

Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model

Xin Liang

University of California, Riverside
Riverside, CA
xlian007@ucr.edu

Dingwen Tao

University of Alabama
Tuscaloosa, AL
tao@cs.ua.edu

Sheng Di*

Argonne National Laboratory
Lemont, IL
sdi1@anl.gov

Sihuan Li

University of California, Riverside
Riverside, CA
sli049@ucr.edu

Zizhong Chen

University of California, Riverside
Riverside, CA
chen@cs.ucr.edu

Bogdan Nicolae

Argonne National Laboratory
Lemont, IL
bnicolae@anl.gov

Franck Cappello

Argonne National Laboratory
Lemont, IL
cappello@mcs.anl.gov

ABSTRACT

With the ever-increasing volumes of data produced by today's large-scale scientific simulations, error-bounded lossy compression techniques have become critical: not only can they significantly reduce the data size but they also can retain high data fidelity for postanalysis. In this paper, we design a strategy to improve the compression quality significantly based on an optimized, hybrid prediction model. Our contribution is fourfold. (1) We propose a novel, transform-based predictor and optimize its compression quality. (2) We significantly improve the coefficient-encoding efficiency for the data-fitting predictor. (3) We propose an adaptive framework that can select the best-fit predictor accurately for different datasets. (4) We evaluate our solution and several existing state-of-the-art lossy compressors by running real-world applications on a supercomputer with 8,192 cores. Experiments show that our adaptive compressor can improve the compression ratio by 112~165% compared with the second-best compressor. The parallel I/O performance is improved by about 100% because of the significantly reduced data size. The total I/O time is reduced by up to 60X with our compressor compared with the original I/O time.

KEYWORDS

Error-Bounded Lossy Compression, Rate Distortion, Data Dumping/Loading, Compression Performance

*Corresponding author: Sheng Di, Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SC '19, November 17–22, 2019, Denver, CO, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6229-0/19/11...\$15.00

<https://doi.org/10.1145/3295500.3356193>

ACM Reference Format:

Xin Liang, Sheng Di, Sihuan Li, Dingwen Tao, Bogdan Nicolae, Zizhong Chen, and Franck Cappello. 2019. Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3295500.3356193>

1 INTRODUCTION

Today's large-scale scientific simulations on leadership supercomputers produce extremely large amounts of data such that data dumping/loading has become a major bottleneck. For instance, the Hardware/Hybrid Accelerated Cosmology Code (HACC) [16] can simulate 1~10 trillion particles in one simulation [45], producing up to 220 TB of data for each snapshot, for a total of 22 PB of data if there are 100 snapshots during the simulation. Even considering a sustained bandwidth of 500 GB/s, the I/O time will still exceed 10 hours, which is prohibitive. To address this issue, the researchers generally output the data by decimation, in other words, storing one snapshot every K time steps in the simulation. This process definitely degrades the temporal constructiveness of the simulation and also loses valuable information for postanalysis. Error-controlled lossy compression techniques have been proposed as a better solution than the simple decimation method; these techniques reduce the data size significantly while guaranteeing that the distortion of compression data is acceptable [38]. Moreover, they usually lead to much higher compression ratios given the same distortion, as demonstrated in [22].

In the past decade, several error-controlled lossy data compressors (including [1, 2, 4, 10, 12, 21, 27, 29, 33, 36, 38]) have been developed to significantly reduce the scientific data size for different purposes [6]. These lossy compressors can be classified into two groups, based on how they decorrelate the original data. The first group of compressors [4, 10, 27, 33] use a transform-based model that leverages invertible transforms for the decorrelation. The second group of compressors [12, 21, 29, 36, 38] use a prediction-based

model that leverages various prediction methods for the decorrelation. Generally speaking, no compression model can always outperform the others. Even for the same dataset, the best-fit model may differ depending on distortions. Hence, the method must be carefully selected at runtime.

In this work, we focus on the significant improvement of compression quality over the existing state-of-the-art lossy compression techniques. This task raises the following challenges. (1) Scientific simulations may produce vast volumes of data with largely different characteristics, such that none of the existing lossy compression techniques can work well stably on all datasets. (2) Designing a lightweight, adaptive framework that can always choose the best compressor is nontrivial because many lossy compressors exist each with distinct design principles. Moreover, comparing the compression quality between any two lossy compressors is nontrivial because, given the same error bound, one compressor may have a higher compression ratio with higher overall precision—such as the peak signal-to-noise ratio (PSNR)—than the other compressor has, leading to a dilemma in making a choice. (3) Further improving any of these compressors is nontrivial because each has an elaborate design and optimized implementation. ZFP, for instance, adopts an optimized transformation method by extensive explorations on various transforms and couples it with embedded encoding to provide fast and efficient lossy compression.

In our solution, we improve the lossy compression quality significantly, especially for cases with relatively high compression ratios, by combining a prediction-based model and a transform-based model in an error-bounded prediction-based lossy compression framework. Specifically, we leverage the multidimensional transform in the transform-based model as a predictor in the prediction-based model, taking advantage of both models. We then optimize the two predictors, followed by an online adaptive selection approach. To distinguish the predictors in the final framework from the two initial models, we use the terms data-fitting predictor (for the predictor used in the prediction-based model) and transform-based predictor (for the proposed and improved predictor inspired by the transform-based model) throughout the paper. Our contributions are summarized as follows.

- We improve the coefficient-encoding efficiency for the data-fitting predictor such that the compression ratio is improved significantly for relatively high-compression cases.
- We design a transform-based predictor by adopting the transform techniques of transform-based models in the prediction stage of prediction-based models. This is the first such design, to the best of our knowledge.
- We optimize the encoding strategy for the transform-based predictor, significantly improving the compression ratio for cases requiring relatively high compression ratios.
- We develop a best-fit predictor selection algorithm that can automatically select the best predictor between the data-fitting one and transform-based one in the compression.
- We conduct parallel experiments by running three well-known scientific simulation datasets on a supercomputer with up to 8,192 cores. Experiments show that our lossy compressor can improve the compression ratio by 112~165% compared with the second-best lossy compressor. With our

compressor, the total I/O time can be reduced to only half of the I/O time with the second-best lossy compressor. The total I/O performance is improved by 60X compared with the original I/O performance without compression.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we formulate the research problem. In Section 4, we describe our optimization strategies and propose an adaptive solution to select the best-fit model. In Section 5, we present the evaluation results from using three real-world simulation datasets on a supercomputer. In Section 6, we present our conclusion and discuss our future work.

2 RELATED WORK

To mitigate the storage burden and I/O bottleneck, researchers have developed many data compressors. Lossless compressors [3, 5, 9, 11, 17, 49, 51] are designed to guarantee that the decompressed data is valid, but they are not suitable for significantly reducing the scientific simulation data size because of the hard-to-compress random ending mantissa bits of the floating-point values. Their compression ratios are usually around 2 [28, 32, 34], which is far from that desired for large-scale scientific simulations running on modern high-performance computing (HPC) systems [6, 14].

Error-bounded lossy compressors have been developed as an important option for the scientific simulations to significantly reduce the data size while guaranteeing that the loss of data respects user-defined error bounds. Two state-of-the-art lossy compression models exist: prediction-based models [7, 12, 15, 21, 25, 29, 38] and transform-based models [10, 23, 24, 27, 33, 42, 46]. The typical related lossy compressors are described as follows.

- SSEM [33] is a wavelet-transform-based lossy compressor designed to improve the overall checkpoint/restart performance of climate simulations.
- ISABELA [21] is a prediction-based lossy compressor designed mainly to save storage space, but it suffers from a fairly low compression/decompression rate because of the expensive sorting operations in the compressor.
- SZ [12, 38] is a prediction-based lossy compressor that includes four key steps: data prediction, linear-scaling quantization, customized variable-length encoding, and lossless compression using gzip [11] or zstd [51]. Prior work [13, 20, 25, 26, 37, 40, 50] shows that SZ leads the compression quality among all the prediction-based compressors.
- ZFP [27] is a transform-based lossy compressor that compresses the dataset block by block (blocksize: 4x4 for 2D data and 4x4x4 for 3D data). Each block involves four steps: exponent alignment, fixed-point alignment, nonorthogonal block transform to decorrelate the values, and embedded encoding of the ordered coefficients one “bit plane” at a time. Recent research [27, 38] indicates that ZFP is one of the best error-controlled lossy compressors for scientific simulation datasets.
- TTHRESH [4] is a recent Tucker-decomposition-based compressor aiming at effectively compressing data under large error bounds. However, it is not practical to be applied in in-situ compression tasks due to the costly high-order singular value decomposition.

To date, several researchers have explored how to combine SZ and ZFP for better compression quality. Lu et al. [30] assessed multiple compressors using nine simulation datasets and concluded that SZ and ZFP are the two best compressors in their class. They also proposed a method to estimate the compression ratios for SZ and ZFP. That method, however, works based only on the point-wise relative error bounds (error divided by original data) instead of statistical error metrics (such as PSNR). In [41], the authors proposed an online approach to select the better strategy, SZ or ZFP, automatically in terms of PSNR. It adopts a simple sampling method designed based on SZ1.4 [38] (non-blockwise compression with Lorenzo predictor [19]) but is not suitable for the block-wise, multi-algorithm design in SZ2.0 [26], which is critical for fine-grained optimization. Moreover, its estimation of the bit rate for ZFP overlooks the metadata used by embedded coding, which may cause a large estimation error in cases with relatively large user-defined error bounds. Unlike these two works, we carefully analyze the two compressors from an algorithmic perspective, optimize the coefficient-encoding methods for both the data-fitting predictor and the transform-based predictor, and design a lightweight method to select the best strategy automatically. Our results show 100% I/O performance gain compared with the second-best existing compressor for large-scale simulations.

