# Bit-Error Aware Quantization for DCT-based Lossy Compression

Jialing Zhang[1] Jiaxi Chen[1] Aekyeung Moon[2] Xiaoyan Zhuo[1] Seung Woo Son[1]
[1]*University of Massachusetts Lowell* [2]*ETRI*
Lowell, MA, USA Daegu, South Korea

***Abstract*—Scientific simulations run by high-performance computing (HPC) systems produce a large amount of data, which causes an extreme I/O bottleneck and a huge storage burden. Applying compression techniques can mitigate such overheads through reducing the data size. Unlike traditional lossless compressions, error-controlled lossy compressions, such as SZ, ZFP, and DCTZ, designed for scientists who demand not only high compression ratios but also a guarantee of certain degree of precision, is coming into prominence. While rate-distortion efficiency of recent lossy compressors, especially the DCT-based one, is promising due to its high-compression encoding, the overall coding architecture is still conservative, necessitating the quantization that strikes a balance between different encoding possibilities and varying rate-distortions. In this paper, we aim to improve the performance of DCT-based compressor, namely DCTZ, by optimizing the quantization model and encoding mechanism. Specifically, we propose a bit-efficient quantizer based on the DCTZ framework, develop a unique ordering mechanism based on the quantization table, and extend the encoding index. We evaluate the performance of our optimized DCTZ in terms of rate-distortion using real-world HPC datasets. Our experimental evaluations demonstrate that, on average, our proposed approach can improve the compression ratio of the original DCTZ by 1.38x. Moreover, combined with the extended encoding mechanism, the optimized DCTZ shows a competitive performance with state-of-the-art lossy compressors, SZ and ZFP.**

***Index Terms*—Lossy Compression, DCT, Quantization Table, Rate-Distortion.**

## I. Introduction

High-performance computing (HPC) systems are used in various domains for scientists to validate theories and investigate new phenomena in the scale, which was not conceivable in the past. Terabytes or even petabytes of analysis data would be easily produced by this process. For instance, climate scientists need to run large ensembles of high-fidelity simulations, estimating one ensemble member per simulated day may generate 260 TB of data every 16 seconds across the ensemble [1], [2]. Storing or transferring such big datasets may cause a huge overhead of storage space and I/O time. Applying compression techniques can reduce the size of data, thereby mitigating such overhead [3]–[6].

Lossy compression techniques, such as JPEG compression [7], are widely used for visualization data (i.e., image and video) as human eyes are less sensitive to minor changes (i.e., high frequencies) [8]. Recently, the consideration of lossy compression for scientific data has been increased. One main reason is that data need to be considerably compressed before transferring to the storage system because of the slow increase in storage bandwidth compared with other components in supercomputers [9].

While conventional lossy compressions, such as ISABELA [10] and FPZIP [11], are less capable of bounding errors, error-controlled lossy compression which is designed for domain scientists who have high compression precision demands, is coming into prominence. SZ [2], [12], [13] and ZFP [6] are two well-known error controllable lossy compressors designed for scientific data. SZ is a prediction-based compressor that has been widely evaluated in the scientific data compression community [14]–[16], showing a promising compression performance. However, the speed of SZ decreases largely when the error bound is tightened. ZFP is a transform-based compressor that combines a decorrelation scheme with an embedded coding scheme. ZFP is fast but hard to fix the compression ratio using its error-bound mode.

Motivated by image/video compression algorithms, a discrete cosine transform (DCT) based lossy compressor, namely DCTZ [17], [18], that combines two specific task-oriented quantizers, is designed for scientific data. DCTZ can achieve high compression ratios while guaranteeing specified error bounds, showing a comparable performance with SZ and ZFP. However, several areas for potential improvement exist as the quantization approach employed in DCTZ is still conservative (error introduced is much lower than the error bound) compared with ZFP [15] (also considered as a conservative lossy compressor).

In this work, we focus on the optimization of DCTZ. We first exploit the potential improvements in DCTZ by understanding its compression framework as well as its quantization and encoding process. We then present our proposed approach, namely DCTZ$^+$. Specifically, we design a bit-efficient quantizer that employs normalization of outbound high-frequency coefficients into small areas close to global bound. Moreover, we develop a unique ordering mechanism based on the quantization table to improve redundancy. We also extend the encoding index from 1-byte to 2-byte to further improve the compression ratio. To demonstrate the effectiveness of our optimization, we experiment on MIT Supercloud [19] using real-world scientific datasets, and compare the performance of DCTZ$^+$ with state-of-the-art lossy compressors, SZ and ZFP. Our experimental results show that our proposed mechanisms can improve the compression ratio of DCTZ by 1.38x on average, and shows a very competitive performance with SZ and ZFP in terms of rate-distortion.

