Efficient Encoding and Reconstruction of HPC Datasets for Checkpoint/Restart

Jialing Zhang Xiaoyan Zhuo Aekyeung Moon Hang Liu Seung Woo Son
Department of Electrical and Computer Engineering
University of Massachusetts Lowell
Lowell, MA, USA

***Abstract*—As the amount of data produced by HPC applications reaches the exabyte range, compression techniques are often adopted to reduce the checkpoint time and volume. Since lossless techniques are limited in their ability to achieve appreciable data reduction, lossy compression becomes a preferable option. In this work, a lossy compression technique with highly efficient encoding, purpose-built error control, and high compression ratios is proposed. Specifically, we apply a discrete cosine transform with a novel block decomposition strategy directly to double-precision floating point datasets instead of prevailing prediction-based techniques. Further, we design an adaptive quantization with two specific task-oriented quantizers: guaranteed error bounds and higher compression ratios. Using real-world HPC datasets, our approach achieves 3x–38x compression ratios while guaranteeing specified error bounds, showing comparable performance with state-of-the-art lossy compression methods, SZ and ZFP. Moreover, our method provides viable reconstructed data for various checkpoint/restart scenarios in the FLASH application, thus is considered to be a promising approach for lossy data compression in HPC I/O software stacks.**

***Index Terms*—Lossy Compression, Checkpoint/Restart, DCT**

I. Introduction

HPC applications periodically produce extremely large amounts of data, mainly for snapshotting their states for possible failure/restart and post-simulation data analysis [1], [2]. Storing these raw data generated on supercomputers incurs an excessive overhead of storage space and I/O time. For example, a total of 170 terabytes of CESM (Community Earth System Model) data is being produced for CMIP5 (Coupled Model Intercomparison Project), and multiple petabytes of data will be generated for the upcoming CMIP6 experiments [3], [4] per entire run. At the same time, however, a supercomputer like Yellowstone [5] has only tens of petabytes of centralized file system and data storage, with less than 100 GB/s of aggregated I/O bandwidth.

Compression techniques can help mitigate the burden during the data I/O phase by reducing checkpoint data size, thereby shortening the checkpoint (I/O) time [6]–[9]. Traditional lossless compression techniques, while preserving 100% of data fidelity, are not able to achieve appreciable data reduction on floating-point scientific data [10]–[15]. For example, the lossless compressors used in MCRENGINE [14], ISOBAR [15], and Welton et al. [13] achieved only about 1.15x–1.6x compression ratios which do not meet the demanding datasize reduction requirement. Lossy compression, which has been widely used in image, video, and audio compression, can realize higher compression ratios than lossless ones. To date, lossy compression has not been widely adopted in the scientific domain for several reasons: scientific domain knowledge is required for the design of efficient lossy compressors, high compression precision on scientific data is demanded, and the viability of reconstructed data from lossy compressors in scientific workflows such as checkpoint/restart is not well quantified.

On the other hand, lossy compression, which has been widely used in image, video and audio compressions, can bring a higher compression ratio than lossless one. Though it will introduce errors which are not easy to bound, recent studies have found that scientific data can actually tolerate some error-bounded loss in their simulation data accuracy [6]. Moreover, errors, to some extent, are inherent in scientific simulations where they can be generated from inaccurate scientific sensors [16]–[19]. Thus, applying lossy compression can mitigate the overall overhead faced by today's HPC systems.

Existing lossy compression techniques used for scientific data often apply strategies such as prediction, binary representation, data transform, and vector quantization. Di and Cappello [20] proposed a technique called SZ, where predictable data were represented based on several curve-fitting models and unpredictable datasets were compressed using a binary representation analysis. Tao et al. [20], [21] extended SZ by employing an adaptive quantization mechanism to improve the accuracy of their prediction-based compression algorithm. The limitation of the prediction-based compressors is that they were data dependent. The compression performance would degrade if the simulation data exhibited less structure [22], [23]. Lakshminarasimhan et al. [24] proposed a technique called ISABELA, where the B-Spline transformation was applied on sorted data. However, the compression ratio of ISABELA was limited due to the sorting process. NUMARCK [25] adopted quantization mechanisms on the change ratios between consecutive checkpoints, but it could not ensure compression error within the bounds. Yuan et al. [26] presented a parallelized version of [25], but it had a drawback of its large memory requirement.

The above-mentioned challenges motivate us to design a lossy compressor that is efficient, error bounded, able to achieve high compression ratios, and viable for check-

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

point/restart. In this work, we propose a novel block decomposition strategy and combine it with the well-known discrete cosine transform (DCT) on double-precision floating point datasets directly. The reason for choosing DCT is because it has high decorrelation efficiency [9], [27], and its inverse has the same spectrum as the original data [28]. The major contributions of our approach presented in this paper are described as follows:

1) With the proposed novel block decomposition strategy directly applied to floating-point numbers, the modified transform-based technique not only efficiently decorrelates data content but also eliminates potential computational overheads, such as sorting or ordering coefficients, while achieving high compression ratios. Combining DCT with customized decomposition, which improves the ability of DCT to distinguish dominant coefficients, and quantization, we were able to achieve comparable compression ratios with high efficiency.

2) An adaptive quantization customized with our decomposed DCT coefficients is designed, in which users can select the compressor modes based on their task requirements: either guaranteed maximum relative errors within specified bounds (by an error-controlled Quantizer-EC) or high compression ratios within acceptable error rates (by a quantization table based Quantizer-QT). Strict error control in original (spatial) domain is achieved in transform-based lossy compressors. The quantizer is determined for each checkpoint by learning the distribution of high-frequency DCT coefficients. We also create a customized encoding model that exploits increased redundancies in bin indexes by our quantization table designed for block-based DCT coefficients, improving compression ratios further.

3) We compare our compressor with state-of-the-art compressors, SZ-1.4 and ZFP, using six scientific datasets from three real scientific applications, FLASH [29], CMIP5 [30], and Nek5000 [31], [32]. Our experimental results demonstrated that our compression approach achieves comparable performance with regard to compression ratio, compression accuracy (Maximum Relative Error, Normalized Root-Mean-Square Error (NRMSE), the Peak Signal-to-Noise Ratio (PSNR), and Pearson correlation) and compression speed. On the evaluated data, our compressor achieves 3x–38x compression ratios while ensuring user-specified error bounds.

4) Lastly, we reconstruct the data from two solvers (Sedov and Cellular) in FLASH, and investigate them under several checkpoint/restart scenarios. It is shown that restarts from lossy state are viable without any application disruptions, and the propagation of single and compounding errors remain within the user-specified error bounds. This demonstrates that our compression technique can seamlessly work for checkpoint/restart in FLASH application workflows.

II. Background

A. Discussion of Prior Work

Datasets generated by HPC applications usually exhibit diverse characteristics, and thus the compression performance may vary largely in different applications. Prior studies such as ZFP [9] have demonstrated that transform-based compression can provide high data decorrelation efficiency. Because this is desirable, researchers have been attracted to apply transforms (well utilized in JPEG, JPEG2000 [33], and MPEG [34]) to various HPC datasets.

Woodring et al. [35] used the JPEG 2000 technique on climate data compression, and Belmon et al. [36] used wavelet transform for spacecraft data compression. However, these works were limited to evaluating datasets from only one specific domain. Sasaki et al. [37] applied the Haar wavelet transform (HWT), and Li et al. [38] applied the Cohen-Daubechies-Feauveau wavelet transform (CDF 9/7) in their compressors. However, HWT requires multiple levels of decomposition while CDF 9/7 is limited to the levels of transforms for information compaction. Yeo et al. [39] and Ratnakar et al. [40] applied discrete cosine transform (DCT) for its high efficiency on data volume rendering, however, their approaches are less capable of bounding errors. It should be noted that applying JPEG or MPEG techniques directly in scientific data compression will usually introduce a large number of errors. Therefore, quantifying errors and proving the viability of the compressors for checkpoint/restart become important. Though Sasaki et al. [37] and Woodring et al. [35] showed certain precision of their compressors, their techniques were limited to exhibiting the stability and viability for checkpoint/restart mechanisms. ZFP [9] employed its own optimized data transform rather than using existing discrete ones, however, the reconstruction errors were not strictly bounded in ZFP, and the compression ratio was optimized mainly for 2D or 3D structured datasets.

B. Discrete Transforms

There are several well-known transforms such as discrete cosine transform (DCT), discrete wavelet transform (DWT), Cohen-Daubechies-Feauveau (CDF), and Fast Walsh-Hadamard (FWHT) that can be applied in a lossy compressor. While selecting the appropriate transform type for HPC datasets (in floating-point numbers) or designing an optimized transform like [9] is challenging, one should note that most of the commonly used discrete transforms share the same beneficial property: many natural signals have concise representations of original data after transforms.

