practical usage of such compressors. We observed that users often have different precision demands for different value ranges, which are determined by their diverse post hoc analysis purposes and quantities/features of interest.

III. Multi-Range Error-Bounded Lossy Compression Framework

We introduce a multi-range error-bounded lossy compression framework as shown in Fig. 1. In the following text, we first introduce the compression model and then describe our approach to allow different error bounds to coexist in the same compression run.

A. Prediction-based Lossy Compression Stages

As is shown in Fig. 1, the compression framework is composed of four key stages: prediction, quantization, Huffman encoding, and lossless compression. Given a set of raw data, the predictor scans the whole dataset (either pointwise [2, 8] or blockwise [15]) to predict the data values. In a 1D dataset, the prediction method is simply a first-order Lorenzo predictor [8]. In a 2D or 3D dataset, the predictor adopts a hybrid data prediction method combining the first-order Lorenzo predictor and a linear-regression-based predictor [15].

In the second stage, the traditional method uses a linear quantization algorithm to convert the distance between the predicted value and original value to an integer value (called the quantization code) for each data point. In spite of the simplicity it brings to correct predicted values, the linear quantization does not allow for different error bounds because different error bounds will change the distance represented by each quantization code and cause confusion when decompressing the data. Therefore, we design a new quantization algorithm that compiles multiple value ranges with various error bounds into the same quantization array.

A customized Huffman encoder is then applied to compress the integer quantization codes, followed by a lossless dictionary coding (using Zstd [16] by default). We summarize all the notations in Table I in order to help understand the following text.

B. Preserving Multi-Range Error Bounds

The multi-range error-bounded model is defined by an array of triplets, each containing the *low*, *high*, and *error bound*. Fig. 2 illustrates our fundamental idea using a simplified diagram with relatively large error bounds. In this example, the user specifies different error bounds for four value ranges: $[-100, 0)$, $[0, 14)$, $[14, 38)$, and $[38, 238)$; the error bounds are 10, 1, 3, and 50, respectively. The idea is to use different quantization bins in different ranges. In Fig. 2, each square denotes a quantization bin in its corresponding value range. Our algorithm will identify in which range the original data

and predicted value fall (they may be in two different ranges). The quantization method will then differ in different situations. In the decompression stage, the only available information are the predicted value, the quantization code, and the definitions of ranges. In contrast to the linear quantization method, our solution can reconstruct the data of different error bounds using the same quantization array.

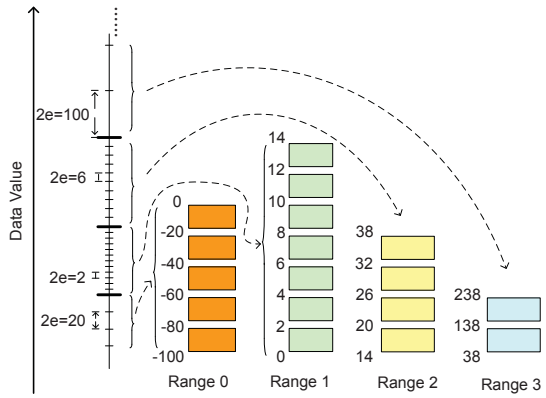


Fig. 2: Multi-range error-bounded model. This example shows a few error bounds in each range (illustrated by the colored boxes) for simplicity of description; in practice, each range may contain hundreds or thousands of error bounds.

TABLE I: Notations For Multi-Error-Bounded Compression

Notation	Description
d_i	original data value at position i
p_i	predicted value of d_i
$\hat{d}_i$	reconstructed data value after decompression
$r(x)$	the value range of data point x (d_i)
$e(x)$	specified error bound based on a value range (x=$r(d_i)$)
$l(x)$	length of some value range ($x = r(d_i)$)
$low(r(x))$	lower boundary of value range $r(x)$
$high(r(x))$	higher boundary of value range $r(x)$
q	quantization index (a.k.a., quantization code)
q_s	shifted quantization number

We briefly describe how our algorithm handles different situations, using the notation of TABLE I. For each data point, there are three situations to deal with.

Situation 1: If the original value d_i and its predicted value p_i fall in the same range (i.e., $r(d_i) = r(p_i)$), the quantization problem falls back to the traditional linear-scale quantization [8]. Specifically, we can use the following formulas to compute the logic quantization code and decompressed data.

$$q = round(\frac{d_i - p_i}{2e(r(d_i))}) \tag{1}$$

$$\hat{d}_i = p_i + 2e(r(d_i)) \cdot q \tag{2}$$

Situations 2 and 3: These correspond to the situation where the original d_i and its predicted value p_i fall in different ranges (i.e., $r(d_i) \neq r(q_i)$). In the following text, we describe only the situation where $r(d_i) > r(q_i)$ (i.e., lines 7∼11 in the algorithm); the other situation is similar.

