Towards Guaranteeing Error Bound in DCT-based Lossy Compression

1st Jiaxi Chen
University of Massachusetts Lowell
Lowell, MA
jiaxi_chen@student.uml.edu

2nd Aekyeung Moon
ETRI
Daegu, Korea
akmoon@etri.re.kr

3rd Seung Woo Son
University of Massachusetts Lowell
Lowell, MA
seungwoo_son@uml.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020345

***Abstract*—High-performance computing (HPC) systems that run scientific simulations of significance produce a large amount of data during runtime. Transferring or storing such big datasets causes a severe I/O bottleneck and a considerable storage burden. Applying compression techniques, particularly lossy compressors, can reduce the size of the data and mitigate such overheads. Unlike lossless compression algorithms, error-controlled lossy compressors could significantly reduce the data size while respecting the user-defined error bound. DCTZ is one of the transform-based lossy compressors with a highly efficient encoding and purpose-built error control mechanism that accomplishes high compression ratios with high data fidelity. However, since DCTZ quantizes the DCT coefficients in the frequency domain, it may only partially control the relative error bound defined by the user. In this paper, we aim to improve the compression quality of DCTZ. Specifically, we propose a preconditioning method based on level offsetting and scaling to control the magnitude of input of the DCTZ framework, thereby enforcing stricter error bounds. We evaluate the performance of our method in terms of compression ratio and rate distortion with real-world HPC datasets. Our experimental result shows that our method can achieve a higher compression ratio than other state-of-the-art lossy compressors with a tighter error bound while precisely guaranteeing the user-defined error bound.**

***Index Terms*—Lossy compression, DCT, Precondition, Rate-Distortion.**

I. Introduction

Scientists in various domains use modern high-performance computing (HPC) systems to validate theories and investigate new phenomena on a scale that was nearly impossible in the past. As a result, a massive amount of data in terabytes or even petabytes would be produced by this process. For instance, large ensembles of high-fidelity simulation can generate 260 TB of data every 16 seconds across the ensemble [1]. Storing and transferring such big datasets may cause huge overhead on storage space and the I/O system. One way to mitigate such overhead [2]–[5] is to apply compression techniques, especially lossy ones, which can significantly reduce the data size.

The idea of lossy compression algorithms is to make a trade-off between the file size and the quality. A well-known example of a lossy compression technique is JPEG [6], which is widely used in image data compression as human eyes cannot notice every detail of images. The idea of lossy compression for scientific data is becoming popular recently since the slow increase in storage bandwidth barely caught up with the computing performance improvement of the supercomputers. Therefore, compressed data can reduce data traffic or burden to storage systems and network transfer. Though it will introduce errors and make the reconstructed data not identical to the original, lossy compression algorithms are applied to achieve a significantly higher compression ratio.

While for several conventional lossy compressors, such as ISABELA [7], which uses a B-spline-based curve fitting to the sorted data, the compression rate is limited due to the sorting process. FPZIP [8] traverses data in a coherent order and then applies the corresponding n-dimensional Lorenzo predictor to predict the neighboring values, which is less capable of bounding errors. SZ [9]–[11] and ZFP [12] are two well-known error-controllable lossy compressors designed for scientific datasets. SZ relies on each input value's predictability and compresses the unpredictable data by analyzing their binary representation. The limitation of the prediction-based compressor is the data dependency which means the compression performance would degrade if the data shows less structure or the error bound becomes tighter. ZFP combines a transform-based decorrelation scheme with an embedded coding scheme. With the same level of error bounds, SZ usually produces higher compression ratios than ZFP, while ZFP is faster than SZ [13].

Inspired by the JPEG compression algorithm, another transform-based lossy compressor named DCTZ [14], [15], which is based on discrete cosine transform (DCT), is designed for scientific datasets. DCTZ shows comparable performance with SZ and ZFP regarding compression ratios. However, since DCTZ truncates DCT coefficients in the frequency domain, in some cases, DCTZ cannot guarantee the user-defined error bound. Fig. 1 shows two data points (red circle) with a relative error larger than the user-defined error bound, 1E-3, in the first block of cellular-eint dataset.