We illustrate this property by using the "rlds" dataset from CMIP5 (shown in Figure 1a). Figure 1b shows the distribution of coefficients (in frequency domain) after applying a discrete transform (the Haar from DWT is used as an example). The trend sub-signal on the left half of Figure 1b shows a concise representation of the original data. The sub-signal on the right half of the figure presents its variations (defined as high-frequency coefficients), which are significantly smaller than the original datapoints. This is because discrete transforms tend to redistribute the energy contained in the signal and condense most of the energy into a small number of dominant coefficients (defined as low-frequency coefficients). The energy represents the data information, and the sum of squares

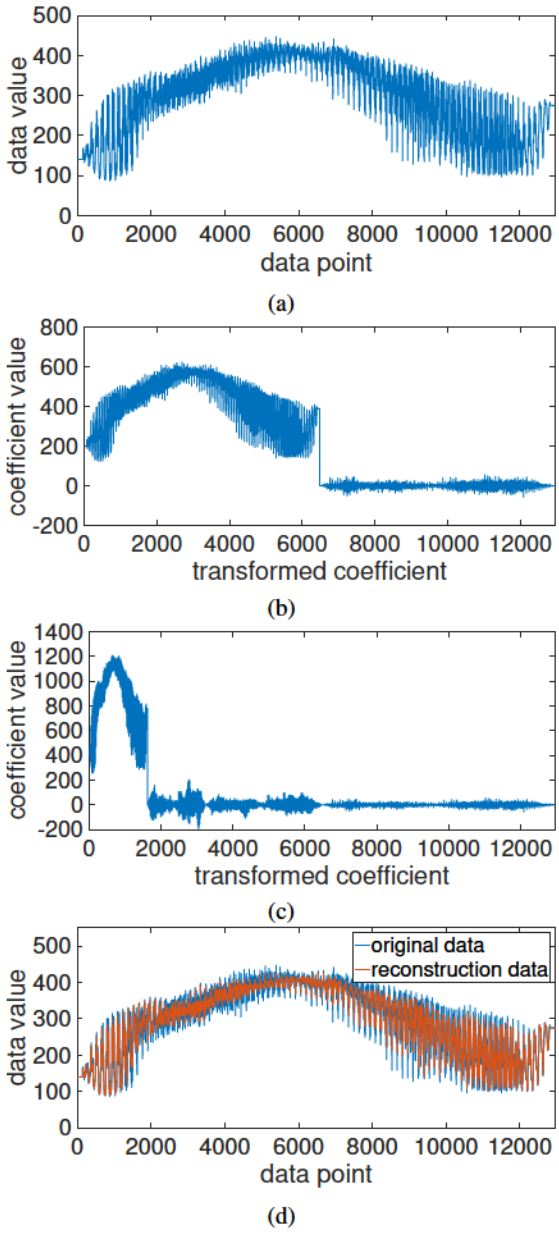


Fig. 1: The distribution of rlds dataset in various forms. (a) original data. (b) after applying 1-level Haar wavelet transform (HWT). (c) after applying 3-level HWT. (d) comparison between the original and reconstructed data with 3-level HWT (discarding high-frequency coefficients).

of the magnitudes of the coefficients usually represents the total data information [22].

For more concise representations of the original signal in transform-based compression, a decomposition process (e.g., multi-level decomposition) is commonly implemented. This process involves a recursive application of partitioning and transforming on high-frequency components, which is compute-intensive. For instance, Figure 1c shows the distribution of coefficients after applying 3-level decomposition. It is observed that the amount of low-frequency coefficients

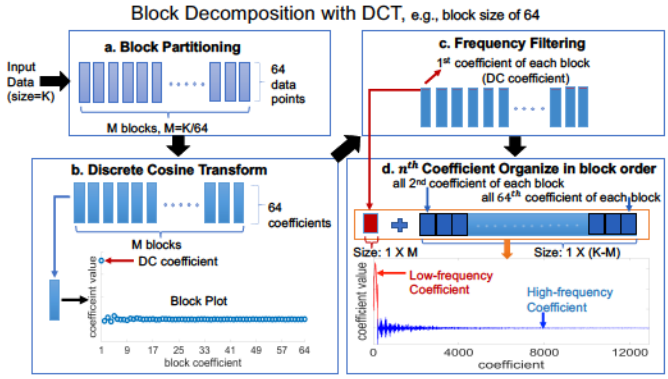


Fig. 2: An illustration of block decomposition with DCT using the rlds dataset. Our method is applicable to data of any dimensions as we regard all of them as flattened data.

in Figure 1c is less than that in Figure 1b, and also the values of low-frequency coefficients in Figure 1c are higher. This indicates that each of these low-frequency coefficients (shown on the left in Figure 1c) carries a larger percentage of information from the original data.

The above-mentioned unique features of discrete data transforms inspired us to design a transform-based lossy compressor. We designed our compressor with the following key questions in mind: 1) How do we capture dominant coefficients without incurring compute-intensive steps, such as recursive transforms? 2) Since we know that most of the transforms are lossless and the errors are introduced during the quantization or the elimination of high frequency coefficients (as shown in Figure 1d), how do we control the errors during quantization while achieving high compression ratios? 3) What is the impact of lossy compression in real-world application workflows?

III. Proposed Lossy Compressor

A. Block Decomposition with DCT

In our compressor, we select Discrete Cosine Transform (DCT) (the most commonly used DCT-II) as our transform method. Since DCT itself does not clearly favor any frequencies (as shown in Figure 3a), we design a novel block decomposition strategy, as illustrated in Figure 2. The inspiration comes from JPEG and MPEG techniques, where the first pixel of an image (i.e., the top left pixel in a 2D image) or the first frame of a video is stored as the most informative content during compression. In a similar way, we consider the first DCT coefficient as the most informative coefficient (i.e., DC coefficient which contains zero frequency), and the remaining coefficients as the AC coefficients (contain non-zero frequencies).

In our decomposition, we first partition data into small blocks. Specifically, we choose the block size of 64 as it can be used as 8×8 in 2D and $4\times4\times4$ in 3D. Then we apply DCT on block-based coefficients. Next, we collect the DC coefficient from each block, organize them based on the block sequence order, and consider them as low-frequency coefficients for the

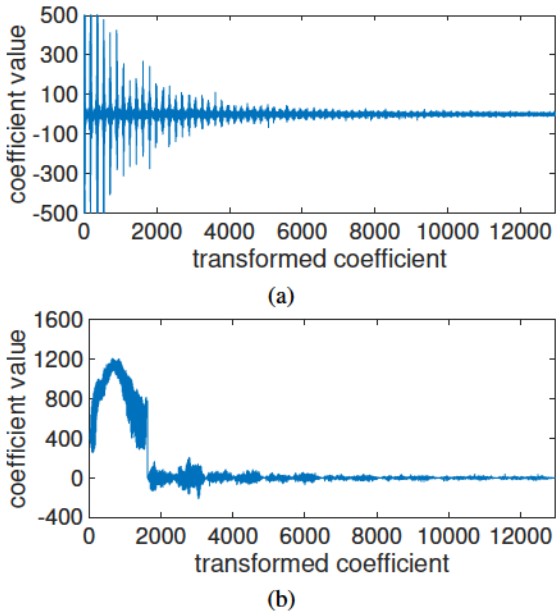


Fig. 3: The distribution of DCT coefficients. (a) after applying DCT transform. Note that we show the coefficients in range [-500, 500] for illustrative purposes only. The actual coefficients are in range [-8.4879e+03, 3.3362e+04]. (b) after applying block decomposition with the block size of 8. Note that the block size of 8 (instead of the block size of 64) is used here to show its similarity to 3-level HWT (as shown in Figure 1c) in terms of coefficient distribution.

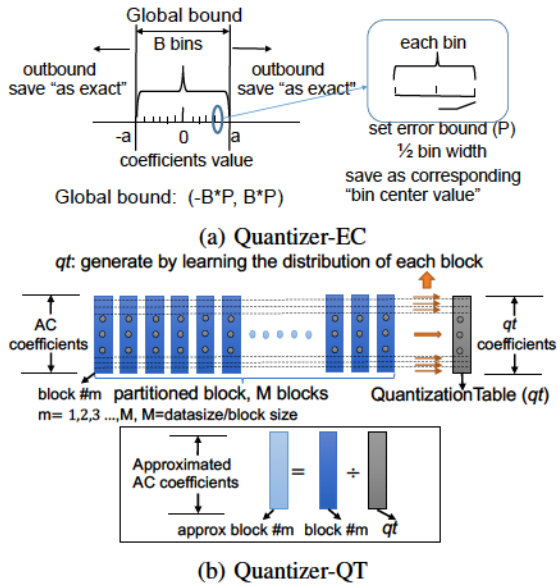


Fig. 4: Overview of our proposed quantizers.

entire data. The remaining AC coefficients are considered as the high-frequency coefficients. We note that the block size affects not only precision but also compression ratios and availability of parallelism. We also note that, while one can employ recursive DCT, but it gives lower compression ratios because of an extra index and sorting/ordering coefficients.