The fundamental idea to handle this situation is to adjust the quantization policy to use various bin lengths (or intervals) in different value ranges. The formula for reconstructing the decompressed value is given below (we assume the raw data is greater than the predicted value, without loss of generality):

$$q_t = round(\frac{d_i - low(r(d_i)) - e(r(d_i))}{2e(r(d_i))}) \tag{3}$$

$$\hat{d}_i = low(r(d_i)) + e(r(d_i)) + 2e(r(d_i)) \cdot q_t. \tag{4}$$

The core idea of decompression is to execute the similar operations in the compression stage reversely to get the decompressed data from a predicted data value and the corresponding quantization bins. We first calculate the number of quantization bins for each value range. We then decompress each data point based on the multivalue-range quantization.

Note that in the real implementation, we need to also consider several edge cases. For instance, when the original data are near the high or low bound of a range, the quantization value in this final range might be equal to $quantRange[i]$, causing the decompressed value to be in the next range. In this case, we shift the quantization by 1 in the compression stage to ensure the decompressed data and original data are in the same range.

IV. Evaluation

In this section, we evaluate our multi-range error-bounded compression quality using two real-world simulations, and compare the compression quality and performance with SZ. SZ is a global constant error-bounded lossy compressor and has been verified as one of the best error-bounded lossy compressors in most cases.

A. Evaluation of Multi-Range Error-Bounded Compression based on Visual Quality

We evaluate our approach on QMCPACK [27] and Miranda [28] datasets, as presented in Table II.

TABLE II: Basic dataset information

Dataset	# Fields	Dimensions	Science
QMCPACK	1	33120*69*69	electronic structure of atoms, molecules, and solids
Miranda	7	256*384*384	hydrodynamics code for large turbulence simulations

Figures 3 and 4 show the substantial advantage of our multi-range error-bounded compression over the traditional constant error-bounded compression, using two datasets (QMCPACK and Miranda). Specifically, the multi-range error-bounded compression preserves higher visual quality for the value ranges of interest, while achieving the same or even higher overall compression ratios by reducing precision on uninteresting values. For instance, in the QMCPACK dataset, more than 90% of the data points gather around 0, but they are smooth and easy to predict based on neighbor data points; however, the data points with values in the range of $[-8, -5]$ are the sparse and interesting values that are harder to predict

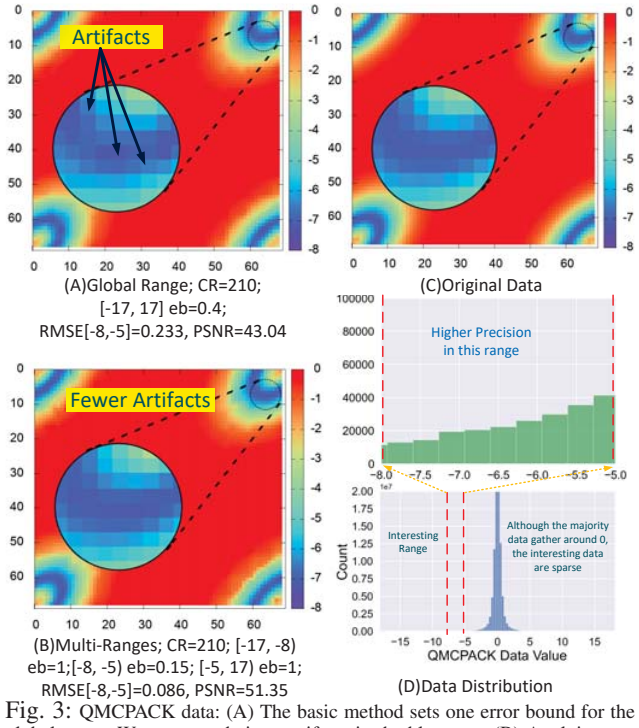


Fig. 3: QMCPACK data: (A) The basic method sets one error bound for the global range. We can see obvious artifacts in the blue area. (B) Applying our multi-range algorithm, we focus on the interesting range [-8,-5), and give it a tighter error bound of 0.15 while leaving other ranges with a higher error bound of 1. We can see fewer artifacts, while the compression ratio is kept the same as the global range method. Compared to the original data shown in (C), we can see the data in the interesting range has better visualization result.

accurately with a large error bound. They are more important and the distortion of their values are easier to observe in the visualization graphs. Our method grants a tighter error bound and thus a higher precision in this specific range while allowing more distortion in uninteresting ranges. This is the key reason that our solution can obtain higher visual quality and lower root mean squared error (RMSE) in the interesting range. A detailed comparison between the two algorithms is shown in Table III and Table IV.