This paper focuses on optimizing DCTZ with real-world scientific data sets. We explore the DCTZ framework by understanding the relationship between the original data and the relative error of the reconstructed data. We then design a preconditioning method, which employs level offsetting to map the center of the input data set to its mean value, and a scaling method, which significantly decreases the input data's magnitude. To demonstrate the effectiveness of the preconditioning method, we evaluate the performance of our

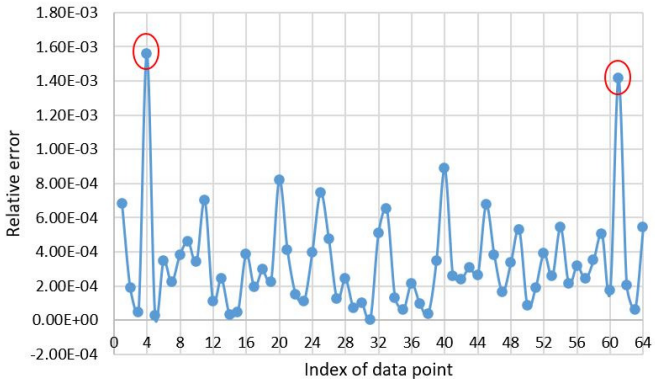


Fig. 1: The relative error of the first block of cellular-eint with an error bound of 1E-3.

method using real-world scientific datasets and compare it with other state-of-the-art lossy compressors, SZ and ZFP. Our experiments show that our proposed method can strictly guarantee user-defined error bound and achieve a competitive performance with SZ and is much better than ZFP regarding rate distortion and compression ratio. Our implementation of DCTZ in C is available at: https://github.com/swson/DCTZ.

II. Background and Motivation

A. DCTZ framework

The main idea of DCTZ is to compose a discrete cosine transform (DCT) and a truncation of DCT coefficients. Discrete cosine transform redistributes the energy contained in the signal and condenses most of the information into a small number of coefficients (low-frequency coefficients). Prior studies showed that DCT or equivalent, whose inverse has the same spectrum as the original data, has high decorrelation efficiency [16]. After DCT, most signal information is preserved in a few low-frequency coefficients (DC), while the high-frequency coefficients (AC) only contain very little information. This mechanism allows the compression of scientific datasets with a much higher compression ratio than lossless compressors. Fig. 2 depicts the structure of the DCTZ framework.

B. Compression Procedure in DCTZ

As shown in Fig. 2, DCTZ compression involves the following steps, and the quantization of AC coefficients within the bin range introduces compression errors due to truncation (marked in red box).

- Apply block decomposition, which divides the input data into blocks with 64 data points each (8×8 in 2D and $4 \times 4 \times 4$ in 3D). The 8×8 DCT is widely used in image compression, and prior studies show that it can be seamlessly applicable to floating-point numbers while providing its high energy compaction property [14], [15], [17].
- Apply DCT on each decomposed block to retrieve the representation in the frequency domain [18]–[20]. Each block's first coefficient is a DC component, and the remaining ones are AC components.
- The DC coefficients are saved as is to preserve the most informative data.
- For the AC coefficients, which take the majority part in all coefficients, DCTZ would check whether each of them is inside the bin range.
- Quantize AC coefficients inside the bin range with a uniform quantizer.
- Compress the data from previous steps, including DC, AC outside bin range, and bin index, with a lossless compressor such as zlib [21].

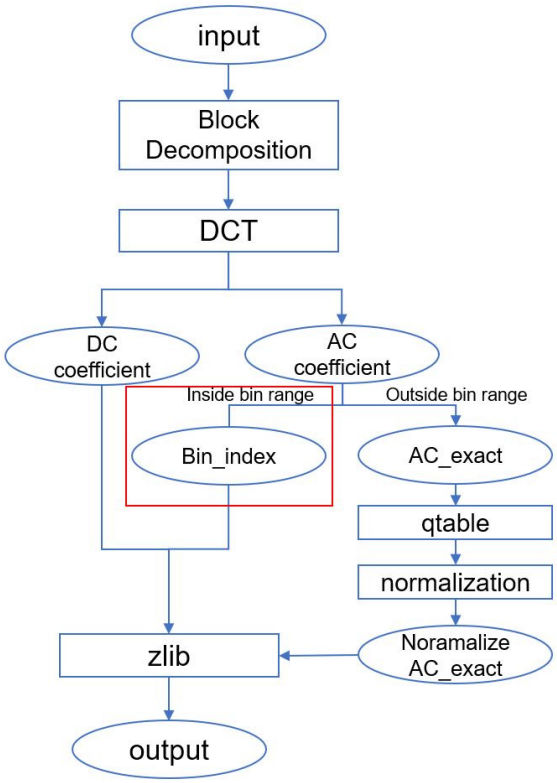


Fig. 2: The current implementation of the DCTZ framework.