Our implementation of the proposed approach is available at https://github.com/swson/DCTZ.

## II. Analysis of DCT-based Lossy Compressor: DCTZ

### A. DCTZ compression framework

State-of-the-art lossy compressors used for scientific data often apply strategies such as prediction, binary representation, data transform, energy compaction representation, vector quantization, and encoding. In DCTZ, an effective data transform, an error-controlled quantization, and an efficient encoding are implemented. The procedure is shown in Figure 1. DCTZ performs compression based on the following steps:

- Apply block decomposition to decorrelate data content. The input data is partitioned into small blocks with 64 datapoints (i.e., $8 \times 8$ in 2D and $4 \times 4 \times 4$ in 3D).
- Adopt a discrete cosine transform (specially DCT-II) on the block-based data for effective representation [20]–[22]. The first coefficient in each block is defined as DC coefficient and the remaining ones are AC coefficients.
- Quantize AC coefficients with either an error-controlled Quantizer-EC (EC) to guarantee maximum relative errors within specified bounds or a quantization table based Quantizer-QT (QT) to achieve high compression ratios within acceptable error rates.
- Use a customized encoding model to increase redundancies in bin indices and exploit lossless compressor (e.g., zlib [23]) for further compression.

### B. Quantization and Encoding Process in DCTZ

The quantization step is often used to compress a range of values to a single quantum value [24]. Rounding and truncation used in image compression (e.g., JPEG) are exemplars of quantization, but they introduce large errors which are hard to control. In DCTZ, as the DC coefficients (low frequency) are saved as is to preserve the most informative data, the quantization step thus plays an important role for AC coefficients (high frequency) on compression ratios and error control.

DCTZ has an adaptive quantization that includes two quantizers. EC is an error-controlled quantizer where the difference between an AC coefficient and its quantized value is *strictly* limited within a specified error bound. Specifically, EC is a uniform quantizer where AC coefficients that fall into the global bound (the range of quantization area) will be saved as its corresponding bin center value, and its represented bin index will be mapped to an integer with a value from 0 to 255 (1-byte unsigned char). The global bound ($GB$) is determined by the number of bins ($B$) and the specified error bound ($P$), which is defined as $[-P * B, P * B]$. This global bound is calculated once and fixed during the compression process.

To achieve higher compression ratios, DCTZ provides QT (an aggressive approach based on EC). Unlike EC, where the outbound (outside the global bound) AC coefficients will be saved as is, these coefficients in QT will be divided by a corresponding quantization table ($qt$). The $qt$ is generated

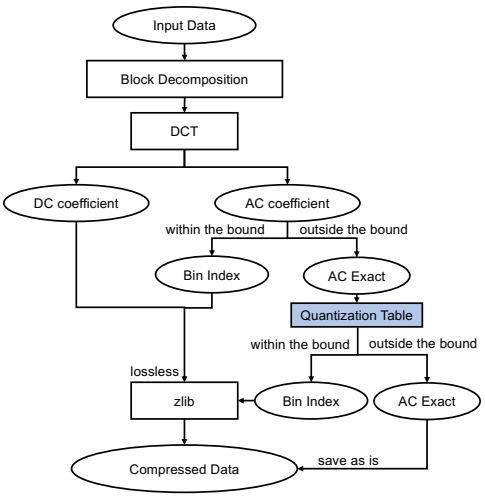



Fig. 1: Compression procedure (DCTZ with Quantizer-QT as an example, decompression is in reverse order).

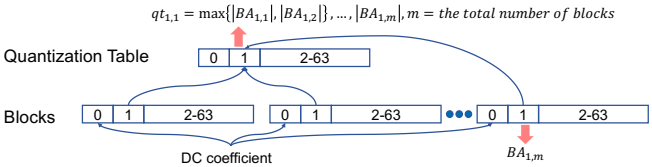



Fig. 2: The procedure of generating quantization table.

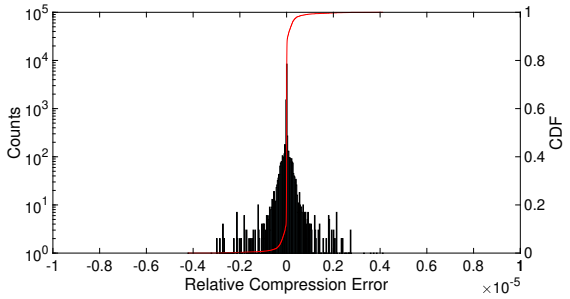



Fig. 3: The distribution of relative compression errors, with the error bound of 1E-5. The primary y-axis (left) shows the histogram and the secondary y-axis (right) shows the cumulative distribution function (CDF).

by finding the maximum value of the $n^{th}$ AC coefficient over all blocks as shown in Figure 2. This way, some of the AC coefficients will be mapped into the global bound and represented by the bin index as EC does. For the remaining AC coefficients that are still outside the global bound, we save them as is. As a result, QT can achieve higher compression ratios than EC but introduces extra compression errors.