TABLE III: QMCPACK RMSE & PSNR Comparison

Method	Range	eb	RMSE	PSNR
Global Range CR=210	[-17, -8]	0.4	0.232	43.067
	[-8, -5]		**0.233**	**43.041**
	[-5, 17]		0.051	56.159
Multi-Ranges CR=210	[-17, -8]	1.0	0.538	35.747
	[-8, -5]	0.15	**0.086**	**51.623**
	[-5, 17]	1.0	0.089	51.354

B. Evaluation of Compression Time and Scalability for Multi-Range Error-Bounded Compression

To evaluate the compression time and scalability, we run a series of tests in parallel on thousands of CPU cores on the LCRC Bebop cluster at Argonne National Laboratory [29]. The Bebop cluster has two partitions: KNL and BDW, corresponding to two different sets of hardwares/nodes, as shown in TABLE V.

TABLE IV: Miranda density RMSE & PSNR Comparison

Method	Range	eb	RMSE	PSNR
Global Range CR=206	[0.5, 1.4]	0.07	0.012	44.804
	[1.4, 2]		**0.036**	**34.801**
	[2, 3.5]		0.015	42.379
Multi-Ranges CR=207	[0.5, 1.4]	0.1	0.013	43.5813
	[1.4, 2]	0.05	**0.027**	**37.193**
	[2, 3.5]	0.1	0.018	40.682

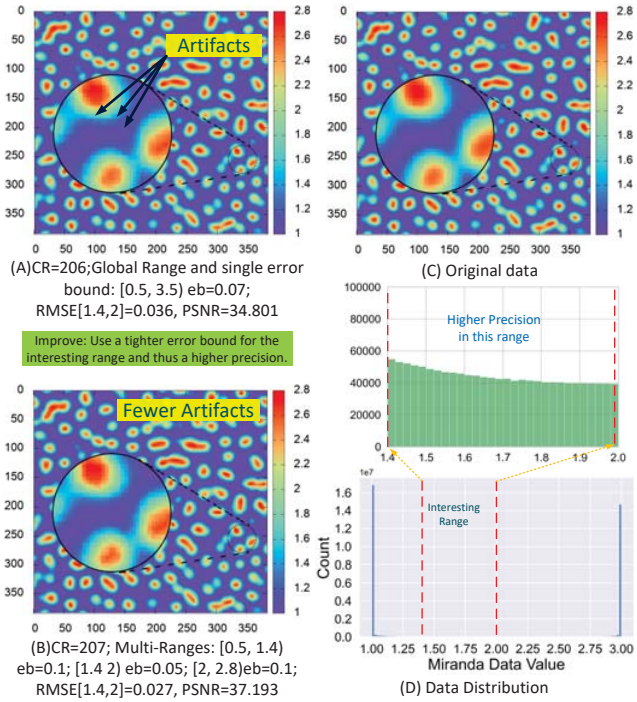


Fig. 4: Miranda density slice No. 120. Comparing (B) with (A), we can see that not only can our multi-range method preserve a high precision at a value range of interest with high compression ratio, but it also prevents the blue regions from getting distorted.

TABLE V: Machine Specifications

Partition	# Nodes	CPU	Cores/node	Memory
bdwall	664	Intel Xeon E5-2695v4	36	128GB DDR4
knlall	348	Intel Xeon Phi 7230	64	96GB DDR4

We used up to 224 nodes on the KNL partition and 70 nodes on BNL partition for our scalability evaluation. The two partitions have quite different CPUs and thus the run time on a single core are incomparable; however, their I/O throughput are similar as they share the same I/O nodes. The main difference between the two partitions is their CPU speed per core (KNL partition is significantly slower than BDW partition). The I/O time can be significantly influenced by other users' tasks and therefore we record the average run

time over 5 runs on both partitions.