C. Quantization and the Problem

Quantization maps a range of values, DCT coefficients in our case, to a small fixed one [22], and this is the mechanism DCTZ employed to implement truncation of the coefficient and reduce the size of the file. The DCTZ framework checks the above AC coefficients to see whether they are inside the bin range. To implement the error boundedness, DCTZ uses the number of bins (B) and the user-defined error bound (P) to determine specific bin ranges, defined as [-P*B, P*B]. Then, DCTZ divides the determined bin range into B small ranges, each with the size of 2*P. The AC coefficients that fall into any of these small bin ranges would be mapped to an integer with a value from 0 to 254, assuming the 1-byte

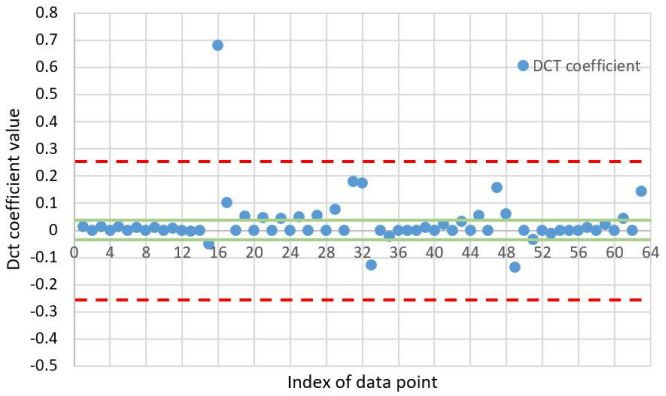


Fig. 3: The DCT coefficients for the first block of cellular-eint. This chart does not show DC as it is not quantized.

bin index representation in the compression process. DCTZ dedicates a bin index of 255 to represent the DC coefficient of each block and AC coefficients outside the bin range.

The size of the bin range is calculated once and fixed during both the compression and decompression processes. Therefore, no additional overhead for maintaining bin ranges. During decompression, DCTZ uses the center value of each small range to represent the original AC coefficients which are supposed to guarantee the user-defined error bound since the difference between the original AC coefficient and its reconstructed value would be smaller than P. Bin center values also require no additional overhead because they can be derivable from the bin ranges.

However, as we investigate the transform property of DCT, we found that as the magnitude of input data becomes bigger, the DCT result also follows this growing trend. To illustrate such behavior, let us consider Table I, which shows the first 10 data points of cellular-eint with different scaling factors. We also observe similar patterns in other data points in other data blocks. As we can see, Input 2 is about 10X bigger than Input 1, and the DCTZ coefficients of Input 2 are also about 10X bigger. Since the size of the bin range is fixed, the large input value will result in more DCT coefficients out of the bin range, thus lowering the compression ratio. As shown in Fig. 3, the number of DCT coefficient falls inside the bin range with error bound equals 1E-4 (between green line) is much smaller than 1E-3 (between red dash line). Therefore, decreasing the magnitude of the input value is considered to improve the compression ratio. However, it is not always true that smaller input values will result in better DCT performance due to the challenge of balancing the compression ratio and quality.

III. Proposed Approach

In this section, we propose our preconditioning method to control the range of the input data as well as the level offsetting mechanism to improve DCTZ in terms of guaranteeing the user-defined error bound while generating high compression ratios.

TABLE I: DCT result for the first 10 values of cellular-eint.

Input 1	DCT result 1	Input 2	DCT result 2
1.128948221903188	8.857954597	11.2894822190318	88.5795459719191
1.10622247854768	0.0012397052313	11.0622247854768	0.012397052313
1.0969064587706	-9.84E-15	10.9690645877069	-9.71E-14
1.09584639402647	0.0012202322376	10.9584639402647	0.012202322376
1.09997819840631	8.75E-15	10.9997819840631	8.4E-14
1.10599456853758	0.0011753396060	11.0599456853758	0.011753396060
1.11113078776719	-2.79E-14	11.1113078776719	-2.79E-13
1.11292744924759	0.0010894831660	11.1292744924759	0.010894831660
1.12894822132908	5.3E-88	11.2894822132908	5.3E-7
1.10622247772078	0.00092555927520	11.0622247772078	0.0092555927520

A. Level-Offsetting

Instead of directly applying DCT to the input value, we apply level-offsetting to the input data set. Specifically, we first find the mean value of the input dataset, which is equal to the sum of all input values divided by the number of input data points. Then find the center of the input data range, which is calculated as the sum of the min and max values divided by 2. Next, we calculate the offsetting factor as the center subtracted from the mean. Then this factor is subtracted from all of the input data. Algorithm 1, lines 1 to 10, shows the detail. This step will reduce the dynamic range requirements in the following DCT phase.