3 PROBLEM FORMULATION

In this paper, we focus on how to improve the compression quality under the restrictions of error-bounded lossy compression. Here we mainly use PSNR instead of the error bounds to assess the compression quality, because domain scientists often care more about the overall statistical errors, especially for visualization purposes. Specifically, our objective is to ensure that the decompressed data follow the error-bounding requirements and to optimize the rate distortion metric (i.e., statistical errors), while incurring little degradation on the compression speed.

Rate distortion is one of the most important metrics to assess lossy compression quality. The *rate* here is short for bit rate (denoted τ), which refers to the mean number of bits used to represent one data point after compression. We define the compression ratio (denoted R) to be the ratio of the original raw data size to the compressed data size. For a floating-point dataset, we have following equations: $\tau = \frac{32}{R}$, based on single precision, and $\tau = \frac{64}{R}$, based on double precision. Obviously, the lower the bit rate, the higher the compression ratio.

To assess the *data distortion*, we use PSNR as follows:

$$PSNR = 20 \cdot \log_{10} (value_range) - 10 \cdot \log_{10} (MSE), \quad (1)$$

where MSE stands for mean-squared error. This formula has been widely used in the community [27, 29, 38, 39], because a higher PSNR generally indicates better visual quality or higher overall precision. In this case, given a dataset with N floating-point data values, the research problem can be formulated as follows,

$$\min_{\tau} (\tau) \quad s.t. \quad |d_i - d'_i| \leq e, \forall i = 1, 2, \dots, N \quad (2)$$

where $\{d_1, d_2, \dots, d_N\}$ and $\{d'_1, d'_2, \dots, d'_N\}$ refer to the original raw data values and decompressed data values, respectively.

I/O performance on parallel file systems is also an important evaluation metric when lossy compressors are used. Specifically,

we also target significantly reducing the overall data-dumping time (dumping data to parallel file systems) and the data-loading time (loading data from parallel file systems) for large-scale parallel executions. In addition to the optimization of the rate distortion within an error-bounded setting, the compression/decompression time are also essential for I/O performance, because the data-dumping time is the compression time plus the compressed data-writing time and the data-loading time is the compressed data-reading time plus the decompression time for lossy/lossless compressors.

4 OPTIMIZED, ADAPTIVE ERROR-BOUNDED LOSSY COMPRESSION TECHNIQUE

In this section, we propose a novel, adaptive lossy compression method based on the prediction-based model, which can improve the lossy compression quality significantly over that of existing state-of-the-art compressors. The fundamental idea is to improve the prediction accuracy by leveraging both data-fitting predictors (such as the Lorenzo predictor [19]) from prediction-based models and transform-based predictors from transform-based models. The design motivation comes from the fact that these two prediction models are particularly effective on different datasets or different error bounds, as we observed in numerous experiments based on large-scale simulation datasets.

We present a workflow of our adaptive error-controlled lossy compressor in Fig. 1. Highlighted boxes indicate the key modules to be described in the following text. The arrows in this figure indicate the execution order of each operation. We note that only one of the dashed boxes in step 4 is executed at runtime because we select the better solution of the two. In the first step we use a lightweight blockwise data sampling technique that samples the data along the diagonal of the datasets (with a granularity of block size 8, e.g., 8x8x8 for 3D datasets). The most critical parts are steps 2~4, which are also the key novelty of our new design. Specifically, we estimate the overall rate distortion for the two types of predictors based on the data sampled in the previous step, respectively. Our estimation method covers the whole compression procedure for each type of predictor, including the efficiency of data prediction, the effectiveness of the corresponding coefficient compression, and the error-bounded quantization and encoding algorithms. We also optimize the strategies of encoding/compressing coefficients for both predictors, which will be detailed later. Step 5 involves error-bounded quantization and encoding, as a result, the overall compression method can strictly respect the user-specified error bound; details can be found in previous work [38].

4.1 Optimization of Lossy Compression with Data-Fitting Predictor

For the data-fitting predictor, we adopt the existing hybrid predictor used by SZ2.0 [26], and we propose an improved coefficient-compression strategy that can increase the compression ratio for high-compression cases. In what follows, we introduce the hybrid prediction model first and then describe our improvement strategy.

4.1.1 Hybrid Design in SZ2.0. In SZ2.0, the whole dataset is split into non-overlapped blocks (e.g., multiple 6x6x6 cubes for the 3D dataset), and the prediction method—either a Lorenzo predictor

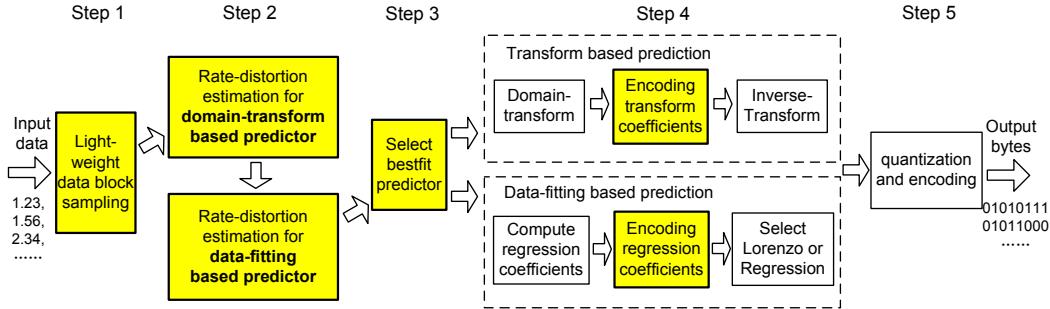


Figure 1: Steps of Our Adaptive Error-Bounded Lossy Compressor

[19] or a linear regression method—is selected in each block by a sampling method. The former method is the same as the prediction method proposed in [38]. In the latter method, a linear hyperplane ($f(i, j, k) = \beta_0 + \beta_1 i + \beta_2 j + \beta_3 k$, where i, j, k are the indices of the data points), is constructed to fit the data in each block by using a least-squares linear regression method. Specifically, the four coefficients $\beta_0, \beta_1, \beta_2$, and β_3 (as per block) are calculated by letting the partial derivatives of the squared error loss function $\sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \sum_{k=0}^{n_3-1} (\beta_0 + \beta_1 i + \beta_2 j + \beta_3 k - f_{ijk})^2$ be equal to 0. The constructed linear hyperplane is then used to predict the data values in the data block. In our implementation, we set the block size to 8x8x8, since it is neither too small for SZ to avoid high coefficient overhead (usually requiring 6+) nor indivisible by the block size required by ZFP (4) so that the efficiency of the two types of predictors on the same block is comparable.

4.1.2 Improved Regression Coefficient Compression. To improve the compression ratio significantly for high-compression cases in the data-fitting predictor, we propose a more efficient coefficient compression strategy for the linear regression prediction. In SZ 2.0 [26], all the regression coefficients were collected in terms of their types and then compressed by an IEEE-754-binary analysis. Compressing each group of coefficients is similar to the unpredictable data compression method used in SZ [12]. Four steps are involved: (1) calculate the value range and median value based on the coefficients in this group, (2) normalize all coefficients by taking away the median value from each data value, (3) encode the XOR-leading zero bytes using a 2-bit code for the data values that have the same leading bits with their preceding adjacent values, and (4) truncate the insignificant mantissa bits based on the error bound. Although this approach alleviates the required storage for the regression coefficients, it cannot help much when the error bound is large, because of the limitation of the XOR encoding.

In our new solution, we compress these coefficients based on the relatively high smoothness of the coefficients across the adjacent blocks selecting the linear-regression method. Specifically, we predict the current coefficient according to its adjacent coefficients of the same type, and we perform the linear quantization on the difference, followed by the Huffman encoding algorithm. That is, we apply a separate compression pipeline (except the lossless compression) to these coefficients besides the data points, at a cost of $\frac{4}{n_1 n_2 n_3}$ overhead since 4 coefficients out of $n_1 n_2 n_3$ data points in the block need to be compressed. Since the blocks selecting the linear

regression may not be consecutive, the multidimensional prediction mechanism in the Lorenzo predictor cannot be used anymore to discard the unused regression coefficients. To overcome this issue, we first linearize the regression coefficients in the blocks selecting linear regression and perform the 1D prediction. If many blocks use regression, this method will lead to a high compression ratio because of the high smoothness in adjacent blocks. If only a small fraction of blocks use regression, the performance will degrade because the prediction cannot be accurate in this case. At this time, however, the data would be hard to compress, and the number of coefficients to be stored is small. Thus, the regression coefficients would take only a small percentage of storage, making little difference to the final rate distortion. A sample visualization of the regression coefficients (Hurricane Uf48, value-range-based error bound $5e - 3$) is displayed in Fig. 2(a), when 86% of the blocks are choosing linear regression. The black pixels indicate the blocks selecting the Lorenzo predictor. From this figure, we can observe that the regression coefficients are smooth, from which the prediction mechanism would definitely benefit.

A critical parameter for this approach is the error bound for the regression coefficients. We choose it so that the deviation caused by the decompressed regression coefficients (compared with the computed regression coefficients) will be bounded by an error bound ϵ . Let $\epsilon_0, \epsilon_1, \epsilon_2$, and ϵ_3 stand for the absolute error bound for $\beta_0, \beta_1, \beta_2$, and β_3 , respectively. A sufficient condition is $\epsilon_0 = \frac{\epsilon}{4}, \epsilon_1 = \frac{\epsilon}{4(n_1-1)}, \epsilon_2 = \frac{\epsilon}{4(n_2-1)}$, and $\epsilon_3 = \frac{\epsilon}{4(n_3-1)}$ because it guarantees $|\beta'_0 + \beta'_1 i + \beta'_2 j + \beta'_3 k - (\beta_0 + \beta_1 i + \beta_2 j + \beta_3 k)| \leq |\epsilon_0| + |\epsilon_1(n_1-1)| + |\epsilon_2(n_2-1)| + |\epsilon_3(n_3-1)| \leq \epsilon$, where $\beta'_0, \beta'_1, \beta'_2$, and β'_3 denote the decompressed regression coefficients.