Figure 3b shows the distribution of DCT coefficients after applying our block decomposition strategy. This indicates that our decomposition strategy is able to improve the ability of DCT to distinguish dominant frequency coefficients. Also, based on the experiments on the rlds dataset, we found that the transformation time of a DCT with block decomposition (block size of 64) was an average of 3x faster than that of a 6-level HWT (N-level HWT transforms or DCT with block decomposition size of 2^N will generate low-frequency coefficients with the same length of $1/2^N$). Therefore, by applying a block decomposition strategy with DCT, we avoid not only the storage-consuming sorting step during transform but also time-consuming recursive transforms.

Another advantage of our block decomposition strategy is that the gradients of small block-based data could be smaller than that of an entire checkpoint (single iteration). As a result, applying DCT on block-based data could be more accurate and efficient, and incurs less memory overhead during encoding than on an entire checkpoint.

Moreover, we apply an exponent normalization on the block decomposed coefficients by aligning data values to a common exponent and expressing each value with reference to the largest exponent. This alignment unifies the range of transformed coefficients so that a smaller number of indices are required during encoding.

B. *Quantization Technique*

Since the majority of original data information is preserved in a small number of low-frequency coefficients, we store them as is and adopt a proper quantization technique on the high-frequency coefficients. In this section, we propose an adaptive quantization with two task-oriented quantizers (shown in Figure 4): either guaranteed maximum relative errors within specified bounds (Quantizer-EC) or high compression ratios within reasonable error rates (Quantizer-QT). Since checkpoint data evolves as simulation goes, our quantizer is determined for each checkpoint by learning the distribution of transformed high-frequency DCT components.

a) Quantizer-EC: Quantizer-EC is an adjustable error-controlled (EC) quantizer, where users can control compression errors by selecting their desired bound, denoted as P (in terms of relative error), and a total number of bins, C. Then a global bound GP is fixed to $[-P * C, P * C]$. We evenly partition the global bound GP into C bins, and the width of each bin is equal to twice of the error bound P. If the value of a coefficient is within a certain bin range, then it is approximated as the bin's center value. By doing this, the maximum error introduced after quantization will be smaller than the specified error bound P. For coefficients outside the global bound GP, we either save them as is to ensure accuracy or apply an extra truncation to improve the compression ratio. Algorithm 1 outlines this quantization method in detail. Although using a larger C can potentially reduce errors, we fix C to 255 ($2^8 - 1$) because a larger number of bins requires

Algorithm 1 The algorithm of Quantizer-EC.

Input: DT_H: high-frequency coefficients.
I: the number of checkpoint iteration.
J: the number of high-frequency coefficient in each iteration.
P: a user-specified error bound.
B: bin.
C: bin index.
Output: DT'_H: approximated coefficients.

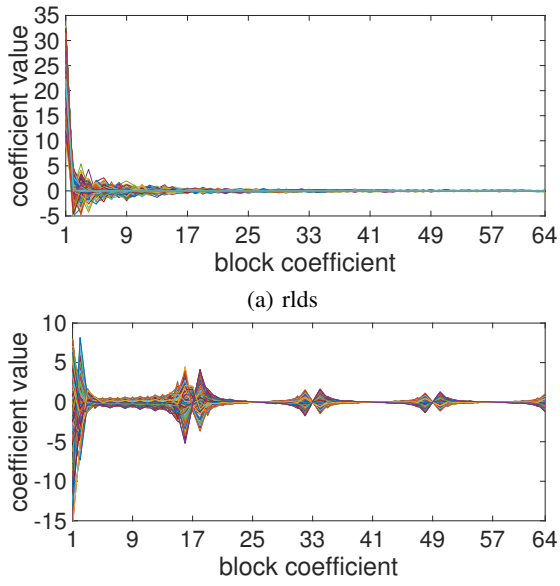
for $i = 1, 2, \ldots, I$ **do**
  Evenly partitioning global bound $[-2P * 255, 2P * 255]$ into 255 bins
  for $j = 1, 2, \ldots, J$ **do**
    if $DT_H i, j \in B_{i,c}, c = 1, 2, \ldots, 255$ **then**
      $DT_H i, j' \leftarrow$ approximation (center value of $B_{i,c}$)
    else
      $DT_H i, j' \leftarrow DT_H i, j$
    end if
  end for
end for

extra encoding steps in addition to Huffman coding for the bin index. Smaller bin sizes also can be used but we find using 255 (or 1-byte) maximizes compression ratios and efficiency with our encoding scheme. This binning strategy is straightforward and can be applied to high-frequency coefficients from other transforms like wavelets.

It should be noted that the binning mechanism described above is applied in the frequency domain (i.e., DCT coefficients), not in the spatial domain (i.e., original data). Therefore, extra errors could be introduced during the inverse transform for reconstructing data from lossy state. If the maximum compression errors (the difference between reconstructed data and original data) must be guaranteed within the user-specified error bound P, a revised error bounding method is needed. This strict error guarantee is dependent on the transform employed because each transform has a different inverse transform property. For DCT, its inverse transform has the same computation as the non-inverse one, which is calculated as the sum of weighted coefficients. Mathematically speaking, the new max error in the spatial domain is then calculated as $\sqrt{N}$ times the max error in the frequency domain (where N is the block size). Therefore, users need to set their error bound to $P/\sqrt{N}$ in the frequency domain such that, after inverse transforming, the compression errors are bounded within P in the original domain. This makes DCT with Quantizer-EC (namely DCT-EC) a conservative yet efficient compressor.

b) Quantizer-QT: Quantizer-EC described so far applied the quantization to AC coefficients (high-frequency) directly. However, there is an opportunity to improve compression ratios further by applying various quantization methods to AC coefficients so that the number of bits required for encoding is reduced. This is inspired by the property of discrete transforms wherein spatial frequencies represent the detailed information of the original data. In other words, if the original data values are spatially smooth (which is common in many scientific applications that model physical phenomenon), a block in the DCT domain will have smooth high-frequency coefficients (i.e., with small variations).

To verify whether there is exploitable smoothness in high-frequency coefficients, we take the rlds and Eddy datasets



(a) rlds

(b) Eddy

Fig. 5: The distribution of DCT block coefficients. (a) and (b) show the overlay of all blocks in a single checkpoint after applying transform and exponent normalization on rlds and Eddy datasets with block size of 64.

(described in Table I) as examples and plot the overlays of their block coefficients, as depicted in Figure 5a and Figure 5b, respectively. Note that these two figures show the overlays of all the blocks of a single checkpoint after applying the data transform, not the distribution of the checkpoints shown in step d of Figure 2. Our investigation of these plots leads to several findings.

First, most distributions of block coefficients demonstrate two distinctive patterns: 1) the DC coefficient (the first block coefficient) contains most of the data information, and the remaining ones are either small values or zeros (as depicted in Figure 5a); and 2) besides the first dominant coefficient, there are a few secondary-informative coefficients, or spikes (as depicted in Figure 5b). To confirm the occurrence of these patterns in scientific datasets, we define two cases: Case 1 is when DC coefficient carries more than 90% of the total energy of a block, and Case 2 is when Case 1 is not applicable but the top three dominant coefficients carry more than 90% of the total energy. Our statistical analysis indicates that there is an average of 62.67% occurrence of Case 1 and 16.95% occurrence of Case 2 on six evaluated datasets (described in Table I). The total occurrence of Case 1 and Case 2 illustrates that most of the data information can be represented by a small number of coefficients. In other words, most of the block coefficients are small in magnitude (i.e., smooth) and contain less data information.

Second, it is worth mentioning that different checkpoint data in the same dataset show a similar block pattern (i.e., the transformed coefficients capture temporal redundancy). Also, the secondary-dominant coefficients (spikes) are always

Algorithm 2 The algorithm of Quantizer-QT applied to block coefficients in a single iteration.