B. Scaling

To further control the range of the input data, we find the max value from the data set after the level offsetting and then take the ceiling value of the logarithm value of the max to calculate the scaling factor. We then divide every input value by this scaling factor. The detail of this part shows in Algorithm 1, lines 12 to 16. After this scaling phase, the transformed data is divided into blocks with a size of 64 and sent to the DCT step as before.

Algorithm 1 Detailed procedure of our proposed algorithm.

Input: D: input dataset. M: the number of data points.
Output: D': Offset data. S: Sum of the input value. ME: Mean value. CE: Center value. OF: Offsetting factor. D'': Preconditioned data. SF: Scaling factor.

1: **Level-Offsetting**:
2: **for** $D_m, m = 1, 2, \ldots, M$ **do**
3: $\quad S \leftarrow S + D_m$;
4: **end for**
5: $ME \leftarrow S/M$;
6: $CE \leftarrow (MAX(D) + MIN(D))/2$;
7: $OF \leftarrow CE - ME$;
8: **for** $D'_m, m = 1, 2, \ldots, M$ **do**
9: $\quad D'_m \leftarrow D_m - OF$;
10: **end for**
11:
12: **Scaling**:
13: $SF \leftarrow pow(10, ceil(log10(MAX(D'))) - 2)$;
14: **for** $D''_m, m = 1, 2, \ldots, M$ **do**
15: $\quad D''_m \leftarrow D'_m/SF$;
16: **end for**

C. Preconditioning Order

The order of applying level-offsetting and scaling will also impact the results. As shown in Fig. 4, the relative error of the first block of cellular-eint shows that applying level-offsetting first will result in a lower relative error than scaling first. This result indicates that level-offsetting first is a better way to guarantee user-defined error bound than scaling first. Since

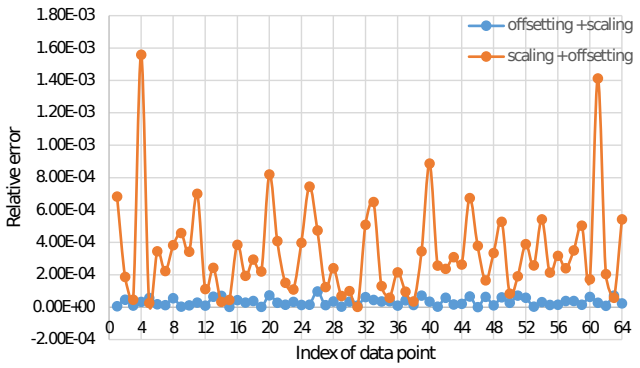


Fig. 4: The relative error for the first block of cellular-eint with different order of application between level-offsetting and scaling.

we observed this consistent performance on all the datasets we evaluated, we adopt this implementation in the rest of this paper.

IV. Experimental Evaluation

A. Evaluated Schemes

We evaluate the performance of DCTZ with our proposed method named DCTZ-P and the current implementation as DCTZ-C. We also compare it with two other state-of-the-art compressors, SZ (version 2.1.12) and ZFP (version 1.0.0). We ran SZ and ZFP with the command examples provided at Scientific Data Reduction Benchmarks (SDRBench) [23].

B. Datasets and Metrics

We evaluate the real-world scientific datasets (in double-precision) from HPC code packages FLASH [24], and Table II shows their characteristics. We focus on these datasets as DCTZ often suffers from guaranteeing user-defined error bound on those, which could prevent successful checkpoint/restart with lossy compression. We configured both Celluar and Sedov solvers as 2D (`-2d -auto` during solver setup) and picked the checkpoint files with the largest blocks. Therefore, each cellular dataset has 512 blocks, while 517 for sedov, with 2D (8 by 8) for each block. Note that FLASH uses adaptive mesh refinement, thus the number of blocks keeps evolving during simulation.

TABLE II: Characteristics of datasets evaluated.

Solver	Variable	Range	Mean	Dimension
cellular	dens	2.62E+7	21.7E+7	8*8*512
	eint	9.66E+17	8.42E+17	8*8*512
	pres	1.23E+25	8.07E+24	8*8*512
sedov	dens	4.2731	1	8*8*517
	eint	3.24E+2	25.94	8*8*517
	pres	3.90	0.82	8*8*517

The performance of the compressor is measured using several metrics:

- Compression Ratio (CR) is defined as the original data size divided by the compressed data size.
- Peak Signal-to-Noise Ratio (PSNR) measures the overall distortion between the original data and the decompressed data
- Bit-rate, means the average number of bits to represent a data point in a compressed file.