However, due to the transform property of DCT [17], even with QT, DCTZ is still conservative regarding to the accuracy requirement. Figure 3 shows the distribution of compression error (relative compression error between the original data and decompressed data). As shown in the figure, most of the compression errors are smaller than 4E-6, which is about 2.5x smaller than the specified error bound of 1E-5. Also, from the cumulative distribution function (CDF), we can see that

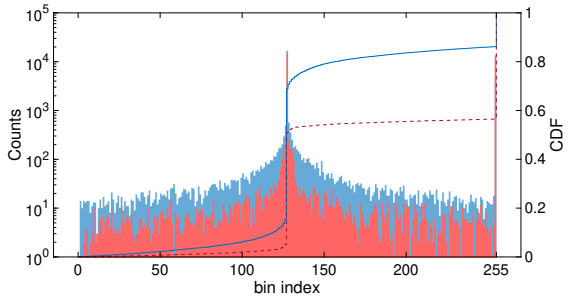



Fig. 4: The distribution of bin index, with error bounds of 1E-3 (blue line) and 1E-5 (red line), respectively. The primary y-axis shows the histogram and the secondary y-axis shows the cumulative distribution function (CDF).

most compression errors are centered around zero, showing a potential improvement in the quantization.

Moreover, we investigate the potential improvements in the encoding mechanism of DCTZ. Figure 4 presents the distribution of the bin index. For each outbound AC coefficient, we assign it with the index of 255; for each inbound one, we assign it using equal-width-binning. As shown in the figure, when the error bound gets tight, the number of coefficients falling within the global bound gets reduced, so do the compression ratios. Therefore, increasing extra redundancy is needed in the encoding mechanism to improve the compression ratio further.

## III. Proposed Approach

In this section, we propose our optimized quantization model as well as ordering and extended encoding mechanism to improve the performance of DCTZ, particularly under QT.

### A. Proposed Quantization Model

Unlike the original DCTZ with QT that applied the quantization table ($qt$) to map the outbound AC coefficients into global bound (as shown in Figure 5a), we develop a bit-efficient quantizer that maps those coefficients into small areas close to the global bound. We achieve this by including a normalization process after the quantization table step.

Specifically, we first divide the outbound AC coefficients by $qt$ so that they will be mapped into the area of [-1, 0) or (0, 1]. Then we multiply them by the specified error bound ($P$), thus ensuring these outbound AC coefficients are still within [-$P$, 0) or (0, $P$]. If the mapped value falls in the area [-$P$, 0), we increment it by the lower global bound ($GB\_min$); otherwise, we increment it by upper global bound ($GB\_max$). As a result, these outbound AC coefficients will be mapped into two small areas, as shown in Figure 5b. The detailed algorithm is illustrated in Algorithm 1. Figure 5 shows the distribution of the DCT block coefficient. The x-axis range is from 1 to 64, which represents the index of the 64 coefficients, whereas the y-axis represents each DCT coefficient value.

Next, we employ zlib on these outbound coefficients for further compression. Since they are mapped from a relatively large range into a small one closer to the global bound, it makes zlib much easier to compress as the normalization increases redundancy in the AC coefficients. Note that this approach would not introduce any extra compression error as no additional coefficient will be mapped into the global bound and saved as its approximated value.

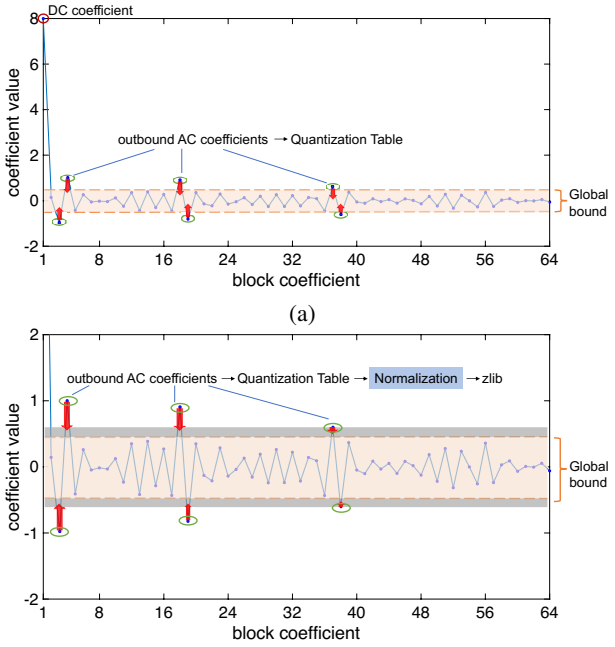



Fig. 5: An example of the distribution of DCT block coefficients (after applying transform with block size of 64) and its compression procedure in (a) original DCTZ with QT. (b) proposed DCTZ with QT.