Input: BA: AC block coefficients.
qt: quantization table (initial value are set to zeros).
M: the number of blocks.
N: the number of block coefficient in each block, also the number of qt's coefficient.
Output: BA': approximated coefficients.
for $n = 1, 2, \ldots, N$ **do**
  for $m = 1, 2, \ldots, M$ **do**
    if abs $(BAn, m) \geq qt_{n,1}$ **then**
      $qtn, 1 \leftarrow$ abs (BAn, m)
    end if
  end for
end for
$BAn, m' \leftarrow \frac{BAn,m}{qtn,1}$, apply algorithm EC on BAn, m'

clustered/oscillated around similar positions, as shown in Figure 5b. Therefore, if we can model the repetitive pattern from one checkpoint into a quantization table, then it can be applied to the blocks showing the same patterns in other checkpoints. Exploiting the same spatiotemporal pattern not only simplifies our quantization step (e.g., elimination of a zigzag order used in JPEG), but also reduces the number of bits required to represent coefficients.

Since most of the block coefficients show descent smoothness and repetitiveness (as mentioned in above findings), we design a quantization table qt in our quantizer, namely Quantizer-QT. We generate qt by finding the maximum value of the n^{th} coefficient over all the blocks and build a quantization table of length $N-1$, where N is the block size and $n \leq N$. Note that the DC coefficients of the blocks are not included in this step as they are saved as is. As outlined in Algorithm 2, qt is calculated as $qt_{n,1} = \max\{|BA_{n,1}|, |BA_{n,2}|, |BA_{n,3}|, ..., |BA_{n,m}|\}$, where m is the total number of decomposed blocks and the input data is a one-dimensional floating-point array. All AC coefficients are then converted into a global bound and quantized using Quantizer-EC after being divided by qt.

As an example of how Quantizer-QT works, we select rlds and Eddy datasets to demonstrate our design. Figure 6c and Figure 6d show the distribution of block coefficients of rlds and Eddy after being divided by their quantization tables shown in Figure 6a and Figure 6b. As we can see, the resulting coefficients require much fewer bit representations after quantization table is applied as they are much narrower in range. Our Quantizer-QT is a mechanism for striking a balance between loss of precision and compression ratio. Therefore, DCT with Quantizer-QT (namely DCT-QT) is designed for scientists who want full compression potential but with less tight error bounds.

C. Data Encoding

The last step of our lossy compression is to encode the data into our compressed format. In this stage, the first dominant coefficient in each block is stored as its original value. The remaining high-frequency coefficients will be quantized and stored as their corresponding approximated values. If a coefficient lies within the global bound, it will be stored as the bin's center value; otherwise, it will be saved as is for

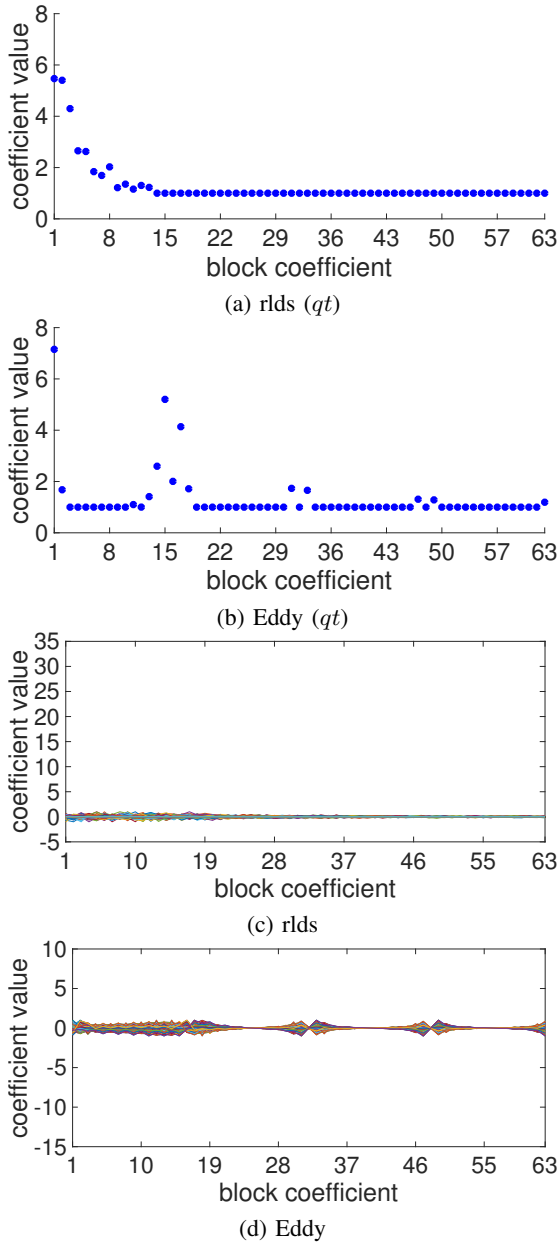


Fig. 6: An example quantization table for a single checkpoint of (a) rlds and (b) Eddy. The overlay of AC coefficients after applying individual qt to (c) rlds (d) Eddy.

guaranteeing error rates. For DCT-QT, a $1 \times N$ quantization table (in double-precision format) needs to be saved. An extra 1 bit is also needed to distinguish the out-of-range coefficients and approximated coefficients. Lastly, we apply Gzip to bin indices and dominant coefficients, which further improves the compression ratio. While we use Gzip, the choice of using add-ons such as Huffman, Gzip, ZSTD depends on user's need, i.e., higher compression ratios vs. higher compression speed. Note that our encoding scheme can be performed on the decomposed block, which is similar to ZFP, allowing random read/write access to compressed floating-point data at block

TABLE I: Evaluated Datasets and their characteristics.

Code	Dataset	Description	Size
CMIP5	rlds	Surface downwelling longwave radiation	218 MB
	mrsos	Moisture content of soil layer	218 MB
FLASH	Sedov	Hydrodynamical test code involving strong shocks and non-planar symmetry	576 MB
	Cellular	Burn simulation: cellular nuclear burning problem	1.35 GB
Nek5000	Eddy	2D solution to Navier-Stokes equations	820 MB
	Vortex	Inviscid vortex propagation: tests the problem in earlier studies of finite volume methods	580 MB

granularity.

Huffman encoding or Gzip employed here can improve compression ratios by removing redundancies but can be time-consuming. However, our approach does not involve other computationally-intensive tasks. For example, unlike ZFP where floating-point data is converted into fixed-point representations before transform, we apply DCT directly on floating-point data. DCT itself is also fast compared to other transforms and can be performed efficiently because of our block decomposition. DCT transform speed is related to data-size and larger data size will lead to a linear time increase.

IV. Evaluation

A. Experimental Setup

1) System: We conducted our experiments on the Massachusetts Green High Performance Computing Cluster (MGHPCC) for running FLASH and Nek5000 applications at various scales ranging from 64 to 1,024 cores to generate datasets for our experiments. We used MPICH-3.2 and PnetCDF-1.7.0 for running the evaluated applications (FLASH-4.2.1 and the latest version of Nek5000 solvers from the repository).

2) Dataset: We used six real-world scientific datasets obtained from three HPC code packages: FLASH, CMIP5, and Nek5000, as shown in Table I. The number of checkpoints is between 153 and 1,000 depending on datasets. The applications used in our evaluation are from three different scientific domains: climate simulation, hydrodynamic simulation, and nuclear combustion simulation. They all produce double-precision floating-point data. For the solvers and datasets provided in the FLASH code, we evaluated the five most important variables: temperature, pressure, density, initial energy, and total energy from each checkpoint file. For the benchmarks provided in the CMIP5 code package, we used two atmospheric outputs of climate simulations: rlds and mrsos (stored in a separate 1D array). These two datasets are known to be hard to compress, due to less correlation between neighboring datapoints compared to the mesh data produced by conventional simulations, such as hydrodynamics and fluid dynamics.

3) Evaluated Schemes: We compared our compression algorithms with two state-of-the-art lossy compressors: SZ (SZ 1.4) and ZFP. SZ represents prediction-based compressors which utilize curve-fitting, scalar quantization and Huffman coding to compress predictable data points. ZFP represents transform-based compressors which include mantissa conversion and orthogonal transformation. Other lossy compressors were not evaluated here as they exhibited similar or less competitive results to SZ and ZFP [20]. Lossless compressors were not considered in our evaluation due to their limitation in compression ratios (e.g., no more than 2x based on Shannon's theorem summarized in [6]).

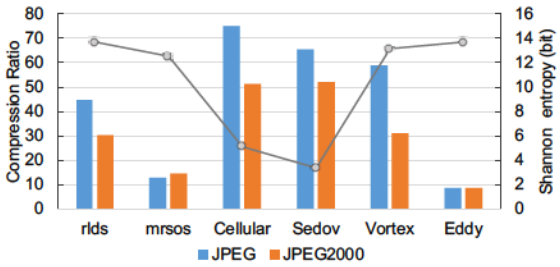


Fig. 7: The bar graphs (primary y-axis) show the compression ratios using off-the-shelf JPEG and JPEG 2000 techniques, with quality of 75; The line graphs show the Entropy of evaluated datasets.