We set the error bound to 1E-3, 1E-4, and 1E-5, respectively. Error bound refers to the maximum relative error (calculated as the maximum difference between original data and reconstructed data divided by the range of the data) allowed. Since SZ and ZFP support several error-bounding mechanisms, we set appropriate runtime options in SZ and ZFP for a fair comparison.

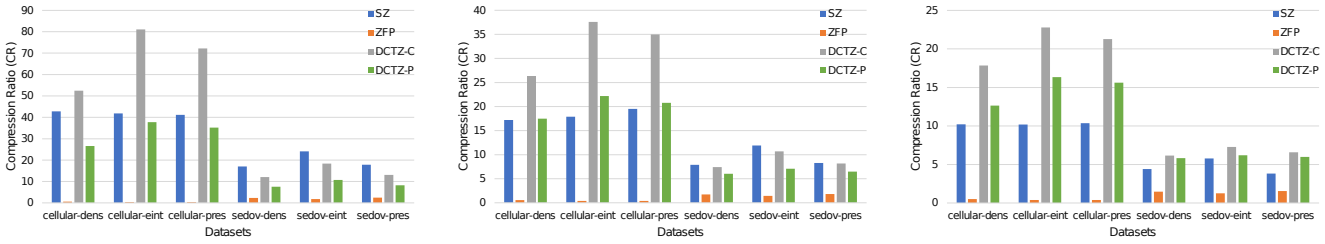
C. Experimental Results

1) Compression Ratio: In Fig. 5, we compare the compression ratio between DCTZ-P with SZ and ZFP with the error bound of 1E-3, 1E-4, and 1E-5. The chart shows that ZFP generates the lowest CR for all six datasets, as ZFP is among the most conservative lossy compression algorithm. With a relatively loose error bound (1E-3), SZ performs better than DCTZ-P. However, when the error bounds get tighter (like 1E-4), SZ outperforms DCTZ-P in 3 out of 6 cases (sedov-dens, sedov-eint, and sedov-pres). However, for an error bound equal to 1E-5 (the tightest), DCTZ-P achieves the highest CR on all six datasets.

We also notice in Fig. 5 that DCTZ-C achieves better CR than DCTZ-P in all cases (18 cases in total). However, as shown in Fig. 6, DCTZ-C cannot guarantee user-defined error bound in 8 cases. In the remaining 10 cases, DCTZ-C achieves as much as 97% better performance than DCTZ-P (cellular-dens with 1E-3), and the smallest gap is about 6% (sedov-dens with 1E-5). This difference indicates that our proposed precondition method provides a more conservative implementation of DCTZ.

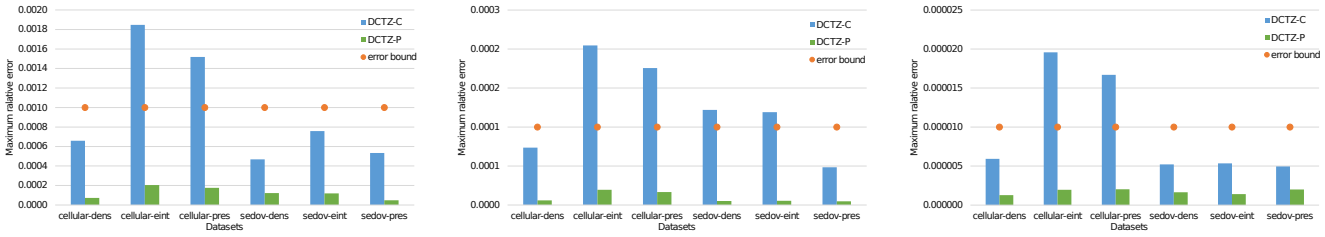
2) Error Boundedness: We compare the maximum relative error bound between DCTZ-C and DCTZ-P. As shown in Fig. 6, DCTZ-C shows a much higher maximum relative error than DCTZ-P in all cases. At the same time, DCTZ-C cannot guarantee the user-defined error bound for 2 cases (cellular-eint and cellular-pres) with 1E-3, 4 cases (cellular-eint, cellular-pres, sedov-dens, and sedov eint) with 1E-4 and 2 cases (cellular-eint and cellular-pres) with 1E-5. This figure also shows the trade-off between CR and compression error.