We used the QMCPACK [27] dataset for these performance/scalability tests. We choose this dataset because it is more than 600MB per file and it is also stored in binary form so that the HDF5 plugins and other complicated data structure handling parts would not influence the performance evaluation. In other words, we can accurately record the time for each step of our algorithm in the MPI program. To be consistent with the previous quality tests, we set the multi-range error-bound requirement as follows: the data points in the range of $[-8, -5)$ has much higher precision than the data points in other value ranges. According to the visualization results presented in the previous section, we observe that no data distortion can be viewed by the naked eye as long as a relatively low error bound of 0.15 is used. Considering the potential impact of the lossy compression to user's analysis, we set a very low error bound (1E-5) for the range of interest: $[-8, -5)$. That is, the compressor applies the error bound of 1E-5 for the value range $[-8, -5)$, while the error bound needs be less than 0.15 in the other ranges. Preserving this condition, we perform the experiments on Bebop with different numbers of cores (each core has 600MB of raw data to compress). The results of BDW and KNL partitions are shown in Figure 5.

Based on these results, we observe that the (de)compression time does not increase with the number of cores, which shows that both our algorithm and SZ have very good scalability. The key reason that our algorithm and SZ scale well is that the lossy compression adopted in practice follows an embarrassing parallel mode: that is, there is no communication among the execution ranks/cores. The key reason our algorithm exhibits lower compression/decompression times than SZ is that our designed multi-range error-bounded compression allows us to set higher error bounds for non-interesting ranges, leading to higher compression ratios.

Based on the bar graphs in Figure 5, we can clearly see that writing times take increasing portions with the number of cores. Specifically, 'Write Zip' and 'Write DP' refer to 'writing compressed data to PFS' and 'writing decompressed data to PFS', respectively. Obviously, the I/O cost has much worse scalability than our lossy compression/decompression, especially because of the limited number of I/O nodes used by the system. This will be true in any system as there is always a limited number of I/O nodes (thus an upper bound in I/O throughput), which may easily cause a serious I/O bottleneck when a large number of cores are used concurrently.

V. Conclusion and Future Work

We propose a novel error-bounded lossy compression method that allows users to set different error bounds in various value ranges, such that the overall lossy compression quality can meet data fidelity requirements more accurately. To the best of our knowledge, this is the first attempt to develop such a compressor. Our evaluation using real-world simulations revealed the following key findings.

- Our multi-range solution can significantly enhance visual quality on the interesting data ranges with similar or even higher compression ratios when compared with traditional methods.
- Our quantitative results showed that our solution achieves better RMSE and PSNR in the user-defined ranges, and the improvement is global rather than local.
- Experiments up to 3500+ cores showed that the multi-range error-bounded lossy compressor exhibits good scalability in two different partitions, especially when compared with the parallel file system's I/O cost, which increases considerally with scale.

In the future, we will investigate the partition strategy and multi-region strategy to see if more customizable prerequisites can be satisfied. We will also investigate if automatic methods can be used to determine the ranges and error bounds for users based on the facts of data.

Acknowledgments

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This work was also supported by the U.S. Department of Energy, Office of Science and by DOE's Advanced Scientific Research Office (ASCR) under contract DE-AC02-06CH11357, and by the National Science Foundation under awards SHF-1617488, OAC-2003709 and OAC-2003624/2042084 and OAC-2104023/2104024. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

References

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

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

[3] F. Cappello, S. Di, and et al., "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.