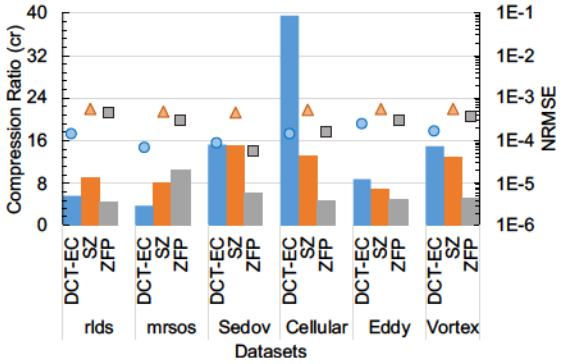
4) Evaluation Metrics:

a) Entropy: Shannon entropy, a popularly used metric in lossless compression, is used to predict the compressibility of datasets. Specially, the Entropy (H in bits) of a dataset x is calculated as follows: $H(x) = -\sum_{i=1}^{n} P(x_i)log(P(x_i))$.

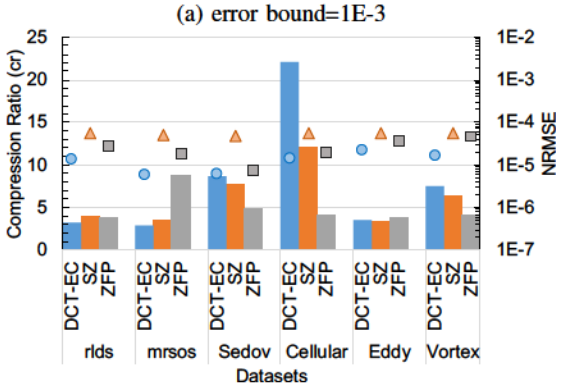
b) Compression Ratio and Accuracy: Compression ratio (CR), is used to evaluate the size reduction from compression, which is defined as the original size divided by the compressed size. To measure the difference between the original data and the reconstructed data, we use the metrics defined in Z-checker [41], a framework for assessing lossy compressors for scientific data. Specifically, we use: (1) Maximum Relative Error (denoted as $\max\theta$), which is calculated as the maximum absolute error divided by the value range of the data. We denote θ as the relative error instead of pointwise error because of the different value ranges exhibited in datasets. In our compression algorithm, users can define their own relative error bound (P). For DCT-EC, the relative error (θ) is guaranteed within the error bound; (2) Average Error: Normalized Root-Mean-Square Error (NRMSE) and Mean Relative Error ($\bar{\theta}$) are used to measure average relative error; (3) Distortion and Correlation: The Peak Signal-to-Noise Ratio (PSNR) and Pearson Correlation are used to evaluate compression error.

B. Evaluation Results

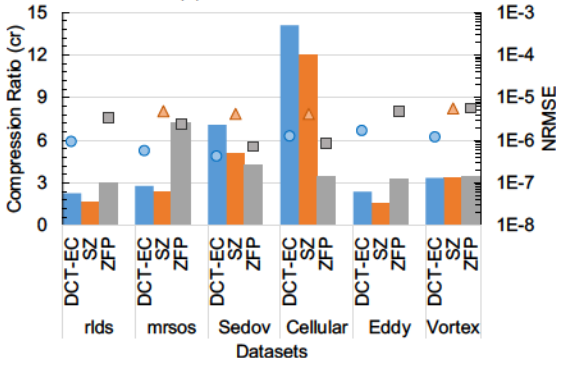
1) Entropy and Compression Ratio: We first use Shannon Entropy as well as the achieved compression ratios from the image-based compression techniques, JPEG and JPEG2000 to estimate the compressibility of the datasets. We use off-the-shelf JPEG techniques as a compression-ratio indicator by fixing compression quality. As shown in Figure 7, the entropies of rlds, mrsos, Vortex and Eddy are higher and their compression ratios are lower than those of Sedov and Cellular. Therefore, we consider rlds, mrsos, Vortex and Eddy as hard-to-compress datasets. It is interesting to note that the



(a) error bound=1E-3



(b) error bound=1E-4



(c) error bound=1E-5

Fig. 8: The bar graphs (primary y-axis) show compression ratios for DCT-EC, SZ, and ZFP with the max error bounds (P) of (a) 1E−3, (b) 1E−4 and (c) 1E−5. The markers (secondary y-axis) show deviations between original data and reconstructed data in terms of NRMSE with different lossy compressors. (blue circle: DCT-EC, orange triangle: SZ, grey square: ZFP).

compression ratios achieved by applying JPEG are higher than the ones from JPEG 2000.

2) Compression Ratio and Error Bound: Figure 8 presents the compression ratios (CRs) of DCT-EC, SZ and, ZFP, when P is set to 1E−3, 1E−4, and 1E−5, respectively. Since ZFP might not respect the error bound, for a fair comparison, we manually adjust P of ZFP to guarantee that the $\max\theta$ is smaller than P. While we observe an overall trend that stricter error bounds give lower CRs as shown in Figure 8, DCT-EC

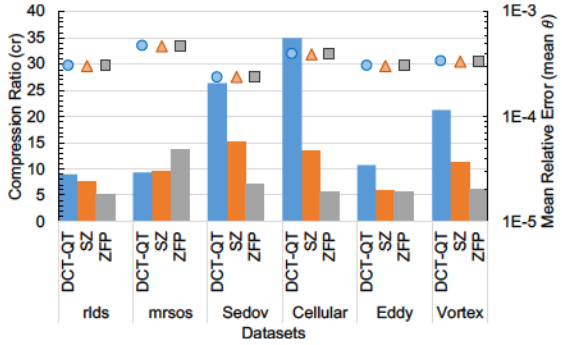


Fig. 9: The bar graphs show compression ratios for DCT-QT, SZ, and ZFP whereas the markers show deviations between original data and reconstructed data in terms of Mean Relative Error ($\bar{\theta}$).

shows the highest CRs for Sedov and Cellular for all three error bounds we evaluated. For example, when P is 1E−3, the CR of DCT-EC for Cellular is 38, which is 290% and 790% higher than SZ's 13 and ZFP's 4.8, respectively. For Sedov, with P=1E−5, the CR of DCT-EC is 7.1, which is 140% and 167% higher than SZ's 5 and ZFP's 4.2, respectively. It is also shown that DCT-EC has the highest CRs for Eddy and Vortex with P of 1E−3. We observe that ZFP has the best compression ratios for mrsos, which is one of the hard-to-compress datasets according to our entropy analysis (discussed in Section IV-B1). DCT-EC did not perform well with mrsos because of its characteristics: high entropy but low energy concentration after transforms. Our analysis indicates that ZFP requires fewer bits to encode the coefficients in mrsos than DCT-EC and SZ. Figure 8 also presents the average errors (in terms of NRMSE) for DCT-EC, SZ and ZFP with P of 1E−3, 1E−4, and 1E−5. It is shown that, for most of the datasets, both ZFP and DCT-EC produce relatively lower average error than SZ does. DCT-EC also produces a low average error while achieving high CRs on Sedov and Cellular.

We next compare the performance of DCT-QT with SZ and ZFP. Since the coefficients in DCT-QT are adjusted by the quantization table, DCT-QT may not strictly bound the user-defined error rates. Therefore, for a fair comparison, we adjusted P and evaluated CRs of SZ and ZFP by aligning the mean relative error ($\bar{\theta}$) to the same value for each dataset. As shown in Figure 9, DCT-QT achieves much higher CRs than SZ and ZFP on rlds, Sedov, Cellular, Eddy, and Vortex. The CR on mrsos is close to 10, which is quite competitive.

To prove how much higher compression ratio DCT-QT could achieve while guaranteeing $\max\theta$ within certain error bound, we compare DCT-QT with SZ by adjusting SZ's P to $\max\theta$ of DCT-QT. The comparison result shows that DCT-QT can achieve CRs of 122.83, 26, 203.10, 183.54, 19.65 and 62.27 on rlds, mrsos, Cellular, Sedov, Eddy, and Vortex, respectively, while SZ can achieve CRs of 42.00, 19.62, 88.21, 141.32, 16.09 and 59.18, respectively. (The specific $\max\theta$ is set to 5E−2, 1E−2, 7E−2, 8E−2, 1.5E−2, and 1E−2). It is shown that with less tight error bounds, the CR of

TABLE II: Evaluation of Peak Signal-to-Noise Ratio (PSNR (dB)) with different lossy compressors on selected datasets.