**Algorithm 1** The normalization process applied to the outbound AC coefficients after divided by quantization table

**Input:** $BA$: Outbound AC block coefficients. $qt$: quantization table. $M$: the number of blocks. $N$: the number of block coefficient in each block, also the number of $qt$'s coefficient. $GB$: the global bound. $P$: a user-defined error bound.
**Output:** $BA'$: approximated coefficients.

```
for n = 1, 2, ..., N do
    for m = 1, 2, ..., M do
        if BAn,m > GB_max then
            BAn,m' ← (BAn,m / qtn,1) * P + GB_max
        else if BAn,m < GB_min then
            BAn,m' ← (BAn,m / qtn,1) * P + GB_min
        end if
    end for
end for
```

**Algorithm 2** The algorithm of bin index ordering.

**Input:** $C$: bin index. $qt$: quantization table. $M$: the number of blocks. $N$: the number of block coefficient in each block.
**Output:** $ax$: array index of the $qt$. $SC$: sorted bin index based on $ax'$. $C'$: bin index after ordering

```
ax←sort(qt)
for m = 0, 1, ..., M − 1 do
    for n = 1, 2, ..., N do
        ax'_{mN+n} ← ax_n + mN
    end for
end for
SC←sort(C) based on ax'
for n = 1, 2, ..., N do
    for m = 0, 1, ..., M − 1 do
        C'_{(n−1)M+m+1} ← SC_{mN+n}
    end for
end for
```

### B. Ordering Mechanism

To further improve the compression ratio, in particular, the bin indices, we develop a unique ordering step in the QT encoding mechanism. We take advantage of the consistent distribution pattern exhibited among block coefficients in the same dataset [17]. In other words, we order the block index based on the descending order of $qt$ before applying to zlib. By doing this, we cluster indexes that are close together to increase the redundancy. The detailed algorithm is illustrated in Algorithm 2. Since the size of the block is fixed, the theoretical complexity of the ordering step is $O(n)$, which is faster than sorting the whole block index with the complexity of $O(nlogn)$.

### C. Extended Data Encoding

From Section II-B, we observe that the number of inbound AC coefficients impacts the compression ratios of DCTZ, and the size of the global bound is determined by the number of bin indices and the user-defined error bound. Therefore, allocating more bin indices will result in a larger global bound, thus potentially a higher compression ratio. In the original DCTZ, we use 1-byte (i.e., unsigned char) to represent a bin index such that the range of the bin index is from 0 to 255. For example, if the error bound is set to 1E-3, the global bound is [-0.255,0.255]. If we use 2-byte (i.e., unsigned short) to represent a bin index, the range of the bin index would be from 0 to 65535, and the global bound would be [-65.535, 65.535] when the error bound is 1E-3. Since bin indices are all compressed using zlib in the optimized DCTZ, we employ the 2-byte bin index option to improve compression ratios further. We note that extending to 2-byte is sufficient as the majority of AC coefficients are covered by the increased global bound.

## IV. Experimental Evaluation

### A. Experiment Setup

We conduct our experiments on MIT Supercloud with two Intel Xeon Gold 6248 processors and a total of 384 GB of memory. We compare our DCTZ$^+$ (optimized DCTZ) with DCTZ (version 0.1) as well as two state-of-the-art methods, SZ (version 2.0) and ZFP (version 0.5). We use Z-checker [25], a framework for assessing lossy compression of scientific data, to evaluate the performances. We analyze our compression approach using the following schemes:

- **DCTZ_QT**: DCTZ-0.1 with Quantizer-QT.
- **DCTZ_QT$^+$** & **DCTZ_QT$^+$_2bytes**: optimized DCTZ with proposed bit-efficient quantizer (QT based) using 1-byte and 2-byte indexing, respectively.

### B. Datasets and Metrics

We perform the evaluation with six real-world scientific datasets (all in double-precision floating-point) from three HPC code packages: FLASH [26], CMIP5 [27], and Nek5000 [28], as summarized in Table I.

The quality of lossy compression approaches is measured with several metrics: Compression Ratio (CR), which is defined as the original data size divided by the compressed data size; Peak Signal-to-Noise Ratio (PSNR), which measures the overall distortion between the original data and the decompressed data (expressed in terms of logarithmic decibel scale); Bit-rate, which refers to the average number of bits used to represent a data point after the compression. For a fair comparison, we set the error bound at 1E-3, 1E-4, and 1E-5. Here, the error bound refers to the maximum relative error (calculated as the maximum absolute error divided by the value range of the data). To assess the overall compression quality, we use rate-distortion (PSNR vs. bit-rate), a critical metric used for evaluating the quality of compressed data.