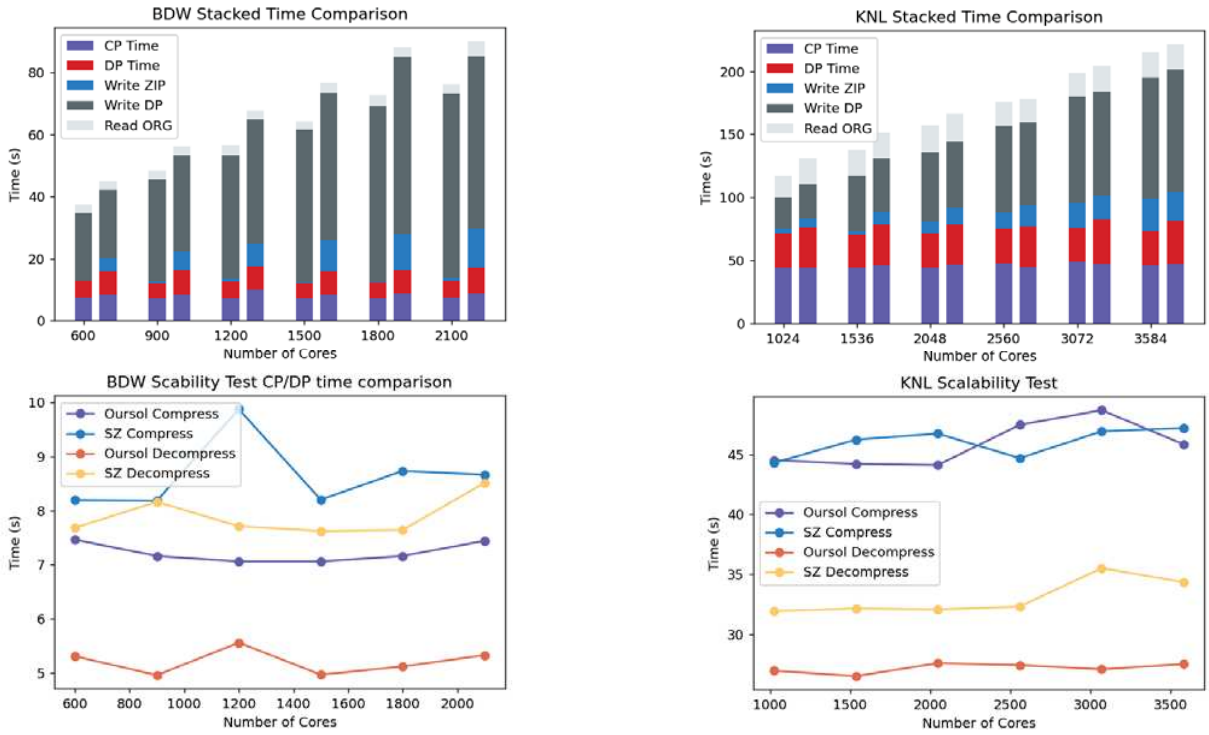
[4] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. NY, USA: Association for Computing Machinery, 2019.

[5] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Improving performance of iterative methods by lossy checkponting," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 52–65.

[6] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '19. New York, NY, USA: ACM, 2019, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/3307681.3326608

[7] D. Tao, S. Di, Z. Chen, and F. Cappello, "In-depth exploration of single-snapshot lossy compression techniques for n-body simulations," in *IEEE International Conference on Big Data*, 2017, pp. 486–493.

[8] ——, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS)*. IEEE, 2017, pp. 1129–1139.



(a) BDW partition

(b) KNL partition

Fig. 5: Performance comparison between our model and SZ for an increasing number of cores on the BDW and KNL partition. In the bar graphs (top), the left bar shows performance of our model and the right shows performance with SZ. Read ORG is the time to read the original data, CP is compression time, DP is decompression time, write ZIP is the time to write the compressed file, and write DP is the time to write the decompressed data to disk. The line graphs (bottom) compare the compression and decompression time for our solution and SZ as we scale the number of cores. The CPU speed of the BDW partition is about 8× faster than KNL partition, and we can see a larger performance advantage in this situation.

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

[10] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 914–922.

[11] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, "A methodology for evaluating the impact of data compression on climate simulation data," in *HPDC'14*, 2014, pp. 203–214.

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

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

[14] 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," https://arxiv.org/abs/2001.06139, 2020, online.

[15] 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 *IEEE International Conference on Big Data*. IEEE, 2018.

[16] Zstd, https://github.com/facebook/zstd/releases, online.

[17] Gzip, https://www.gzip.org/, online.

[18] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2008.

[19] Blosc compressor, http://blosc.org/, 2018, online.

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

[21] C. S. Zender, "Bit grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netcdf operators (nco, v4.4.8+)," *Geoscientific Model Development*, vol. 9, no. 9, pp. 3199–3211, 2016.

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

[23] P. Ratanaworabhan, J. Ke, and M. Burtscher, "Fast lossless compression of scientific floating-point data," in *Data Compression Conference (DCC'06)*, IEEE. New York, NY, USA: IEEE, 2006, pp. 133–142.

[24] J. Diffenderfer, A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom, "Error analysis of zfp compression for floating-point data," *SIAM Journal on Scientific Computing*, 02 2019.

[25] J. Zhang, X. Zhuo, A. Moon, H. Liu, and S. W. Son, "Efficient encoding and reconstruction of HPC datasets for checkpoint/restart," in *Proceedings of the 35th International Conference on Massive Storage Systems and Technology (IEEE MSST19)*, 2019.

[26] X. Delaunay, A. Courtois, and F. Gouillon, "Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netcdf-4 or hdf5 files," *Geoscientific Model Development*, vol. 12, no. 9, pp. 4099–4113, 2019.

[27] QMCPack, https://qmcpack.org/, online.

[28] Miranda, https://wci.llnl.gov/simulation/computer-codes/miranda.

[29] Bebop, https://www.lcrc.anl.gov/systems/resources/bebop, online.