$\max\theta$	rlds			mrsos			Sedov			Cellular			Eddy			Vortex		
	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ
1E−3	76.85	67.01	64.83	83.25	70.56	66.34	81.14	84.53	67.27	76.89	73.51	64.98	72.08	70.49	64.77	75.79	68.67	64.77
1E−4	97.30	91.05	84.79	104.58	94.58	85.647	104.41	101.20	86.55	96.91	91.86	84.83	92.83	88.57	84.77	95.77	86.69	84.77
1E−5	120.81	109.08	Inf	124.88	112.66	106.35	127.94	121.28	107.81	117.98	121.98	109.42	115.70	106.63	Inf	118.51	104.70	104.82

TABLE III: Evaluation of Pearson correlation with different lossy compressors on selected datasets.

$\max\theta$	rlds			mrsos		
	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ
1E−3	$\geq$1-10^{-7}	$\geq$1-10^{-6}	$\geq$1-10^{-6}	$\geq$1-10^{-8}	$\geq$1-10^{-7}	$\geq$1-10^{-6}
1E−4	$\geq$1-10^{-9}	$\geq$1-10^{-9}	$\geq$1-10^{-8}	$\geq$1-10^{-10}	$\geq$1-10^{-9}	$\geq$1-10^{-8}
1E−5	$\geq$1-10^{-12}	$\geq$1-10^{-11}	$\geq$1-10^{0}	$\geq$1-10^{-12}	$\geq$1-10^{-11}	$\geq$1-10^{-10}

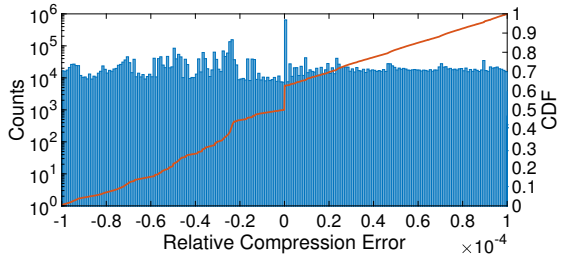
$\max\theta$	Sedov			Cellular		
	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ
1E−3	$\geq$1-10^{-7}	$\geq$1-10^{-8}	$\geq$1-10^{-6}	$\geq$1-10^{-7}	$\geq$1-10^{-7}	$\geq$1-10^{-6}
1E−4	$\geq$1-10^{-10}	$\geq$1-10^{-9}	$\geq$1-10^{-8}	$\geq$1-10^{-9}	$\geq$1-10^{-9}	$\geq$1-10^{-8}
1E−5	$\geq$1-10^{-12}	$\geq$1-10^{-11}	$\geq$1-10^{-10}	$\geq$1-10^{-11}	$\geq$1-10^{-12}	$\geq$1-10^{-11}

$\max\theta$	Eddy			Vortex		
	DCT-EC	ZFP	SZ	DCT-EC	ZFP	SZ
1E−3	$\geq$1-10^{-7}	$\geq$1-10^{-6}	$\geq$1-10^{-6}	$\geq$1-10^{-7}	$\geq$1-10^{-6}	$\geq$1-10^{-6}
1E−4	$\geq$1-10^{-9}	$\geq$1-10^{-8}	$\geq$1-10^{-8}	$\geq$1-10^{-9}	$\geq$1-10^{-8}	$\geq$1-10^{-8}
1E−5	$\geq$1-10^{-11}	$\geq$1-10^{-10}	$\geq$1-10^{0}	$\geq$1-10^{-11}	$\geq$1-10^{-10}	$\geq$1-10^{-10}

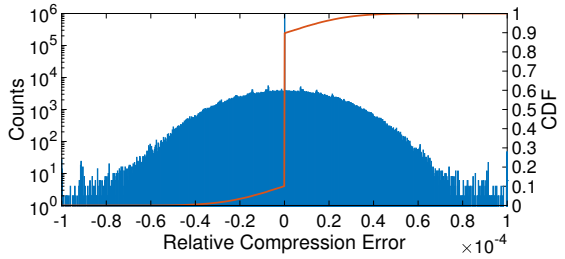
DCT-QT increases faster than that of SZ. The high CRs achieved by DCT-QT can help scientists who have special needs on checkpoint reduction but with fewer constraints on error precision. Overall, both DCT-EC and DCT-QT show remarkably higher compression ratios for FLASH datasets (multi-physics, hydrodynamic code) and provide comparable results for the other datasets.

3) Distortion and Correlation: Table II and III show the PSNRs and the Pearson correlation coefficients for DCT-EC, SZ and ZFP. From Table II, we can see that with P of 1E−5, all three compressors obtain PSNRs higher than 100. Compared with SZ and ZFP, DCT-EC obtains a higher range between 110 and 120 on most datasets. From Table III, we can see that all three compressors have "five nines" [42] or better correlations with all three Ps. For DCT-QT, it obtains PSNRs of no more than 50 and Pearson correlations of no more than 0.9999 on six datasets (with $\max\theta$ of around 1E−3), which shows its limitation in compression precision compared with DCT-EC.

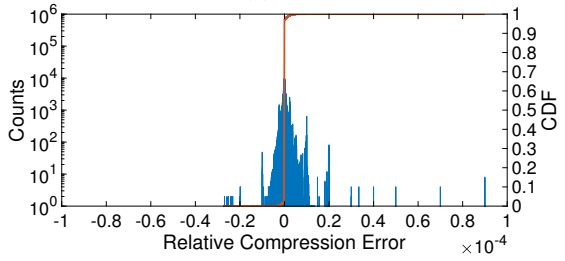
4) Distribution of Compression Errors: Figure 10 shows the distribution of relative errors for DCT-EC, SZ, and ZFP with P of 1E−4. We can see that the distribution of compression error for SZ is nearly uniform while those for ZFP and DCT-EC are nearly normal. Moreover, DCT-EC is more conservative with regard to the accuracy requirement, which is due to its transform property discussed in Section II-B and Section III-B. Also, from the cumulative distribution function (CDF), we can see that more compression errors are centered around zero for DCT-EC than SZ and ZFP. This illustrates that DCT-EC introduces less errors than SZ and ZFP in a given P. We observe similar trends with P of 1E−3 and 1E−5.



(a) SZ



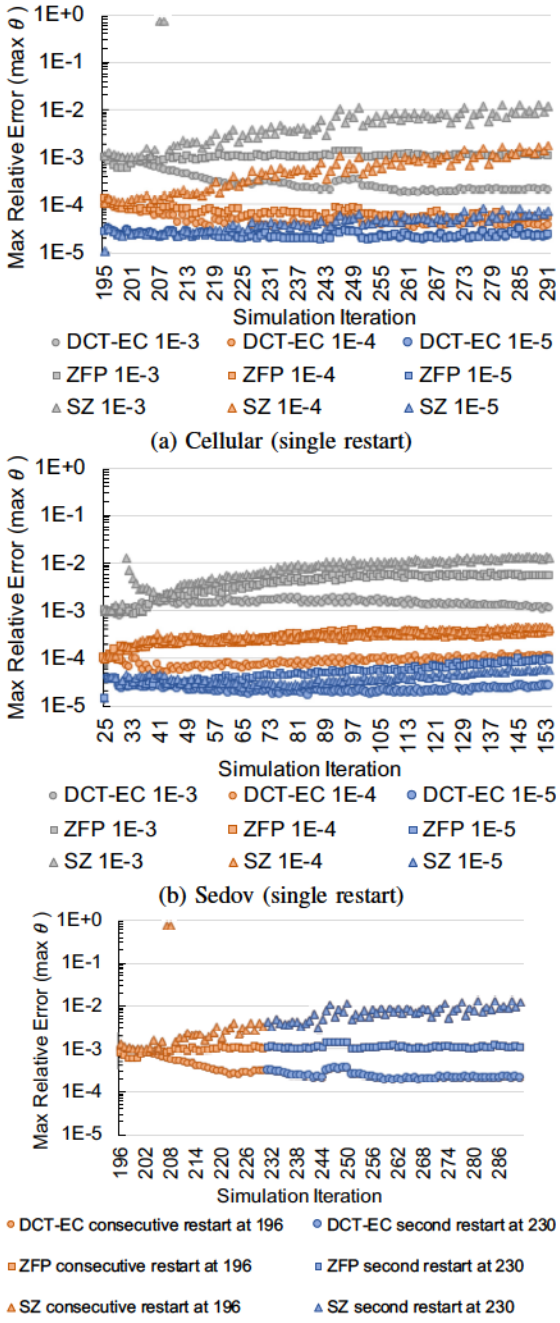
(b) ZFP



(c) DCT-EC

Fig. 10: The distribution of relative error (θ), with P=1E−4, for (a) SZ (b) ZFP (c) DCT-EC (similar trend in DCT-QT). The primary y-axis shows the error histogram and the secondary y-axis shows the cumulative distribution function (CDF).