This method provides a solution to bound the prediction error, but how to set the error bound ϵ is still problematic. One must make a tradeoff between the efficiency of regression coefficients compression and the accuracy of regression prediction. A large error bound will lead to higher compression ratios for the regression coefficients, while degrading the overall compression ratio and/or PSNR since the prediction is not as accurate as before. On the contrary, a small error bound could guarantee the deviation of regression prediction at the cost of a lower compression ratio for the coefficients. We explore the error setting relative to the user-set error bound via an empirical way and the results of two representative fields of the hurricane datasets are shown in Fig. 2(b) and Fig. 2(c) when the value-range-based error bound is set to

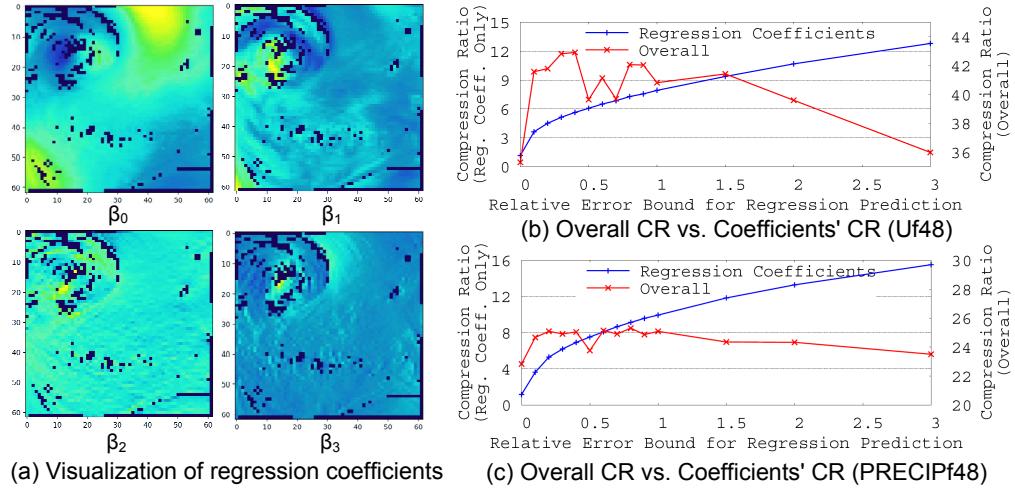


Figure 2: Analysis of Regression Coefficients on the Hurricane Simulation

5e – 3. In the figure, 86% and 66% of the blocks select regression for the given settings, respectively. Both figures indicate that the compression ratio of the regression coefficients keeps increasing while the overall compression ratio goes up and then down as the error bound gets looser, a result consistent with our analysis. We also note that the PSNR also decreases slowly with the looser error bound, from 51.85 to 50.94 (Uf48) and 52.62 to 51.11 (PRECIPf48). Therefore, we choose $\epsilon = 0.1$ as our default setting, because it has a reasonable overall compression ratio and PSNR while guaranteeing that the regression prediction is relatively accurate.

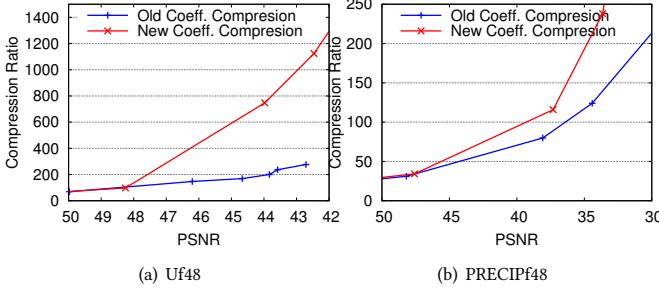


Figure 3: Ratio Distortion of Coefficients Compression on Hurricane Simulation

Figure 3 shows the improvement of the compression ratio based on our new coefficient compression method compared with the XOR encoding used by SZ 2.0 (denoted by old coeff. compression) for two example fields from the hurricane simulation. We can clearly observe that our new solution improves the compression ratio by up to 200~300% in relatively high-compression cases.

4.2 Optimization of Lossy Compression with Transform-Based Predictor

Another critical contribution in this work is that we develop and optimize a transform-based predictor based on transform-based

compression models. This contribution can significantly improve the compression quality in many cases.

Transform-based models usually leverage a multidimensional transform for the purpose of value decorrelation [27], followed by some encoding algorithm. Embedded encoding is one of the most efficient encoding strategies. The key idea of our optimization is to improve the coefficient encoding in the embedded encoding stage for transform-based predictors. We adopt ZFP's nonorthogonal transform [27] as our transform-based predictor because its decorrelation efficiency has been confirmed to be more efficient than that of other transforms such as a discrete cosine transform or wavelet transform. Our approach is a generic solution that can also be applied to other types of transforms.

We apply the transform-based predictor on each data block with block size of 4 (e.g., 4x4x4 for 3D data), following three steps: (1) perform nonorthogonal transform to convert the 64 data points to 64 coefficients, (2) perform our improved coding strategy to compress the coefficients, and (3) predict each data value by the inverse nonorthogonal transform based on the reconstructed coefficients. Step 2 is the most critical in our improvement, and it has two key issues to resolve: developing an efficient coefficient compression method and estimating the optimal error setting for the transform-based predictor.

4.2.1 Efficient Transform Coefficient Compression Method. In what follows, we illustrate in Fig. 4 the classic embedded encoding proposed in [27] and then describe our improvement method. This classic coding strategy first reorders the transform coefficients from the upper left corner (highest energy in the transformed domain) to the bottom right corner (lowest energy in the transformed domain) such that these coefficients will be roughly sorted in a descending order automatically. Then, it divides the 64 coefficients into nine groups, according to the group sizes displayed in Table 1 (see second row). Next, it computes the max bit plane (in terms of the user-defined error bound and max value in current block), after which the bit planes located before the max bit plane (called *significant bit planes*,

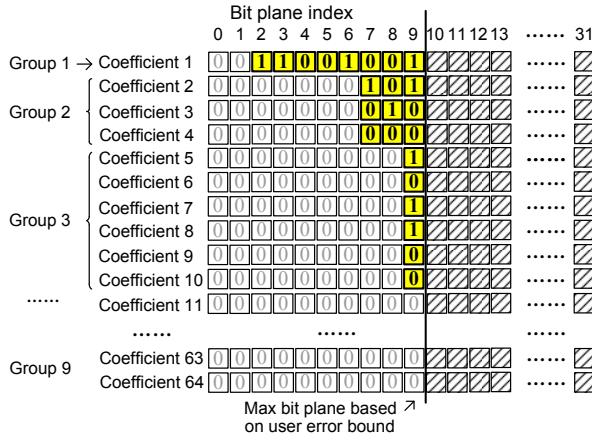


Figure 4: Embedded Coding (shaded bits would be dropped)

bits 0–9 in Fig. 4) will be compressed one by one losslessly. Since there are many zero bits due to the previous reordering of the coefficients, only a few bits (called *encoding bits*) that are organized in groups need to be stored (as highlighted in Fig. 4). In addition, three more types of metadata need to be stored: (1) *sign bits*, used to indicate the sign for the nonzero coefficients; (2) *group bits*, used to mark the grouping outlines; and (3) *skip bits*, used to indicate the ending bit of the current bit plane. As shown in Fig. 4, for example, group 1 has 8 encoding bits, 1 sign bit, 1 group bit, and 5 skip bits; group 2 has 9 (=3×3) encoding bits, 2 sign bits (because one data is 0), 1 group bit, and 2 skip bits; and group 3 has 6 (=1×6) encoding bits, 3 sign bits, 1 group bit, and 1 skip bit. In this example, other groups are not recorded because they are all zeros. Therefore, the total number of bits used to encode this block of data is 15+14+11=40. Note that from among the 40 bits, the number of encoding bits is 9+8+6=23, which means that 17 bits belong to metadata. They cannot be overlooked, as is validated by Table 1, because of the large difference between the average overall bit rate (see row 4) and the average encoding bit rate (see row 3).

A critical drawback of this embedded encoding method is the significantly uneven compression quality for different coefficients. As demonstrated in Table 1, the first and the second groups of coefficients are compressed poorly—only 2x for coefficients in the first group and 7x for those in the second group, far less than desired given the error bound. Nevertheless, these two groups of coefficients take up more than 75% of the storage space in the compressed data, significantly affecting the final compression ratio.

We propose a novel encoding algorithm that can significantly improve the coding efficiency. The key idea comes from our observation that the coefficients are still consecutive across blocks such that we can improve the compression efficiency by taking advantage of this feature. As an example, we demonstrate in Fig. 5 the visualization of the first four sets of transform coefficients (128x128) aggregated across all blocks for the slice 300 (a 512x512 image) of the NYX velocity_x field. In particular, the first coefficients extracted from all the blocks construct an image that looks much like the original raw data image (the left image in Fig. 5). The second, third, and fourth transform coefficients across all blocks

also have a relatively high spatial continuity, based on which we can further improve the coefficients' compression ratios.

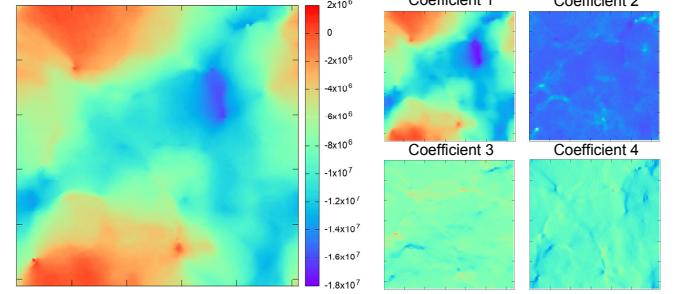


Figure 5: Visualizing Raw Data and Transform Coefficients (NYX:Velocity x)

In our solution, we adopt the data-fitting compression method (see Section 4.1) to compress/encode these coefficients with the same error bound as the traditional embedded encoding. We call it *SZ encoding* in this section. The groupwise average bit rates of these coefficients are displayed and compared with those of embedded encoding in Fig. 6(a). This figure illustrates an interesting trend of the groupwise bit rates on the two encoding strategies: the SZ encoding outperforms the embedded encoding on the first two groups, while the embedded encoding is better on the other groups. These results inspire us to seek a hybrid approach selecting SZ encoding for the first few groups and switching to embedded encoding for the remaining groups.