3) Rate Distortion: To evaluate how the preconditioning method would affect the performance of DCTZ, we compare the compression quality of DCTZ-P, DCTZ-C, SZ, and ZFP with bit-rate. From Fig. 7, we can observe that higher PSNR requires a higher bit-rate. In other words, the curve on the top left part with a higher slope is better than the curve on the bottom right with a lower slope. We can see that both DCTZ-C and DCTZ-P outperform SZ in all six datasets. An evaluation with a broader error-bound spectrum, such as 1E-2 (loosest) and 1E-6 (tightest), could make the curve look more informative as it provides more data points. However, we could not acquire a reasonable PSNR value of ZFP and SZ for some cases. Even with the error bound we tested, two



(a) error bound=1E-3 (b) error bound=1E-4 (c) error bound=1E-5

Fig. 5: The compression ratios for SZ, ZFP, and DCTZ-P with error bound of (a) 1E-3, (b) 1E-4, and (c) 1E-5.



(a) error bound=1E-3 (b) error bound=1E-4 (c) error bound=1E-5

Fig. 6: Maximum relative error for DCTZ-C and DCTZ-P with error bound of (a) 1E-3, (b) 1E-4, and (c) 1E-5.

curves are missing for the same reason (ZFP for cellular-eint in Fig. 7b and SZ for cellular-pres in Fig. 7c). Moreover, since ZFP can achieve a significantly higher PSNR but much lower CR than other compressors, its curve always falls on the right side of the coordinate.

TABLE III: The characteristics of variables.

var name	min	max	Var	Mean	Median
cellular-dens	1.00E+7	3.62E+7	4.28E+13	2.17E+7	2.4E+7
cellular-eint	1.63E+17	1.13E+18	1.32E+35	8.42E+17	1.03E+18
cellular-pres	8.64E+23	1.32E+25	1.53E+49	8.07E+24	9.76E+24
sedov-dens	0.0128	4.2859	0.5937	1	1
sedov-eint	2.5E-05	324.24	5.73E+3	26.94	2.50E-5
sedov-pres	1.00E-05	3.90	1.14	0.823	1.00E-5

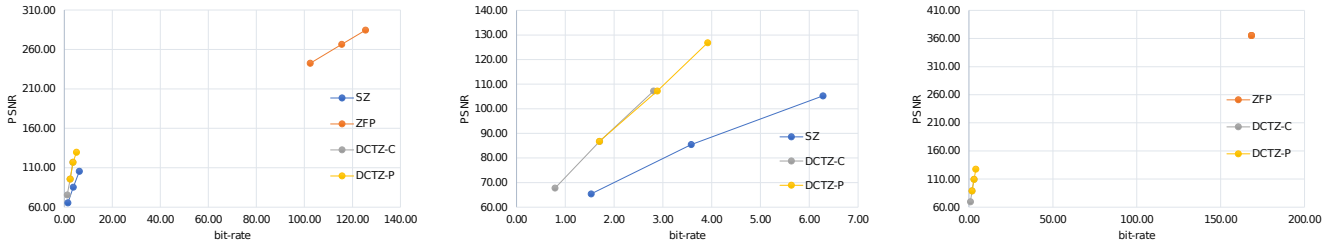
4) Data Characteristic: As shown in Fig. 8 and Table III, sedov-dens and sedov-eint contain more outliers (red "+") than sedov-pres, which introduces sharp discontinuities in the input dataset. In order to represent these sharp discontinuities accurately, in other words, to respect the user-defined error bound, we need a more precise high-frequency coefficient. Since our quantization method implements a fixed range proportional to the user-defined error bound, the DCTZ performance would degrade in terms of compression quality for the dataset with many outliers.