5) Impact on Checkpoint/Restart: In real simulation runs, the errors of successive results after restarting from failures may or may not converge. Therefore, evaluating the impact of errors introduced by lossy compression against the original values at each time step is critical, even if all evaluated compressors provide a mechanism to guarantee error bounds. Since DCT-EC, SZ and ZFP are designed for scientists who have high compression precision demands, we choose them for real simulation workflow comparison. To quantify the impact of errors, we obtain their compressed restart files and compare with their actual restarts. Since the higher CRs on Sedov and Cellular datasets are very promising, we use Sedov and Cellular solvers available in FLASH for comparison. Specifically, we evaluate the restart of Sedov and Cellular at time step 25 and 195, respectively, and run them until the

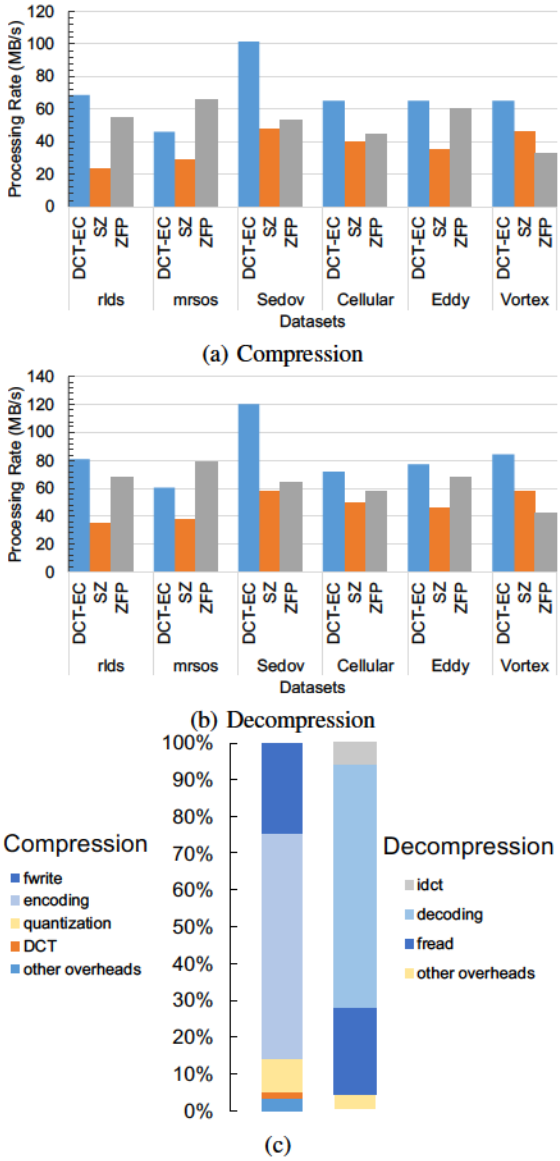


(a) Cellular (single restart)

(b) Sedov (single restart)

(c) Cellular (multiple, or compound, restart)

Fig. 11: The maximum relative error variation during simulation timestamps in two solvers in FLASH. The y-axis shows the maximum relative error between the data generated from restart using reconstructed data and the original data. Note that, in (a) and (b), the restart points of SZ and DCT-EC show value no higher than 1E−5.



(a) Compression

(b) Decompression

(c)

Fig. 12: (a) Compression and (b) Decompression speed (MB/s) using DCT-EC, SZ and ZFP with P of 1E−3. Similar trends were observed with P of 1E−4 and 1E−5. (c) Breakdown of compression and decompression time with P of 1E−3.

simulation ends. We have tried the restart from 1 to the final steps and observed the same effect because each step, after restart, solves the same Sedov explosion problem. We choose step 25 and 195 to reflect the randomness. The evaluations are done with error bound P of 1E−3, 1E−4, and 1E−5.

First, we observe that the overall execution, in terms of the number of simulation steps to converge, is not affected by the reconstructed data. Both solvers end at the same simulation time steps as the original executions. This demonstrates that the error in the reconstructed data is not significantly influencing the mathematical formulas that each solver is computing.

Next, we evaluate how the error introduced by each compressor translates into errors in the actual scientific data and its impact on real application simulations. As shown in Figure 11a, the reconstructed restart file obtained from SZ with P of 1E−3 generates a large spike after 20 steps from the restart. While we conjecture that the exact cause of this spike is related to a specific mathematical formula

in Cellular, it needs further verification from the domain scientist. From Figure 11a, we observe that the reconstructed restart files obtained from SZ with all three Ps show small divergence. The $\max\theta$ keeps increasing for all cases. We attribute this to the inherent error distribution in SZ, where some scientific researchers require the compression errors to follow Gaussian white noise distribution to satisfy the viability of simulation. From Figure 11a, we also observe that, for all reconstructed restart cases, the relative error for ZFP or DCT-EC becomes relatively stable after a couple of simulation time steps and finally shows the trend of decaying. With the highest allowable error rate (P= $1\mathrm{E}{-3}$), DCT-EC shows a quicker stabilization than the other two compressors. A similar trend is also observed for Sedov, as shown in Figure 11b. With the lowest error rate (P = $1\mathrm{E}{-5}$), all three compressors show slower convergence, and thus more simulation steps are needed before they stabilized.

In addition, we evaluate the effect of compounded errors with P of $1\mathrm{E}{-3}$ on Cellular using the following restart scenarios: (1) two successive restarts (which is similar to [43]), at time step 195 and 196, and (2) two restarts at two different time intervals (second restart at time step 230). Both scenarios show that the compounded errors overlap with the errors generated from the original restart, which is encouraging because all three compressors work well with multiple restarts. DCT-EC shows minimal impact of compounded errors and remains below the error bound P.

All of these results demonstrate that both solvers (Sedov and Cellular) worked well with various checkpoint/restart scenarios (single or multiple restarts) in FLASH without disrupting the original execution time and without a noticeable deviation from numerical convergence. Therefore, as far as FLASH solvers are concerned, all compressors do not require users to adjust data for consistency after a restart.

6) Compression Throughput: Figures 12a and 12b present the average compression and decompression speeds (excluding disk I/O) on all evaluated datasets using DCT-EC, SZ, and ZFP, with P of $1\mathrm{E}{-3}$. As shown in these figures, DCT-EC and ZFP outperform SZ on most datasets. This indicates that transform-based compression is overall faster than prediction-based compression, at least for the datasets evaluated in this work. It should be noted that DCT-EC includes Gzip encoding on indices, while ZFP does not. It is also shown in the figures that the decompression speeds are faster than compression speeds on all three compressors, showing a promising result for HPC workloads since data is compressed once and decompressed frequently. Figure 12c, on the other hand, shows the breakdown of DCT-EC. We can see that the encoding time including encoding is around 60% of the total compression time. For DCT-QT, the quantization takes an average of 26.3% of the total compression time on all evaluated datasets.

7) Scalability: Table IV presents the scalability results of the compression and decompression time (excluding the I/O time) of DCT-EC (with P of $1\mathrm{E}{-3}$) on dataset Celluar while varying the number of threads from 2 to 16. We ran each experiment ten times and used the average time. As shown in the table, the parallel efficiencies for both compression and decompression are at least above 91.8% and have almost linear speedup with an increasing number of threads. We attribute this linear speedup to the block-based approach in our compressor design. For example, DCT transform can be done in parallel with the decomposed blocks. Subsequent block-based filtering and quantization can be also done in parallel without any communication among the distributed blocks. In order to calculate the exact file offset for locally encoded data for writing, our method needs to perform parallel prefix operations, which is straightforward to parallelize. Overall, we believe that our proposed method is scalable at a production level.

TABLE IV: Scalability of compression and decompression.

Number of threads	Comp speedup	Comp parallel efficiency	Decomp speedup	Decomp parallel efficiency
2	1.99	99.5%	1.99	99.5%
4	3.98	99.5%	3.97	99.25%
8	7.88	98.5%	7.87	98.38%
16	14.7	91.88%	14.9	93.13%

V. Conclusion and Future Work

In this work, we propose a lossy compression technique. Specially, we apply DCT with a novel block decomposition strategy and design an adaptive quantization with two task-oriented quantizers: Quantizer-EC and Quantizer-QT. We compare our compressor with SZ-1.4 and ZFP, using six scientific datasets from three real scientific applications. Our compression approach achieves comparable performance, showing 3x–38x compression ratio while guaranteeing user-specified error bounds on the evaluated datasets. Moveover, we investigate the reconstructed data from two solvers under several checkpoint/restart scenarios. It is shown that restarts from a lossy state are viable without any application disruptions, and the propagation of single and compounding errors remain within the user-specified error bounds. We empirically demonstrate that our compression technique can seamlessly work for checkpoint/restart employed in the FLASH application workflows and thus is considered as a promising approach for lossy data compression.