TABLE I: Dataset and its characteristics.

| Code | Dataset | Value Range | Avg Value | Dimension |
|---|---|---|---|---|
| FLASH | sedov | 4.239 | 1.000 | 8×8×485×154 |
| | cellular | $2.648E^7$ | $2.208E^7$ | 8×8×512×295 |
| CMIP5 | rlds | 361.230 | 285.884 | 144×90×100 |
| | mrsos | 44.500 | 7.692 | 144×90×100 |
| Nek5000 | eddy | 4.835 | $3.237E^{-8}$ | 256×8×8×1×999 |
| | vortex | 0.055 | 0.002 | 140×8×8×8×99 |

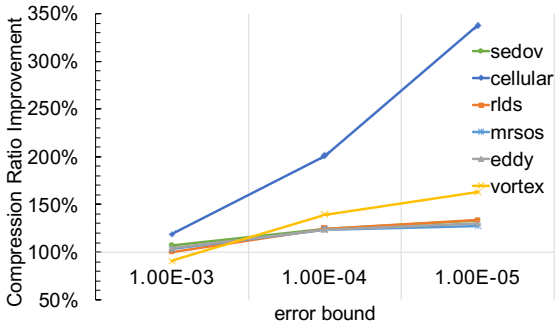
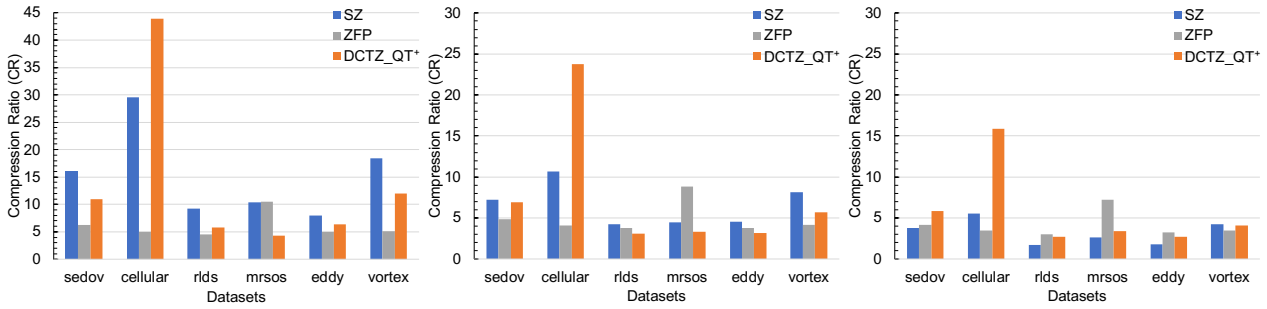



Fig. 6: The performance improvement of DCTZ_QT$^+$ compared with original DCTZ_QT in terms of compression ratio, with error bounds of 1E-3, 1E-4, and 1E-5.