V. Related Work

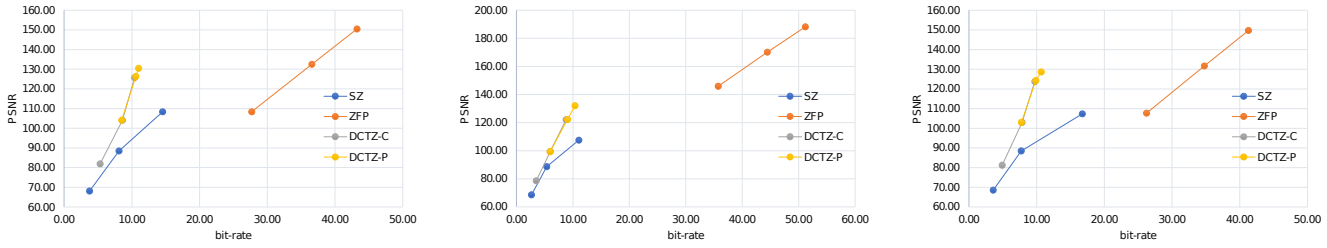
Recently, lossy compressors have attracted considerable attention across scientific datasets containing double and single-precision floating-point numbers. SZ [9]–[11] is a prediction-based compressor that exploits the predictability of the input data value. For each input data, it quantizes the difference between the predicted and actual value, and the difference is bounded inside the user-defined error bound. However, SZ tends to produce less competitive compression ratios with tighter error bounds. ZFP [12] is a transform-based lossy compressor that combines a decorrelation scheme with an embedded coding scheme. It reduces the file size by truncating the precision of the transformed coefficients based on user-defined error bounds. With the same level of error bounds, SZ usually achieves a higher compression ratio than ZFP, while SZ is often 20-30% slower than ZFP [25]. MGARD [26] is a multigrid-based lossy compressor that can bound the loss in different norms. The resulting loss can guide the adaptive reduction of the dataset to meet users' tolerance or memory constrain. FRaZ [27] is a fixed ratio lossy compression framework that respects user-specified error constraints. Unlike other lossy compressors, it can determine the appropriate error setting for different lossy compressors based on target compression ratios. Although it runs slower than fixed-error compression, it provides a new lossy compression technique for huge scientific datasets.

VI. Conclusion and Future Work

In this paper, we design a data preconditioning method for DCTZ, in which we apply level offsetting and scaling to control the magnitude of input data. Our proposed method, DCTZ-P, improves the compression quality while maintaining competitive performance compared to other state-of-the-art lossy compressors. Our experimental evaluations using six real-world scientific datasets show that DCTZ-P can achieve a

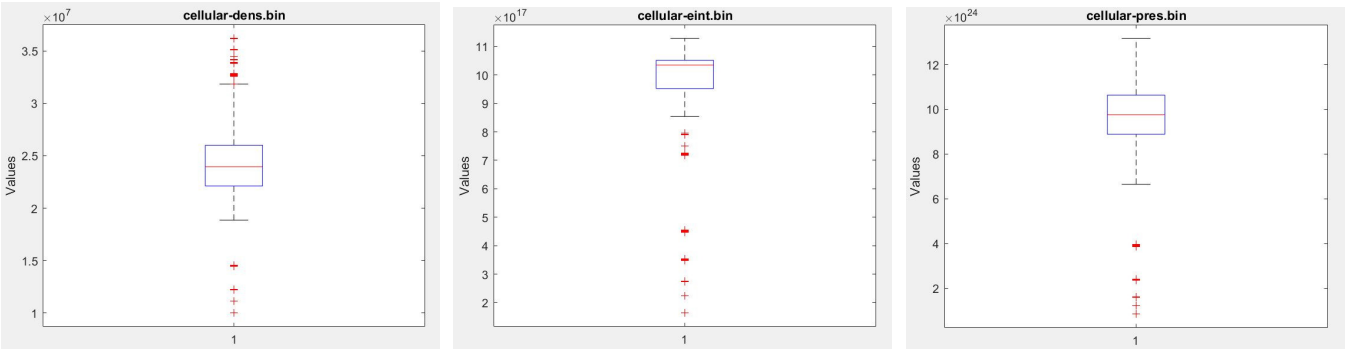


(a) cellular-dens (b) cellular-eint (c) cellular-pres

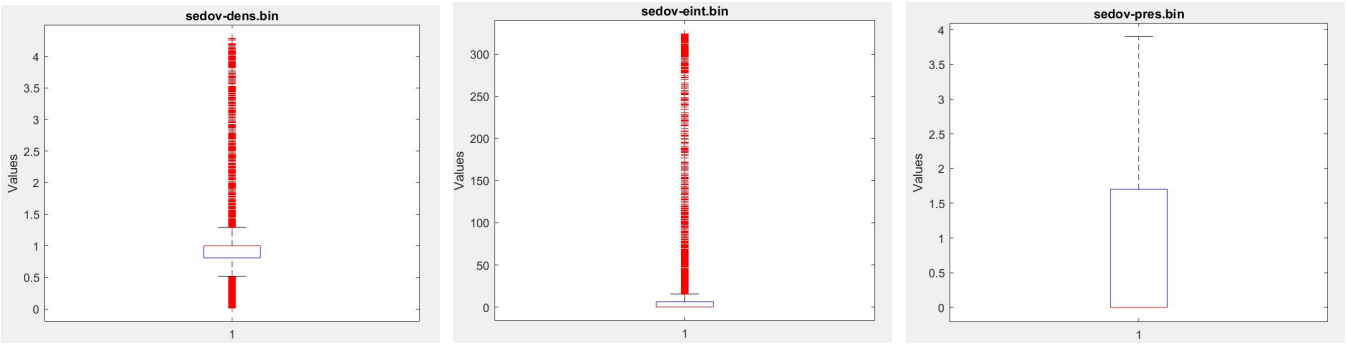


(d) sedov-dens (e) sedov-eint (f) sedov-pres

Fig. 7: Comparison of rate-distortion among different lossy compressors with six datasets.