In our future work, we plan to expand the proposed compression technique in several ways. First, we plan to improve the compression ratio of our technique by optimizing the quantization model. We also want to improve the compression quality of our technique on larger datasets. Lastly, we plan to incorporate our compression mechanism into various layers in the HPC I/O software stack, including MPI-IO [44], PnetCDF [45], HDF5 [46], and ADIOS.

References

[1] P. Beckman, R. Brightwell, B. R. de Supinski, M. Gokhale, S. Hofmeyr, S. Krishnamoorthy, M. Lang, B. Maccabe, J. Shalf, and M. Snir, "Exascale Operating Systems and Runtime Software Report," 2012.

[2] R. Lucas, J. Ang, K. Bergman, S. Borkar, W. Carlson, L. Carrington, G. Chiu, R. Colwell, W. Dally, J. Dongarra, A. Geist, G. Grider, R. Haring, J. Hittinger, A. Hoisie, D. Klein, P. Kogge, R. Lethin, V. Sarkar, R. Schreiber, J. Shalf, T. Sterling, and R. Stevens, "ASCAC Subcommittee for the Top Ten Exascale Research Challenges," 2014.

[3] R. J. Small, J. Bacmeister, D. Bailey, A. Baker, S. Bishop, F. Bryan, J. Caron, J. Dennis, P. Gent, H. ming Hsu, M. Jochum, D. Lawrence, E. Muñoz, P. diNezio, T. Scheitlin, R. Tomas, J. Tribbia, Y. heng Tseng, and M. Vertenstein, "A new synoptic scale resolving global climate simulation using the Community Earth System Model," *Journal of Advances in Modeling Earth Systems*, pp. 1065–1094, 2014.

[4] 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 *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing (HPDC)*, 2014, pp. 203–214.

[5] Computational and Information Systems Laboratory, "Yellowstone: IBM iDataPlex System (Climate Simulation Laboratory)," https://www2.cisl.ucar.edu/resources/computational-systems/yellowstone, 2012.

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

[7] 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

[8] 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.

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

[10] "GNU Gzip," http://www.gzip.org/.

[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] M. Dipperstein, "LZSS (LZ77) Discussion and Implementation," http://michael.dipperstein.com/lzss/.

[13] B. Welton, D. Kimpe, J. Cope, C. M. Patrick, K. Iskra, and R. Ross, "Improving I/O Forwarding Throughput with Data Compression," in *Proceedings of the IEEE International Conference on Cluster Computing*, Sept 2011, pp. 438–445.

[14] T. Z. Islam, K. Mohror, S. Bagchi, A. Moody, B. R. de Supinski, and R. Eigenmann, "MCREngine: A scalable checkpointing system using data-aware aggregation and compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2012.

[15] E. R. Schendel, Y. Jin, N. Shah, J. Chen, C. S. Chang, S. H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "ISOBAR Preconditioner for Effective and High-throughput Lossless Data Compression," in *IEEE 28th International Conference on Data Engineering*, April 2012, pp. 138–149.

[16] G. Han, X. Wu, S. Zhang, Z. Liu, and W. Li, "Error Covariance Estimation for Coupled Data Assimilation Using a Lorenz Atmosphere and a Simple Pycnocline Ocean Model," *Journal of Climate*, vol. 26, no. 24, pp. 10 218–10 231, 2013.

[17] D. Zupanski and M. Zupanski, "Model Error Estimation Employing an Ensemble Data Assimilation Approach," *Monthly Weather Review*, vol. 134, no. 5, pp. 1337–1354, 2006.

[18] J. L. Anderson, "An Ensemble Adjustment Kalman Filter for Data Assimilation," *Monthly Weather Review*, vol. 129, no. 12, pp. 2884–2903, 2001.

[19] J. S. Whitaker, T. M. Hamill, X. Wei, Y. Song, and Z. Toth, "Ensemble Data Assimilation with the NCEP Global Forecast System," *Monthly Weather Review*, vol. 136, no. 2, pp. 463–482, 2008.

[20] 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.

[21] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization," in *Proceedings of the 31th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 2017.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[23] K. Sayood, *Introduction to Data Compression (2nd Ed.)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000.

[24] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data," in *Proceedings of the 17th International Conference on Parallel Processing - Volume Part I*, ser. Euro-Par'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 366–379. [Online]. Available: http://dl.acm.org/citation.cfm?id=2033345.2033384

[25] Z. Chen, S. W. Son, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "NUMARCK: Machine Learning Algorithm for Resiliency and Checkpointing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 733–744. [Online]. Available: https://doi.org/10.1109/SC.2014.65

[26] Z. Yuan, W. Hendrix, S. W. Son, C. Federrath, A. Agrawal, W. Liao, and A. N. Choudhary, "Parallel Implementation of Lossy Data Compression for Temporal Data Sets," in *23rd IEEE International Conference on High Performance Computing, HiPC 2016, Hyderabad, India, December 19-22, 2016*, 2016, pp. 62–71. [Online]. Available: http://dx.doi.org/10.1109/HiPC.2016.017

[27] 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)*, Sept 2017, pp. 1–7.

[28] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press Professional, Inc., 1990.

[29] Flash Center for Computational Science, "FLASH User's Guide: Version 4.4," http://flash.uchicago.edu/site/flashcode/user_support/flash4_ug_4p4.pdf, 2016.

[30] 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.

[31] P. Fischer, J. Lottes, S. Kerkemeier, O. Marin, K. Heisey, A. Obabko, E. Merzari, and Y. Peet, "Nek5000 User Documentation," http://nek5000.github.io/NekDoc/Nek_users.pdf, Argonne National Laboratory, Tech. Rep. ANL/MCS-TM-351, 2015.

[32] J. W. L. Paul F. Fischer and S. G. Kerkemeier, "nek5000 Web page," 2008, http://nek5000.mcs.anl.gov.

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

[34] D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Commun. ACM*, vol. 34, no. 4, pp. 46–58, Apr. 1991. [Online]. Available: http://doi.acm.org/10.1145/103085.103090

[35] J. Woodring, S. Mniszewski, C. Brislawn, D. DeMarle, and J. Ahrens, "Revisiting Wavelet Compression for Large-Scale Climate Data using JPEG 2000 and Ensuring Data Precision," in *2011 IEEE Symposium on Large Data Analysis and Visualization*, Oct 2011, pp. 31–38.

[36] L. Belmon, H. Benoit-Cattin, A. Baskurt, and J.-L. Bougeret, "Lossy compression of scientific spacecraft data using wavelets. Application to the CASSINI spacecraft data compression," *Astronomy & Astrophysics*, vol. 386, no. 3, pp. 1143–1152, 2002. [Online]. Available: https://doi.org/10.1051/0004-6361:20020225

[37] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of Lossy Compression for Application-Level Checkpoint/Restart," in *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 914–922. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2015.67

[38] S. Li, S. Sane, L. Orf, P. D. Mininni, J. Clyne, and H. Childs, "Spatiotemporal wavelet compression for visualization of scientific simulation data," *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 216–227, 2017.

[39] B.-L. Yeo and B. Liu, "Volume Rendering of DCT-Based Compressed 3D Scalar Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 1, pp. 29–43, Mar 1995.

[40] V. Ratnakar and M. Livny, "RD-OPT: an efficient algorithm for optimizing DCT quantization tables," in *Data Compression Conference, 1995. DCC '95. Proceedings*, Mar 1995, pp. 332–341.

[41] D. Tao, S. Di, H. Guo, Z. Chen, and F. Cappello, "Z-checker: A framework for assessing lossy compression of scientific data," *The International Journal of High Performance Computing Applications*, vol. 0, no. 0, p. 1094342017737147, 0. [Online]. Available: https://doi.org/10.1177/1094342017737147

[42] A. Wegener, "Universal numerical encoder and profiler reduces computing's memory wall with software, fpga, and soc implementations," in *2013 Data Compression Conference*, March 2013, pp. 528–528.

[43] D. Laney, S. Langer, C. Weber, P. Lindstrom, and A. Wegener, "Assessing the Effects of Data Compression in Simulations Using Physically Motivated Metrics," in *2013 SC - International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2013.

[44] Message Passing Forum, "MPI: A Message-Passing Interface Standard," Knoxville, TN, USA, 1994.

[45] J. Li, W.-k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netCDF: A High-Performance Scientific I/O Interface," in *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, ser. SC '03. New York, NY, USA: ACM, 2003, pp. 39–. [Online]. Available: http://doi.acm.org/10.1145/1048935.1050189

[46] The HDF Group. (1997-2017) Hierarchical Data Format, version 5. http://www.hdfgroup.org/HDF5/.