We use a sampling approach to determine the appropriate number of coefficients to be compressed by SZ encoding. We sample the data blocks on the diagonal of the dataset to involve data with different positions by minimal sampling rate. Specifically, we divide the whole dataset into $\frac{N_1}{N_m} \times \frac{N_2}{N_m} \times \frac{N_3}{N_m}$ subblocks (each being a $N_m \times N_m \times N_m$ cube), where $N_m = \min(N_1, N_2, N_3)$, and N_i refer to the dimension sizes (i.e., number of elements along the dimension i). Then, we sample the four diagonals in each subblock by the granularity of 8x8x8 blocks,¹ which will glean $4 * \frac{N_m}{8} * \frac{N_1 N_2 N_3}{N_m^3} * 8^3 = \frac{256 N_1 N_2 N_3}{N_m^2}$ data points, introducing small overhead to the dataset with the comparable dimension sizes (e.g., only 0.1% for the NYX dataset since $N_1=N_2=N_3=512$). In fact, other sampling methods such as uniform sampling could work equally well as long as they are done in block granularity. But this diagonal sampling method could lead to reasonable result with quite small sampling rate when the dimension sizes are close.

The sampled data will be transformed to the coefficients, which will be used to estimate the bit rates of the two encoding approaches by our designed estimation method. For the embedded encoding approach, its groupwise number of encoded bits can be computed precisely. The encoding bits of a data point can be computed exactly by subtracting the max bit plane by the position of the first nonzero element. The sign bits of a group can be approximated as the number of coefficients whose encoding bits have at least one nonzero bit. The group bit of one group is incremented by 1 as long as there exhibits one coefficient with non-all-zero encoding bits.

¹We choose the granularity of 8x8x8 blocks in order to be consistent with the sampling granularity in our designed data-fitting method (Section 4.1).

Table 1: Storage Decomposition of the Transform Coefficients on NYX Velocity_x with Value-Range-Based Error Bound 6e-3

Group ID	1	2	3	4	5	6	7	8	9
# coefficients	1	3	6	10	12	12	10	6	4
bit rate (encoding)	8.29	2.938	0.4837	0.1835	0.09343	0.05405	0.03231	0.01808	0.009857
bit rate (overall)	15.64	4.847	0.6865	0.2324	0.1148	0.06632	0.04093	0.02534	0.01462
weighted percent	39.678	36.8899	10.4497	5.8959	3.4949	2.019	1.0384	0.3857	0.1484

The skip bits, on the other hand, can be computed by taking away the length of the encoding bits of the next group from the length of the encoding bits of the current group. Therefore, we can estimate the overall bit rate of each group for the embedded encoding accurately. For the estimation of SZ encoding's bit rate, we estimate the Lorenzo prediction errors by computing the differences between the four top 4x4x4 subblocks and the four bottom 4x4x4 subblocks in each sampled 8x8x8 block. This is accurate enough for the bit rate estimation because it simulates the 1D Lorenzo predictor. We then compute quantization indices and use the entropy formula $br = \sum_i p_i \log p_i$ to estimate the final bit rate, since SZ adopts a Huffman encoding in the last step.

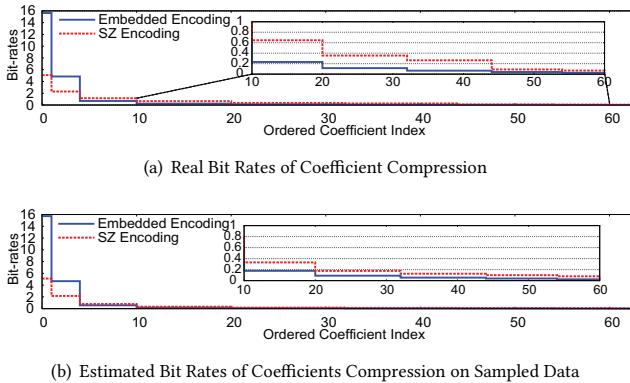
**Figure 6: Efficiency of Compressing Coefficients (NYX: Velocity_x with value-range based error bound 6e-3)**

Figure 6 presents the estimated bit rates based on the sampled data versus the real bit rates based on all data when compressing the transform coefficients over the Velocity_x field in the NYX dataset. By comparing the blue curves and red curves between the two subfigures, respectively, we can see that our estimation is accurate for both embedded encoding and SZ encoding for the first ten coefficients, which take the majority of the storage space (over 85% according to Table 1). The estimation for the remaining coefficients of the SZ encoding is not that accurate because the bit rate estimated by the entropy formula is a lower bound. We argue that it would not degrade the overall performance prominently because the difference of SZ encoding and embedded encoding is relatively small and the weighted percentages of the remaining coefficients are low for the target bit rate. If we compress the example coefficients in Fig. 4 using our hybrid approach, the average number of bits after the coefficient compression would be $5.1 + 2.3 \times 3 + 11 = 23$, obtaining 70% improvement over the original embedded encoding (40 bits).

4.2.2 Estimation of the Optimal Error Setting. In the transform-based prediction model, we still need to set an error bound to compress the transformed coefficients. Doing so involves tradeoff. Too small an error bound may degrade the prediction model to the domain-transform kernel without quantization, overpreserving the errors unexpectedly. Too large an error bound, on the other hand, will cause a large loss after the inverse-transform and thus result in low quantization efficiency.

To obtain the best tradeoff, we develop an efficient strategy by a sampling method. We reuse the same sampled data as well as the transformed coefficients to reduce the overhead. Our objective is to search for the optimal error bound ratio (i.e., the ratio of the error bound used by the transform-based predictor to the user's target error bound) in terms of the sampled data. Specifically, we set the initial error bound ratio to 1, which means that the prediction model is degraded to the domain-transform kernel because the reconstructed data must be within the user's error bound. Then, we estimate the optimal numbers of the transform coefficients for SZ encoding and embedded encoding. Next, we reconstruct the transform coefficients, based on which the inverse transform will be performed. After that, the bit rate can be estimated by adopting the remaining steps of the prediction-based compression model (i.e., error-bounded quantization and lossless compression), and the mean squared error (MSE) can be calculated. This procedure is repeated with doubled error bound ratios 8 times (we set it to 8 times because it can always cover the best tradeoff based on our experiential observations).

Our algorithm selects the best error bound setting based on the 8 rate-distortion results collected from above. Specifically, the rate distortion with an error bound ratio of 1 serves as the baseline (i.e., bit rate br_0 and MSE mse_0) and increases by 2x for each trial. We define the bit-rate decay rate as $\log \frac{br_0}{br_i}$ and the MSE increase rate as $\log \frac{mse_i}{mse_0}$ for all other rate distortion results. The optimal error setting is the one that maximizes $\frac{\log \frac{br_0}{br_i}}{\log \frac{mse_i}{mse_0}}$, because it indicates the highest bit-rate decay over the MSE increase rate.

4.3 Selecting the Best-Fit Prediction Method

The basic idea in choosing the best-fit prediction method is comparing the estimated rate distortion between the two types of predictors and selecting the better solution. A good rate distortion means the situation with high PSNR and low bit rate. Based on this, we present the pseudo-code in Algorithm 1, where $R_f(e)$ and $P_f(e)$ refer to the bit rate and PSNR, respectively, using the data-fitting-based prediction method with the error bound e . Similarly, $R_t(e)$ and $P_t(e)$ refer to the bit rate and PSNR using the transform-based prediction method with e . Given a specific error bound, our algorithm involves

two critical stages:

Stage I (line 1): Using the sampled data, estimate the optimal error bound setting (denoted ϵ_t^*) based on Section 4.2, such that the selected setting may result in the rate distortion being better than others for the transform-based predictor. We denote the corresponding bit rate and PSNR by R_t^* and P_t^* , respectively.

Stage II (line 4~22): Using the same sampled data, estimate the rate distortion for the data-fitting-based prediction model, in the vicinity of the rate distortion point (R_t^*, P_t^*) optimized in Stage I.

Algorithm 1 BEST-FIT PREDICTOR SELECTION ALGORITHM

Input: user-specified error bound (denoted as ϵ)

Output: bestfit predictor (either data-fitting or domain-transform)

```

1: Estimate the optimal rate-distortion point, denoted by  $(R_t^*, P_t^*)$ ,  

   for the transform-based prediction model (Section 4.2);  

2:  $e \leftarrow \epsilon$ ; /*Set the initial error setting to the user-specified error  

   bound*/  

3: Selection←null,  $R'_f \leftarrow$ null,  $P'_f \leftarrow$ null; /*initialization*/  

4: repeat  

5:   if  $(R_f(e) \leq R_t^* \text{ and } P_f(e) > P_t^*)$  then  

6:     Selection ← data-fitting; /*data-fitting is better*/  

7:     break;  

8:   else if  $(R_f(e) \geq R_t^* \text{ and } P_f(e) < P_t^*)$  then  

9:     Selection ← domain-transform; /*domain-transform is  

   better*/  

10:    break;  

11:   else if  $((R'_f < R_t^* \text{ and } P'_f < P_t^*) \text{ and } (R_f(e) > R_t^* \text{ and } P_f(e) > P_t^*))$   

   or  $((R'_f > R_t^* \text{ and } P'_f > P_t^*) \text{ and } (R_f(e) < R_t^* \text{ and } P_f(e) < P_t^*))$  then  

12:     Estimate  $P_f(\epsilon_t^*)$  by linear interpolation with the two rate-  

   distortion points  $(R'_f, P'_f)$  and  $(R_f(e), P_f(e))$ ;  

13:     Selection← $\arg\max(P_f(\epsilon_t^*), P_t^*)$ ; /*select with higher  

   PSNR*/  

14:     break;  

15:   else if  $(R_f(e) < R_t^* \text{ and } P_f(e) < P_t^*)$  then  

16:      $R'_f \leftarrow R_f(e)$ ,  $P'_f \leftarrow P_f(e)$ ; /*record the rate-distortion point*/  

17:      $e \leftarrow 2e$ ; /*double the error bound for next checking*/  

18:   else if  $(R_f(e) > R_t^* \text{ and } P_f(e) > P_t^*)$  then  

19:      $R'_f \leftarrow R_f(e)$ ,  $P'_f \leftarrow P_f(e)$ ; /*record the rate-distortion point*/  

20:      $e \leftarrow e/2$ ; /*half the error bound for next checking*/  

21:   end if  

22: until (Selection ≠ null)  

23: Output selection;
```

We illustrate in Fig. 7 all six cases corresponding to the if-branches of Algorithm 1. The algorithm keeps searching for the error bound setting (with exponential increase/decrease) for the data-fitting-based prediction model (in the cases (e) and (f)) until we can judge which predictor is better (via the cases (a)–(d)). That is, the subfigures (a)–(d) are the termination condition, corresponding to lines 5–7, lines 8–10, and lines 11–14, respectively, in the algorithm, while the subfigures (e) and (f) correspond to lines 15–21, meaning that the error bound needs to be doubled or halved to keep searching toward the target estimated optimal point (R_t^*, P_t^*) .