(a) cellular-dens (b) cellular-eint (c) cellular-pres



(d) sedov-dens (e) sedov-eint (f) sedov-pres

Fig. 8: Distribution of input datasets displayed in box plots.

higher compression ratio than SZ and ZFP with a tighter error bound while guaranteeing the user-defined error bound.

In our future work, we plan to expand the evaluation of our method to other available scientific datasets [23] and the newest lossy compressors, such as SZ3 [28], in terms of compression ratio and throughput. We also plan to utilize the distribution pattern of the bin index data among blocks to improve the compression ratios further.

VII. Acknowledgment

This material is in part based upon work supported by the National Science Foundation under Grant No. 1751143.

References

[1] I. Foster, "Computing just what you need: Online data analysis and reduction at extreme scales," in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, 2017, pp. 306–306.

[2] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W. keng Liao, and A. Choudhary, "Data Compression for the Exascale Computing Era-Survey," *Supercomputing Fronties and Innovations*, vol. 1, no. 2, pp. 76–78, 2014. [Online]. Available: http://superfri.org/superfri/article/view/13

[3] D. Ibtesham, D. Arnold, K. B. Ferreira, and P. G. Bridges, "On the viability of Checkpoint Compression for Extreme Scale Fault Tolerance," *Berlin, Heidelberg: Springer Berlin Heidelberg*, pp. 302–311, 2012. [Online]. Available: http://dx.doi.org/10.100/978-3-642-29740-3_34

[4] X. Ni, T. Islam, K. Mohror, A. Moody, and L. V. Kale, "Lossy Compression for Checkpointing: Fallible or Feasible?" *Proceedings of the International Conference For High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

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

[6] D. S. Taubman and M. W. Marcellin, "JPEG 2000: Image Compression Fundamentals, Standards and Practice," *Norwell, MA, USA: Kluwer Academic Publishers*, 2001.

[7] X. Ni, T. Islam, K. Mohror, A. Moody, and L. V. Kale, "Isabela for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, pp. 523–540, 2013.

[8] 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.

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

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

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

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

[13] X. Zou, T. Lu, W. Xia, X. Wang, W. Zhang, H. Zhang, S. Di, D. Tao, and F. Cappello, "Performance optimization for relative-error-bounded lossy compression on scientific data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1665–1680, 2020.

[14] J. Zhang, X. Zhuo, A. Moon, H. Liu, and S. W. Son, "Efficient Encoding and Reconstruction of HPC Datasets for Checkpoint/Restart," in *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, 2019, pp. 79–91.

[15] J. Zhang, J. Chen, A. Moon, X. Zhuo, and S. W. Son, "Bit-Error Aware Quantization for DCT-based Lossy Compression," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1–7.

[16] K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, Applications," 1990.

[17] J. Zhang, A. Moon, X. Zhuo, and S. W. Son, "Towards improving rate-distortion performance of transform-based lossy compression for hpc datasets," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–7.

[18] ——, "Towards improving rate-distortion performance of transform-based lossy compression for hpc datasets," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–7.

[19] A. Moon, J. Kim, J. Zhang, and S. W. Son, "Lossy compression on iot big data by exploiting spatiotemporal correlation," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, 2017, pp. 1–7.

[20] X. Cai and J. S. Lim, "Algorithms for transform selection in multiple-transform video compression," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5395–5407, 2013.

[21] https://github.com/madler/zlib.

[22] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression," 1991.

[23] https://sdrbench.github.io/.

[24] Flash Center for Computational Science, "FLASH User's Guide: Version 4.6.2," 2019.

[25] X. Zou, T. Lu, W. Xia, X. Wang, W. Zhang, H. Zhang, S. Di, D. Tao, and F. Cappello, "Performance optimization for relative-error-bounded lossy compression on scientific data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1665–1680, 2020.

[26] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

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

[28] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, Z. Chen, and F. Cappello, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, pp. 1–14, 2022.