### C. Evaluation Results

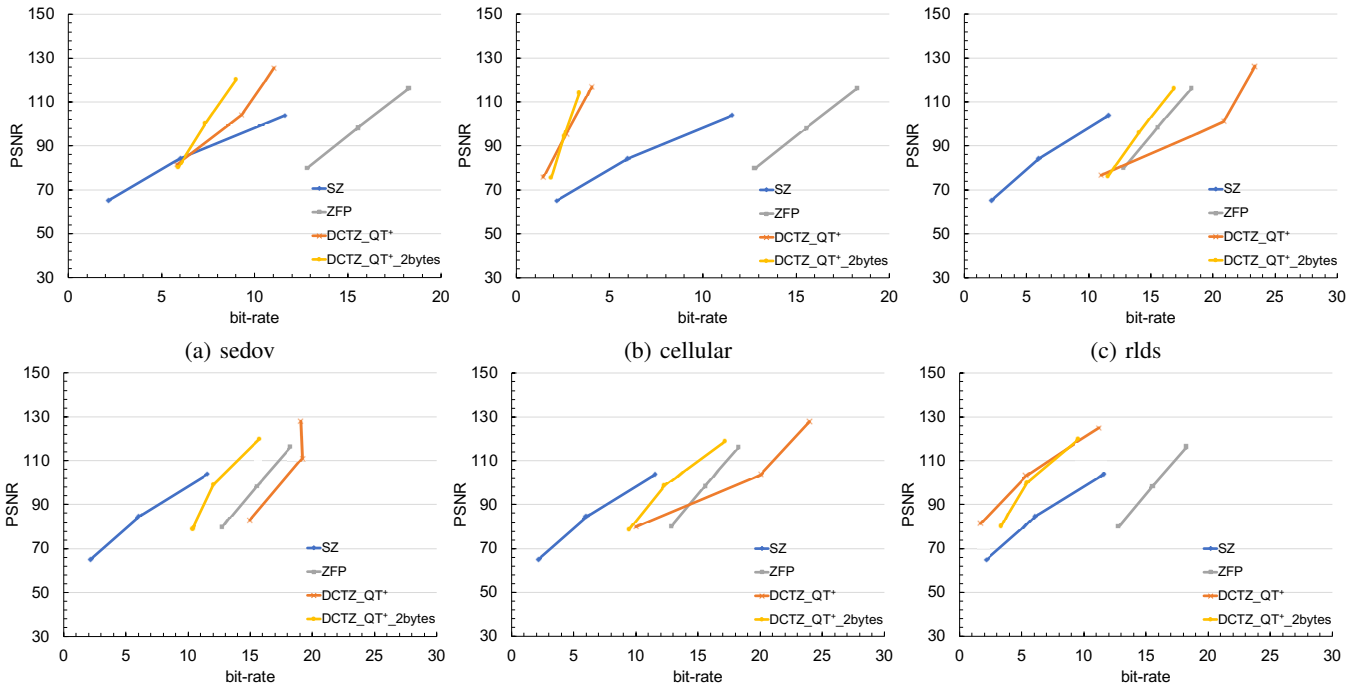
*a) Compression Ratio & Error Bound:* Figure 6 shows the improvements in compression ratio (CR) brought by the proposed approach (DCTZ_QT$^+$) compared with the original DCTZ_QT, with error bounds of 1E-3, 1E-4, and 1E-5. As shown in the figure, DCTZ_QT$^+$ shows a better compression ratios than DCTZ_QT on evaluated datasets, except for one case: vortex with error bound of 1E-3. Especially for cellular, the CRs of DCTZ_QT$^+$ are 119%, 201%, and 338%, respectively, higher than DCTZ_QT. On average, the compression ratio of our proposed approach is 138% higher than the original one. We also observe an overall trend that DCTZ_QT$^+$ generates improved CR with stricter error bound on all six evaluated datasets. This demonstrates that our bit-efficient quantizer reduces those save-as-is coefficients and improves the space efficiency of bin indices on outbound AC coefficients.

We next present the compression ratios of SZ, ZFP, and DCTZ_QT$^+$, as shown in Figure 7. We first compare our compressor with SZ. As shown in the figure, DCTZ_QT$^+$ shows the highest CRs on cellular for all three error bounds we




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

Fig. 7: The compression ratios for SZ, ZFP and DCTZ_QT+ with the error bounds of (a) 1E-3, (b) 1E-4 and (c) 1E-5.




(a) sedov (b) cellular (c) rlds

(d) mrsos (e) eddy (f) vortex

Fig. 8: Comparison of rate-distortion using different lossy compression methods on selected datasets with different error bounds.

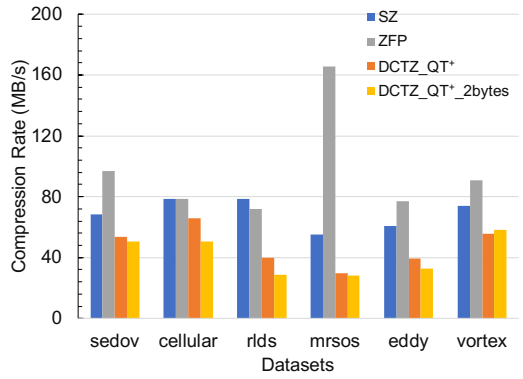
evaluated. When error bound is tight (e.g., 1E-5), DCTZ_QT+ achieves higher CRs than SZ, while SZ shows better performance when error bound is relatively loose (e.g., 1E-3). We then compare our approach with ZFP. We observe that DCTZ_QT+ achieves higher CRs than ZFP on five datasets (sedov, cellular, rlds, eddy, and vortex) when error bound is set to 1E-3, while ZFP shows higher CRs on datasets rlds, mrsos and eddy when error bound is set to 1E-4 and 1E-5.

*b) Rate Distortion & Indexing:* To illustrate how our quantization and extended indexing mechanism would affect the performance of our algorithm, we compare overall compression qualities of SZ, ZFP, DCTZ_QT+, and DCTZ_QT+_2bytes. Figure 8 presents the rate-distortion of different compressors. From the figure, we can observe that higher PSNR usually needs a higher bit-rate, thus the curve on the upper left part with a higher positive slope is better than the ones on the lower right with a lower positive slope. We can see that DCTZ_QT+_2bytes outperforms DCTZ_QT+ (1-byte by default) on most evaluated datasets, showing a competitive performance, in particular, on sedov, cellular and vortex when comparing with SZ, and outperforms ZFP. As discussed in Section III-C, increasing the number of bin indices will enlarge the global bound. Hence, mapping more AC coefficients into the global bound results in a higher compression ratio (i.e., lower bit-rate). We also notice that different datasets show different compression qualities, which indicates that the design of lossy compression largely depends on the characteristics of the dataset as well as the error bound.