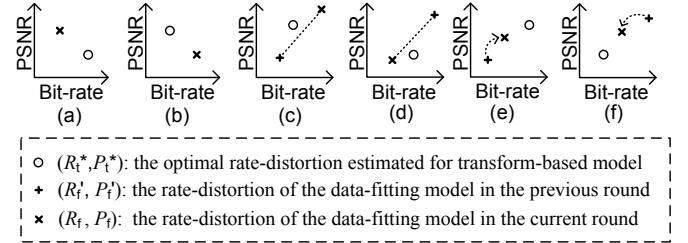


Figure 7: Illustration of Best-Fit Predictor Selection Algorithm

5 PERFORMANCE EVALUATION

In this section, we present the performance evaluation results to show the accuracy of our best-fit selection algorithm and the rate-distortion graph of our adaptive compressor compared with other state-of-the-art lossy compressors. We also evaluate the parallel I/O performance by launching thousands of MPI ranks to dump/load real-world datasets on a supercomputer with different scales. We show that our optimized lossy compressor can significantly improve the rate distortion and also outperform the second-best compression method by about 100% with respect to the overall I/O performance.

5.1 Experimental Setting

We conducted our experimental evaluation on the Bebop supercomputer [35] at Argonne National Laboratory using 2,048~8,192 cores (i.e., 64~256 nodes, each with two Intel Xeon E5-2695 v4 processors and 128 GB of memory, and each processor with 18 cores). The storage system uses General Parallel File Systems (GPFS), which are located on a raid array and served by multiple file servers. The I/O and storage systems are typical high-end supercomputer facilities equipped with two I/O nodes. Such a system is a good case for demonstrating the I/O bottleneck when I/O is saturated, which is a common case when running extremely large-scale applications (e.g., with millions of cores) on cutting-edge supercomputers because I/O nodes are always far fewer than compute nodes. We use the file-per-process mode with POSIX I/O [48] on each process for reading/writing data in parallel.¹

The application data is from multiple domains including Hurricane Isabel climate simulation [18], NYX cosmology simulation [31], and SCALE-LETKF weather simulation [47]. Each application involves multiple simulation snapshots (or time steps). Without loss of generality, we assessed only meaningful fields with relatively large data sizes (other fields have either constant data or data sizes that are too small). Also, some datasets with closed clustered data in $[0, 1]$ are transformed to its logarithmic domain for better visualization quality, as suggested by the domain scientists. The data sizes per snapshot are 1.3 GB, 3 GB, and 6 GB per core for the three applications, respectively. Thus, the total data sizes per snapshot

¹POSIX I/O performance is close to other parallel I/O performance such as MPI-IO [43] when thousands of files are written/read simultaneously on GPFS, as indicated by a recent study [44]. We also validated that the read/write performance difference of POSIX I/O and MPI-IO is within $\pm 10\%$, when the execution scale is between 2k cores and 8k cores. POSIX I/O and MPI-IO are both widely used in scientific simulations; we choose POSIX I/O for simplicity and without loss of generality. The result about I/O performance gain under our solution with MPI-IO is actually similar to that with POSIX I/O, according to our experiments.

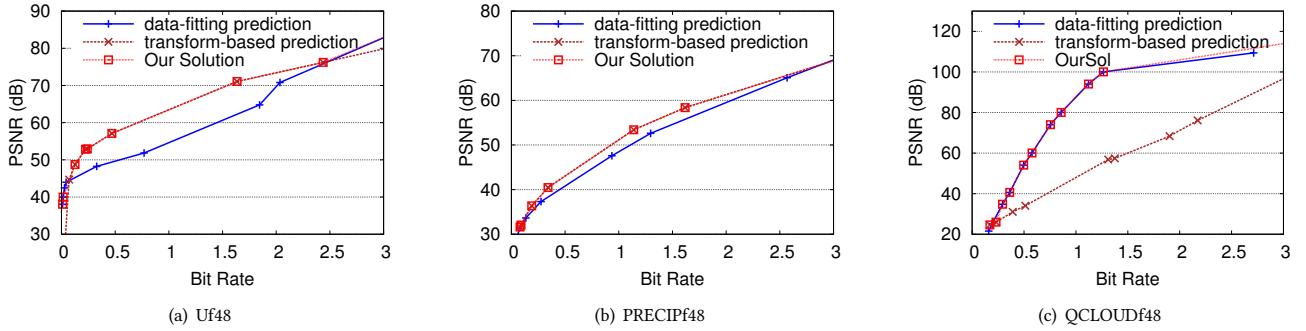


Figure 8: Assessment of Bestfit Selection Algorithm on Hurricane Isabel

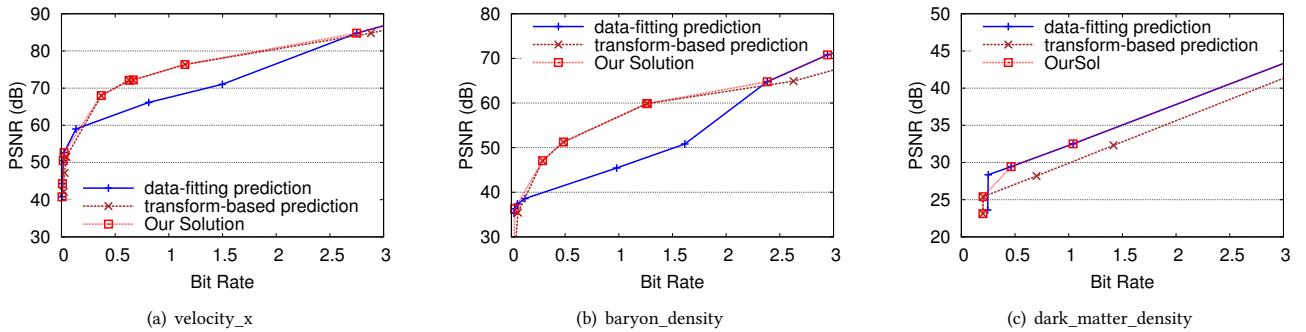


Figure 9: Assessment of Bestfit Selection Algorithm on NYX

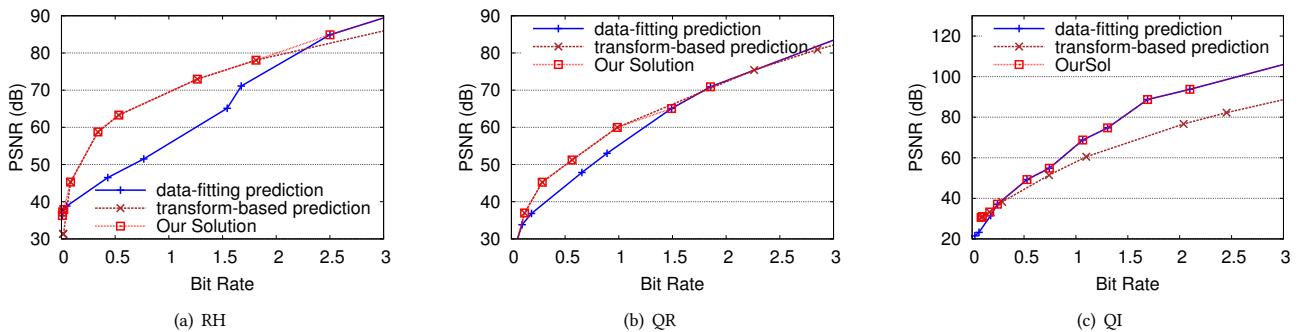


Figure 10: Assessment of Best-fit Selection Algorithm on SCALE-LETKF

are 10.4 TB, 24 TB, and 48 TB, respectively, when the execution scale is 8,192 cores in our experiment.

We focus on the improvement in rate distortion and in parallel I/O performance by leveraging our designed lossy compressor, as compared with three state-of-the-art lossy compression methods: SZ2.0 [26], ZFP0.5.4 [27], and an automatic online selection between SZ and ZFP from [41] (called AOS for short). They have been confirmed as the best in class [8, 30, 41]. Since the compression ratio will dominate the I/O performance in a large-scale environment (as discussed in Section 3), we also investigated the rate distortion of different compressors. We note that we did not evaluate error

bounds as compression quality because domain scientists care more about overall statistical errors, as mentioned in Section 3.

In what follows, we first assess our selection algorithm in terms of the overall rate distortion results, that is, the bit rate vs. data distortion (assessed via PSNR). Next we compare the overall rate distortion of our compressor with that of other state-of-the-art compressors, and we investigate the satisfactory level of data distortion by visualizing the decompressed data compared with the original raw data in high resolution. We then report on a series of parallel experiments to evaluate the I/O performance gain of our lossy compressor against other state-of-the-art compressors.

5.2 Assessment of Our Bestfit Prediction Selection Algorithm

In this subsection, we analyze the effectiveness of our best-fit prediction selection algorithm proposed in Section 4.3. We present the rate distortion curves of the three solutions based on our improved data-fitting prediction, our improved transform-based prediction, and our best-fit selection algorithm, respectively, in Figs. 8, 9, and 10. We demonstrate three fields for each application because of the limited space; other fields lead to the similar results. Based on the definitions of PSNR and bit rate given in Section 3, the more left a curve is located, the higher its overall compression quality is. We can see that our selection algorithm can always select the best-fit predictor in the given bit rate range, such that the rate distortion result is optimized for any point in this range. In Fig. 8(a), for instance, our solution selects the data-fitting predictor when the bit-rate is lower than 0.1 or higher than 2.5 and the transform-based predictor otherwise, leading to the best rate distortion for all bit rates. On the other hand, our solution is able to select the data-fitting predictor when it is better than the transform-based predictor, as shown in 8(c). Therefore, our solution can lead to a much better result when all the fields in the dataset are considered. We note that our solution does not fully overlap with the data-fitting predictor because it does not yield the same error bound (due to the selection mechanism) as the data-fitting predictor for one or two points on the switching boundary. They would have the same result, however, when the error bound in the selector is restricted to the same value.

5.3 Rate Distortion of Our Solution versus State-of-the-Art Methods

We present the rate distortion curves (bit rate vs. PSNR) for the three applications in Figs. 11, 12, and 13. In each figure, we present the overall result (with the same error bounds for all fields) in the first subfigure and demonstrate the results for two specific fields in the other two subfigures. One can clearly see that our solution always outperforms other compressors in compression quality. The key reason is that our new encoding scheme for the transform-based predictor is more efficient than the traditional embedded encoding. Also, it can automatically switch to the data-fitting predictor when the data-fitting predictor performs well as our sampling strategy can effectively select the best-fit one. Note that AOS is worse than SZ2.0 in some cases because it builds on the Lorenzo predictor in SZ1.4 and its mechanism cannot be applied to the hybrid design in SZ2.0 directly. Also note that the overall improvement on all fields for each application appears not as high as the individual gain on each field, because we can only adopt the same error bound to perform the compression for each field, such that the PSNR values across different fields have a high variance, leading to a mitigated overall PSNR. This situation would not happen when people compress the data based on the statistical distortion (PSNR) instead of the same error bound for different fields, which is a usual case in the real world.

5.4 Parallel Performance

Before performing the evaluation of parallel I/O performance with different lossy compressors, we need to determine the acceptable

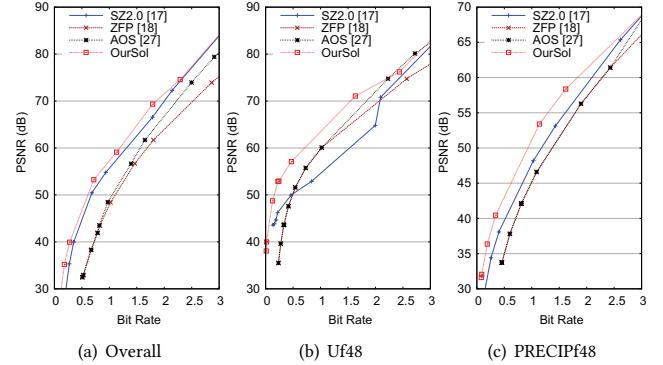


Figure 11: Rate Distortion in Hurricane Isabel Simulation

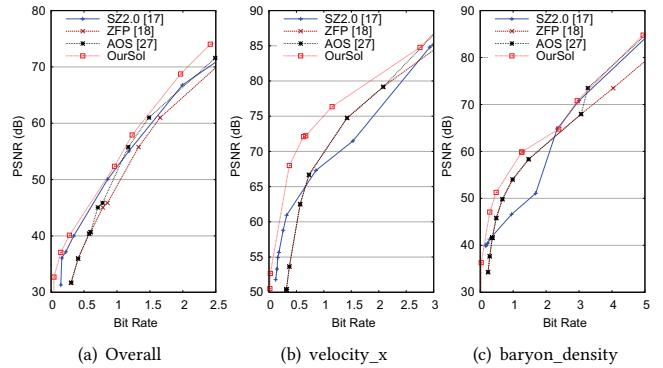


Figure 12: Rate Distortion in NYX Simulation

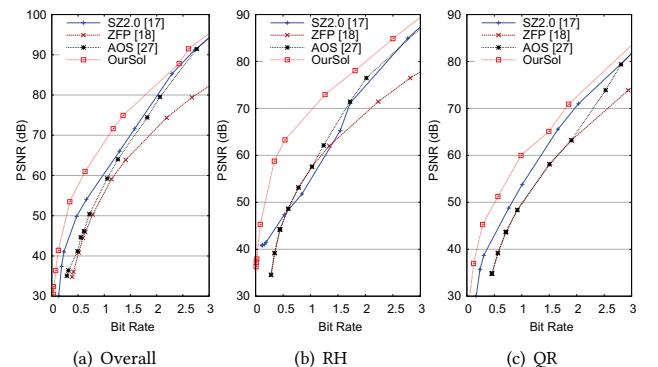


Figure 13: Rate Distortion in SCALE-LETKF Simulation

level of data distortion. To this end, we plot the data using a visualization tool for each field involved in our experiment and select the distortion level (PSNR) with a high visual quality for each field. We demonstrate two examples in Figure 14 and Figure 15, respectively. In the NYX simulation, for instance, the reconstructed data of the velocity_x field has a fairly high visual quality (even visualizing it with a higher resolution) when the PSNR is about 68, whereas its visual quality decreases with lower PSNR. By comparison, for the QS field in the SCALE-LETKF simulation, PSNR=46 is enough for

getting a good visual quality (see Fig. 15), whereas PSNR=32 causes a noticeable distortion with high resolution. In our experiments, we tune the PSNR to the same value for different lossy compressors. This value is set to 60 ~ 70 for normal data (e.g. NYX velocity_x) and 40 ~ 50 for the logarithmic data (e.g. SCALE-LETKF QS).

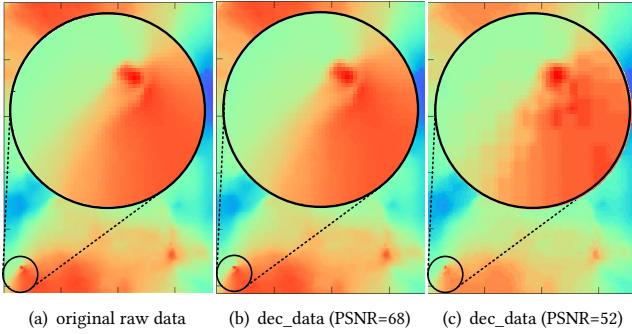


Figure 14: Visualization of NYX (velocity_x) by Comparing Raw Data and Reconstructed Data with Different PSNRs

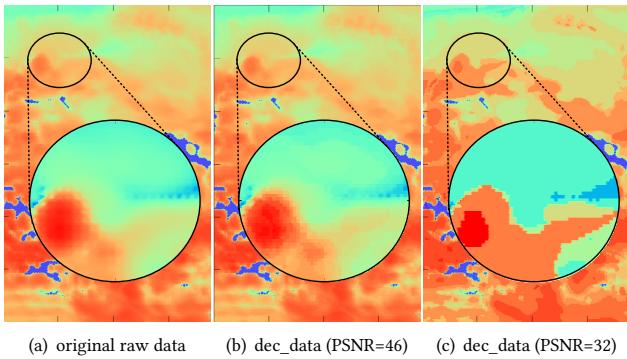


Figure 15: Visualization of SCALE-LETKF (QS) by Comparing Raw Data and Reconstructed Data with Different PSNRs

Table 2: Compression Ratios (Raw Size over Compressed Size) and Memory Overhead

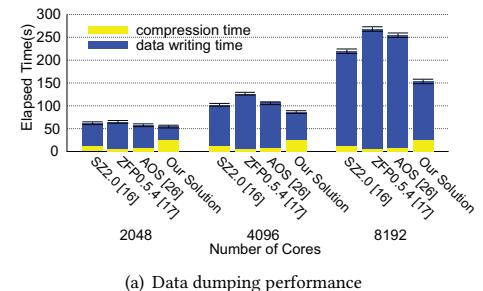
	SZ2.0	ZFP0.5.4	AOS	OurSol	OurSol_b
Hurricane Isabel	36.83	29.46 (<<1%)	30.04	54.5	51.69 (1.1%)
NYX	53.78	46.51 (<<1%)	46.51	103.7	96.92 (1.6%)
SCALE-LETKF	51.72	41.26(<<1%)	43.82	109.5	102.1 (1.1%)
Memory Overhead	100%	Constant ¹	100%	100%	Constant ¹

Table 2 presents the compression ratios when compressing the three application datasets by four compressors with the same data distortion (PSNR). The table shows that our solution leads to the highest compression ratio in all cases. In absolute terms, its compression ratio is higher than that of the others by 48~85%, 93~123%, and 112~165%, respectively. We also show the memory overhead of these different approaches. Since SZ requires extra space to store the frequencies in the Huffman tree, it incurs 100% memory overhead. On the other hand, ZFP only has constant memory overhead since it processes the data block by block. AOS follows either SZ

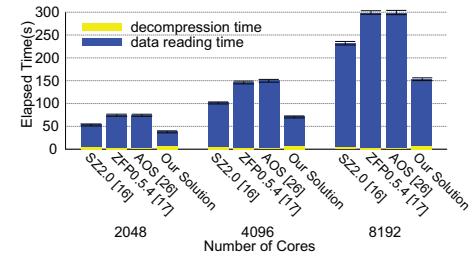
¹exact number in the brackets

or ZFP, so it has 100% memory overhead in the worst case. Our vanilla solution adopts the SZ prediction scheme as it is, leading to 100% overhead. In fact, the memory overhead can be significantly reduced by dividing the data into small blocks and performing the compression block by block, such that the memory overhead can be limited to the block size (constant). We show the result of this block-wise optimization with 128x128x128 block division (64x64x64 for Hurricane to keep around 1% memory overhead) in the last column “OurSol_b”. It has constant memory overhead with slightly lower compression ratio (less than 7%) due to the overhead of the Huffman tree in each block. But it still keeps the dominance in compression ratios, compared with other existing lossy compressors.