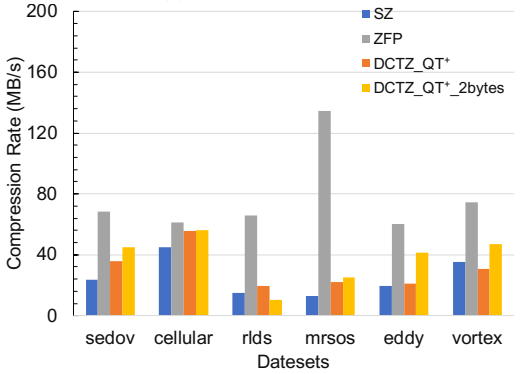
Table II compares the percentage of AC coefficient mapped into the global bound using 1-byte and 2-byte for bin indices. As we can see, more AC coefficients are mapped into the bound with 2-byte indexing, which explains why DCTZ_QT+_2bytes outperforms DCTZ_QT+ in terms of compression ratio. Since we use the bin center value to represent

TABLE II: The percentage (%) of AC coefficient that mapped into the global bound.

| Indexing | sedov | | | cellular | | | rlds | | | mrsos | | | eddy | | | vortex | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1E−3 | 1E−4 | 1E−5 | 1E−3 | 1E−4 | 1E−5 | 1E−3 | 1E−4 | 1E−5 | 1E−3 | 1E−4 | 1E−5 | 1E−3 | 1E−4 | 1E−5 | 1E−3 | 1E−4 | 1E−5 |
| 1-byte | 86.3 | 67.9 | 56.5 | 95.9 | 82.8 | 59.8 | 84.6 | 31.6 | 6.7 | 57.3 | 29.2 | 25.4 | 74.7 | 30.3 | 7.0 | 81.2 | 53.3 | 25.1 |
| 2-byte | 100 | 98.8 | 93.4 | 100 | 98.4 | 97.5 | 100 | 98.4 | 93.7 | 100 | 98.9 | 80.5 | 100 | 99.6 | 89.8 | 100 | 99.6 | 91.4 |




(a) error bound=1E-3




(b) error bound=1E-5

Fig. 9: Comparison of compression rate with error bounds of 1E-3 and 1E-5.

AC coefficient during decompression, extra compression error would be introduced in 2-byte indexing. Thus, there is a trade-off between bin indexing selection and achievable compression ratios.

*c) Compression Rate & Error Bound:* Figure 9 presents the average compression rate (excluding disk I/O) on all evaluated datasets using SZ, ZFP, DCTZ_QT+, and DCTZ_QT+_2bytes, with error bounds of 1E-3 and 1E-5. ZFP is overall the fastest due to its efficient mechanism (transform-based and fixed-rate representation). We also observe that our approach outperforms SZ when error bound is set to 1E-5, while SZ is faster than our approach when error bound is set to 1E-3. Based on all evaluated metrics so far, we can conclude that DCTZ_QT+ outperforms SZ in terms of compression ratio and compression rate with a tight error bound (e.g., 1E-5) on evaluated datasets. SZ, on the other hand, shows better performance with a less tight error bound (e.g., 1E-3).

## V. Related Work

Applying lossy compression on scientific data (single and double-precision floating-point numbers) is becoming popular. SZ [2], [12], [13] is a prediction-based compressor that relies on the prediction for the decorrelation stage. For each value, SZ quantizes the difference between the predicted and actual value using a linear scale, and each difference is strictly limited in a user-set error bound. ZFP [6] is a transform-based compressor that combines a decorrelation scheme with an embedded coding scheme. It fixes the compression rate by truncating the precision of transformed coefficients based on the error bound. With the same error bound setting, ZFP is usually faster than SZ (about 3x more), while SZ often has more than 2x higher compression ratios than those of ZFP [29]. DCTZ [17], [18], [30] is a DCT-based lossy compressor that is designed to obtain high compression quality in both speed and compression ratios. It can seamlessly work for checkpoint/restart on evaluated application but is conservative on error control. MGARD [31] is a multigrid based error-controlled lossy compressor that provides different norms to control data distortion. It is particularly applicable to the case of turbulence modeling and has guaranteed and computable bounds on the errors generated by the reduction. Unlike the other error controllable compressors, FRaZ [32] is a lossy compression framework that provides high-fidelity fixed-ratio for scientific data. It can identify the optimum error setting of selected lossy compressors. Although slower than fixed error compression, it has lower runtime for very large scientific floating-point datasets.