As discussed in Section 3, the data dumping/loading performance with lossy compression includes both I/O performance of the parallel file system and compression/decompression performance of the compressor. For large-scale executions when I/O is saturated, however, the compression ratio will dominate the overall data dumping/loading performance for weak-scaling problems, which is often the case for scientific simulations. The reason is that the I/O throughput will remain constant while the total I/O size will increase linearly with the scale. On the other hand, the compression/decompression time will not change because the workload on each rank/node will remain the same even when the scale increases because of limited memory capacity per rank/node (i.e., weak-scaling case). Therefore, the compression ratio will always be the dominant factor for the total I/O performance of current parallel systems when the scale becomes large, as confirmed in our following experiments.



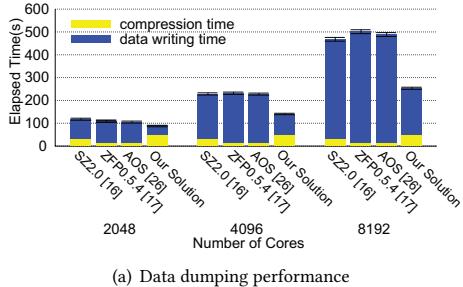
(a) Data dumping performance



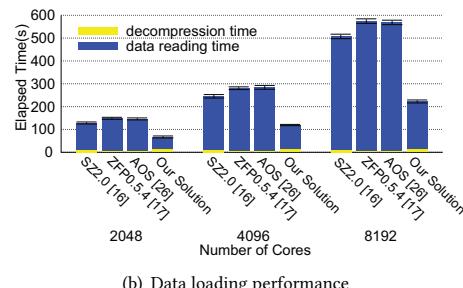
(b) Data loading performance

Figure 16: Parallel Performance on Hurricane Isabel Simulation

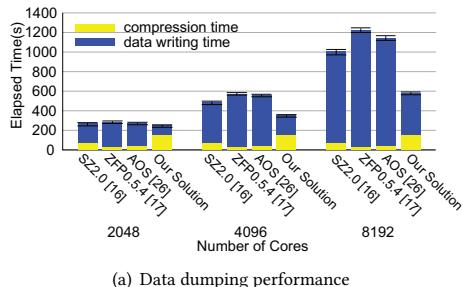
We present the overall data-dumping time and data-loading time for the three applications (Hurricane Isabel climate simulation, NYX cosmological simulation, and SCALE-LETKF weather simulation),



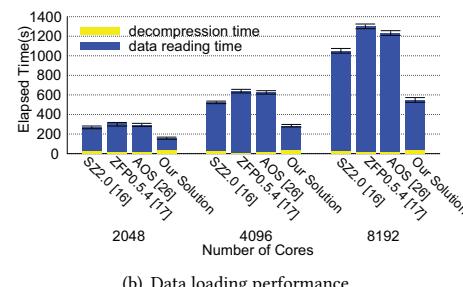
(a) Data dumping performance



(b) Data loading performance

Figure 17: Parallel Performance on NYX Simulation

(a) Data dumping performance



(b) Data loading performance

Figure 18: Parallel Performance on SCALE-LETKF Simulation

in Figs. 16, 17, and 18, respectively. Specifically, we launched 2,048 ranks (~ 8,192 ranks) to dump and load the total data in parallel. The original raw data writing times would be over 2 hours, 5 hours, and 10 hours, respectively, because of the extremely large data size (10 TB~48 TB) to process. We ran each case five times and record the compression/decompression time and writing/reading time, as well as the error bars of the total time. The three figures clearly

show that the solution with our new compressor improves the total I/O performance by 60X than the original I/O performance without any compressor. Moreover, our solution outperforms the other three solutions significantly, especially when the execution scale reaches 8,192 cores. The performance gains over other solutions are approaching the compression ratios with execution scale (or total data size), as is consistent with our analysis. For instance, for the data-dumping performance of the SCALE-LETKF application, our solution outperforms others by 9.6~15%, 39.1~64%, and 72.5~110%, respectively, which gets closer and closer to the gains on the compression ratios (112~165%). For the NYX simulation, our solution outperforms the other solutions by 17.8~30% in dumping 6 TB of data by 2,048 cores, and it increases up to 85.3~94% for dumping 24 TB data by 8,192 cores. Moreover, our solution may get further I/O performance gains over other solutions at larger execution scales when I/O is saturated, since the compression time is constant whereas the I/O increases nearly linearly with scale.

6 CONCLUSION AND FUTURE WORK

In this paper, we seek to optimize the rate distortion of error-bounded lossy compression by adopting an adaptive, hybrid prediction framework that combines the prediction-based compression model and transform-based compression model. We improve the coefficient encoding efficiency for both models and also optimize the estimation accuracy to select the best-fit predictor. We perform our evaluation using large real-world simulation datasets across different scientific domains on a supercomputer with up to 8,192 cores. The important insights are summarized as follows.

- Our adaptive predictor selector has very high accuracy in selecting the best-fit strategy between data-fitting predictor and domain-transform predictor. This approach leads to the best rate distortion graph among existing lossy compressors.
- The compression ratio of our adaptive compressor is higher than that of other state-of-the-art compressors by 112~165%, with the same level of acceptable data distortion.
- Our solution outperforms other compression solutions by about 100% when dumping/loading 10 TB~48 TB of data on a parallel file system with 8,192 cores.

We plan to further improve the compression quality by exploring more effective prediction models and coding algorithms.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant No. 1619253. This work was also supported by National Science Foundation CCF 1513201. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