## VI. Conclusion

In this work, we improve the performance of DCT-based compressor by optimizing the quantization model as well as the encoding process. We design a bit-efficient quantizer based on the framework of DCTZ, develop a unique ordering mechanism based on the quantization table, and extend the encoding index. We conduct the experiment using real-world scientific datasets and compare the performance of optimized DCTZ (DCTZ+) with state-of-the-art lossy compressors, SZ and ZFP. Our experiments show that DCTZ+ can improve the compression ratio of DCTZ by 1.38x on average, and the extended bin indexing combined with the proposed quantizer shows a very competitive performance with SZ and ZFP, especially when error bound is tight.

In our future work, we plan to improve DCTZ with dynamic bin indexing for different datasets. We also plan to apply non-uniform quantizer rather than equal-width binning to make the reconstructed data more accurate.

## Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No.1751143. The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing (HPC, database, consultation) resources that have contributed to the research results reported within this paper.

## 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] 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.

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

[4] S. 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, January 2014.

[5] 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, 2012, pp. 302–311. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29740-3_34

[6] P. Lindstrom, "Fixed-Rate Compressed Floating-Point Arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, Dec 2014.

[7] D. Taubman and M. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated, 2013.

[8] S. Wahlstrom, "Optimising JPEG image compression by identifying image characteristics," Master's thesis, VU University Amsterdam and Mälardalens University, 2015.

[9] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong, "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. [Online]. Available: https://doi.org/10.1177/1094342019853336

[10] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C. S. Chang, S. Klasky, R. Latham, R. B. Ross, and N. F. Samatova, "Isabela for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, pp. 524–540, 2013.

[11] P. Lindstrom and M. Isenburg, "Fast and Efficient Compression of Floating-Point Data," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1245–1250, 2006.

[12] S. Di and F. Cappello, "Fast Error-Bounded Lossy HPC Data Compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 730–739.

[13] 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 *Proceedings of the 31th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 2017.

[14] 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," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2019. [Online]. Available: http://dx.doi.org/10.1145/3295500.3356155

[15] P. Lindstrom, "Error distributions of lossy floating-point compressors," 10 2017.

[16] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne, "Toward a multi-method approach: Lossy data compression for climate simulation data," in *High Performance Computing*, J. M. Kunkel, R. Yokota, M. Taufer, and J. Shalf, Eds. Cham: Springer International Publishing, 2017, pp. 30–42.

[17] J. Zhang, X. Zhuo, A. Moon, H. Liu, and S. W. Son, "Efficient Encoding and Reconstruction of HPC Datasets for Checkpoint/Restart," in *35th International Conference on Massive Storage Systems and Technology*, 2019.

[18] J. Chen, "Optimizing DCT-based Lossy Compression for Scientific Datasets," Master's thesis, University of Massachusetts Lowell, 2020.

[19] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. S. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive supercomputing on 40, 000 cores for machine learning and data analysis," *CoRR*, vol. abs/1807.07814, 2018. [Online]. Available: http://arxiv.org/abs/1807.07814

[20] 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.

[21] 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)*, Sep. 2017, pp. 1–7.

[22] X. Cai and J. S. Lim, "Algorithms for transform selection in multiple-transform video compression," *Trans. Img. Proc.*, vol. 22, no. 12, p. 5395–5407, Dec. 2013. [Online]. Available: https://doi.org/10.1109/TIP.2013.2284073

[23] M. A. Greg Roelofs, "zlib," https://github.com/madler/zlib, 2017.

[24] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. USA: Kluwer Academic Publishers, 1991.

[25] D. Tao, S. Di, H. Guo, Z. Chen, and F. Cappello, "Z-checker: A framework for assessing lossy compression of scientific data," *CoRR*, vol. abs/1707.09320, 2017. [Online]. Available: http://arxiv.org/abs/1707.09320

[26] Flash Center for Computational Science, "FLASH User's Guide: Version 4.4," 2016.

[27] G. A. Meehl, C. Covey, B. McAvaney, M. Latif, and R. J. Stouffer, "Overview of the Coupled Model Intercomparison Project," *Bulletin of the American Meteorological Society*, vol. 86, no. 1, pp. 89–93, 2005.

[28] P. Fischer, J. Lottes, S. Kerkemeier, O. Marin, K. Heisey, A. Obabko, E. Merzari, and Y. Peet, "Nek5000 User Documentation," Argonne National Laboratory, Tech. Rep. ANL/MCS-TM-351, 2015.

[29] 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.

[30] "DCTZ," https://github.com/swson/DCTZ, 2019.

[31] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the unstructured case," *in preparation*, dec 2018.

[32] 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," 2020.