- [1] Mark Ainsworth, Scott Klasky, and Ben Whitney. 2017. Compression using lossless decimation: analysis and application. *SIAM Journal on Scientific Computing* 39, 4 (2017), B732–B757.
- [2] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. 2018. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science* 19, 5–6 (2018), 65–76.
- [3] Francesc Alted. 2017. Blosc, an extremely fast, multi-threaded, meta-compressor library.
- [4] Rafael Ballester-Ripoll, Peter Lindstrom, and Renato Pajarola. 2019. TTHRESH: Tensor Compression for Multidimensional Visual Data. *IEEE transactions on visualization and computer graphics* (2019).
- [5] Martin Burtscher and Paruj Ratanaworabhan. 2009. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Trans. Comput.* 58, 1 (Jan 2009), 18–31.
- [6] Franck Cappello, Sheng Di, Sihuan Li, Xin Liang, Ali Murat Gok, Dingwen Tao, Chun Hong Yoon, Xin-Chuan Wu, Yuri Alexeev, and Frederic T Chong. 2019. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications* (2019), 1094342019853336.
- [7] Zhengzhang Chen, Seung Woo Son, William Hendrix, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. 2014. NUMARCK: machine learning algorithm for resiliency and checkpointing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 733–744.
- [8] Jong Youl Choi, Choong-Seock Chang, Julien Dominski, Scott Klasky, Gabriele Merlo, Eric Suchyta, Mark Ainsworth, Bryce Allen, Franck Cappello, Michael Churchill, Philip E. Davis, Sheng Di, Greg Eisenhauer, Stéphane Ethier, Ian T. Foster, Berk Geveci, Hanqi Guo, Kevin A. Huck, Frank Jenko, Mark Kim, James Kress, Seung-Hoe Ku, Qing Liu, Jeremy Logan, Allen D. Malony, Kshitij Mehta, Kenneth Moreland, Todd Munson, Manish Parashar, Tom Peterka, Norbert Podhorszki, Dave Pugmire, Ozan Tugluk, Ruonan Wang, Ben Whitney, Matthew Wolf, and Chad Wood. 2018. Coupling wxscale multiphysics applications: Methods and lessons learned. In *Proceedings of IEEE International Conference on eScience*.
- [9] Steven Claggett, Sahar Azimi, and Martin Burtscher. 2018. SPDP: An automatically synthesized lossless compression algorithm for floating-point data. In *the 2018 Data Compression Conference*, 337–346.
- [10] John Clyne, Pablo Mininni, Alan Norton, and Mark Rast. 2007. Interactive desktop analysis of high resolution simulations: application to turbulent plume dynamics and current sheet formation. *New Journal of Physics* 9, 301 (2007), 1–29.
- [11] L Peter Deutsch. 1996. GZIP file format specification version 4.3.
- [12] Sheng Di and Franck Cappello. 2016. Fast error-bounded lossy HPC data compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 730–739.
- [13] Sheng Di, Dingwen Tao, Xin Liang, and Franck Cappello. 2018. Efficient lossy compression for scientific data based on pointwise relative error bound. *IEEE Transactions on Parallel and Distributed Systems* 30, 2 (2018), 331–345.
- [14] Ian T. Foster, Mark Ainsworth, Bryce Allen, Julie Bessac, Franck Cappello, Jong Youl Choi, Emil M. Constantinescu, Philip E. Davis, Sheng Di, Zichao Wendy Di, Hanqi Guo, Scott Klasky, Kerstin Kleese van Dam, Tahsin M. Kurç, Qing Liu, Abid Malik, Kshitij Mehta, Klaus Mueller, Todd Munson, George Ostrovchov, Manish Parashar, Tom Peterka, Line Pouchard, Dingwen Tao, Ozan Tugluk, Stefan M. Wild, Matthew Wolf, Justin M. Wozniak, Wei Xu, and Shinjae Yoo. 2017. Computing just what you need: Online data analysis and reduction at extreme scales. In *European Conference on Parallel Processing*. Springer, 3–19.
- [15] Ali Murat Gok, Sheng Di, Alexeev Yuri, Dingwen Tao, Vladimir Mironov, Xin Liang, and Franck Cappello. 2018. PaSTRI: A novel data compression algorithm for two-electron integrals in quantum chemistry. In *IEEE International Conference on Cluster Computing (CLUSTER)*, 1–11.
- [16] Salman Habib, Vitali A. Morozov, Nicholas Frontiere, Hal Finkel, Adrian Pope, Karin Heitmann, Kalyan Kumaran, Venkatram Vishwanath, Tom Peterka, Joseph A. Insley, David Daniel, Patricia K. Fasel, and Zarija Lukic. 2016. HACC: extreme scaling and performance across diverse architectures. *Commun. ACM* 60, 1 (2016), 97–104.
- [17] David A Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101.
- [18] Hurricane ISABEL simulation data. 2019. <http://vis.computer.org/vis2004contest/data.html>. Online.
- [19] Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. 2003. Out-of-core compression and decompression of large n-dimensional scalar fields. In *Computer Graphics Forum*, Vol. 22. Wiley Online Library, 343–348.
- [20] Sian Jin, Sheng Di, Xin Liang, Jiannan Tian, Dingwen Tao, and Franck Cappello. 2019. DeepSZ: A Novel Framework to Compress Deep Neural Networks by Using Error-Bounded Lossy Compression. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 159–170.
- [21] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Seung-Hoe Ku, Choong-Seock Chang, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F Samatova. 2013. Isabela for effective in situ compression of scientific data. *Concurrency and Computation: Practice and Experience* 25, 4 (2013), 524–540.
- [22] Sihuan Li, Sheng Di, Xin Liang, Zizhong Chen, and Franck Cappello. 2018. Optimizing lossy compression with adjacent snapshots for N-body simulation data. In *2018 IEEE International Conference on Big Data*. IEEE.
- [23] Shaomeng Li, Stanislaw Jaroszynski, Scott Pearse, Leigh Orf, and John Clyne. 2019. VAPOR: A Visualization Package Tailored to Analyze Simulation Data in Earth System Science. (2019).
- [24] Shaomeng Li, Nicole Marsaglia, Christoph Garth, Jonathan Woodring, John Clyne, and Hank Childs. 2018. Data reduction techniques for simulation, visualization and data analysis. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 422–447.
- [25] Xin Liang, Sheng Di, Dingwen Tao, Zizhong Chen, and Franck Cappello. 2018. Efficient Transformation Scheme for Lossy Data Compression with Point-wise Relative Error Bound. In *IEEE International Conference on Cluster Computing (CLUSTER)*, 179–189.
- [26] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2018. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data*. IEEE.
- [27] Peter Lindstrom. 2014. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2674–2683.
- [28] Peter Lindstrom. 2017. Error Distributions of Lossy Floating-Point Compressors. *Joint Statistical Meetings* (2017), 2574–2589.
- [29] Peter Lindstrom and Martin Isenburg. 2006. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 1245–1250.
- [30] Tao Lu, Qing Liu, Xubin He, Huizhang Luo, Eric Suchyta, Jong Choi, Norbert Podhorszki, Scott Klasky, Matthew Wolf, Tong Liu, and Zhenbo Qiao. 2018. Understanding and modeling lossy compression schemes on HPC scientific data. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 348–357.
- [31] NYX simulation. 2019. <https://amrex-astro.github.io/Nyx>. Online.
- [32] Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. 2006. Fast lossless compression of scientific floating-point data. In *Data Compression Conference (DCC'06)*. IEEE, 133–142.
- [33] Naoto Sasaki, Kento Sato, Toshio Endo, and Satoshi Matsuoka. 2015. Exploration of lossy compression for application-level checkpoint/restart. In *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 914–922.
- [34] Seung Woo Son, Zhengzhang Chen, William Hendrix, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. 2014. Data compression for the exascale computing era-survey. *Supercomputing Frontiers and Innovations* 1, 2 (2014), 76–88.
- [35] Bebop supercomputer. 2019. Available at <https://www.lcrc.anl.gov/systems/resources/bebop>. online.
- [36] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. 2017. In-depth exploration of single-snapshot lossy compression techniques for N-body simulations. In *2017 IEEE International Conference on Big Data*. IEEE, 486–493.
- [37] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. 2017. In-depth exploration of single-snapshot lossy compression techniques for N-body simulations. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 486–493.
- [38] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. 2017. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 1129–1139.
- [39] Dingwen Tao, Sheng Di, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2017. Z-checker: A framework for assessing lossy compression of scientific data. *The International Journal of High Performance Computing Applications* 0, 0 (2017).
- [40] Dingwen Tao, Sheng Di, Xin Liang, Zizhong Chen, and Franck Cappello. 2018. Improving performance of iterative methods by lossy checkpointing. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 52–65.
- [41] Dingwen Tao, Sheng Di, Xin Liang, Zizhong Chen, and Franck Cappello. 2019. Optimizing Lossy Compression Rate-Distortion from Automatic Online Selection between SZ and ZFP. *IEEE Trans. Parallel Distrib. Syst.* 30, 8 (2019), 1857–1871.
- [42] David Taubman and Michael Marcellin. 2013. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated.
- [43] Rajeev Thakur, William Gropp, and Ewing Lusk. 1999. On Implementing MPI-IO portably and with high performance. In *Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems (IOPADS '99)*. ACM, New York, NY, USA, 23–32.
- [44] Andy Turner. 2019. Parallel I/O Performance. https://www.archer.ac.uk/training/virtual/2017-02-08-Parallel-IO/2017_02_ParallelIO_ARCHERWebinar.pdf. Online.
- [45] S. Crusan V. Vishwanath and K. Harms. 2019. Parallel I/O on Mira. https://www.alcf.anl.gov/files/Parallel_IO_on_Mira_0.pdf. Online.

- [46] Gregory K Wallace. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38, 1 (1992), xviii–xxxiv.
- [47] SCALE-LETKF weather model. 2019. <https://github.com/gylien/scale-letkf>. Online.
- [48] Brent Welch. 2005. POSIX IO extensions for HPC. In *4th USENIX Conference on File and Storage Technologies (FAST05)*.
- [49] Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on information theory* 23, 3 (1977), 337–343.
- [50] Xiangyu Zou, Tao Lu, Wen Xia, Xuan Wang, Weizhe Zhang, Sheng Di, Dingwen Tao, and Franck Cappello. 2019. *Accelerating Relative-error Bounded Lossy Compression for HPC datasets with Precomputation-Based Mechanisms*. Technical Report. Argonne National Lab.(ANL), Argonne, IL (United States).
- [51] Zstd. 2019. <https://github.com/facebook/zstd/releases>. Online.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We ran our software and the baselines on the bebop supercomputer with Intel MPI and Intel MKL. The system is equipped with 2 Intel Xeon E5-2695 v4 processors per node, each with 18 cores, and GPFS file system with 2 I/O nodes.

ARTIFACT AVAILABILITY

Software Artifact Availability: All author-created software artifacts are maintained in a public repository under an OSI-approved license.

Hardware Artifact Availability: There are no author-created hardware artifacts.

Data Artifact Availability: All author-created data artifacts are maintained in a public repository under an OSI-approved license.

Proprietary Artifacts: None of the associated artifacts, author-created or otherwise, are proprietary.

List of URLs and/or DOIs where artifacts are available:

<https://www.lcrc.anl.gov/systems/resources/bebop>
https://github.com/lxAltria/hybrid_lossy_compressi
→ on.git
<https://sdrbench.github.io>

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Intel Xeon E5-2695 v4 processors, 2 processors/node, 18 cores/per processor, Omni-Path Fabric Interconnect, GPFS with 2 I/O nodes

Operating systems and versions: CentOS 7 running Linux kernel 3.10.0-957.5.1.el7.x86_64

Compilers and versions: gcc 4.8.5

Libraries and versions: intel-mpi/2017.3-dfphq6k, intel-mkl/2017.3.196-v7uuj6z

Input datasets and versions: All fields in Hurricane, NYX, SCALE-LETKF

Paper Modifications: No modifications on the softwares/hardwares.

Output from scripts that gathers execution environment information.

```
LMOD_FAMILY_COMPILER_VERSION=17.0.4-74uvhji
MKLROOT=/blues/gpfs/home/software/spack-0.10.1/opt/s
→ pack/linux-centos7-x86_64/intel-17.0.4/intel-mkl
→ -2017.3.196-v7uuj6zmthzln35n2hb7i5u5ybncv5ev/comp
→ ilers_and_libraries_2017.4.196/linux/mkl
```

```
MANPATH=/blues/gpfs/home/software/spack-0.10.1/opt/s
→ pack/linux-centos7-x86_64/intel-17.0.4/intel-mpi
→ -2017.3-dfphq6kavje2olnichisvjndtridrok/compiler
→ s_and_libraries_2017.4.196/linux/mpi/man:/blues/
→ gpfs/home/software/spack-0.10.1/opt/spack/linux-
→ centos7-x86_64/gcc-4.8.5/intel-17.0.4-74uvhjiuly
→ qgvsmwyifbbuo46v5n42xc/man:/blues/gpfs/home/soft
→ ware/spack-0.10.1/opt/spack/linux-centos7-x86_64
→ /gcc-4.8.5/lmod-7.4.9-ic63herzfgw5u3na5mdtvp3nw
→ y6ojz2/lmod/lmod/share/man:::
HOSTNAME=xxxx
_ModuleTable003_=N3V1ajZ6IixbImxvYWRPcmRlcJdPTMscHJ
→ vcFQ9e30sWyJzdGF0dXMiXT0iYWN0aXZ1IixbInVzZXJ0YW1
→ iI09ImludGVsLW1rbCIsfSxbImludGVsLW1waSJdPxTbImZ
→ uI09Ii9ibHV1cy9ncGZzL2hvbWUvc29mdHdhcmUvc3BhY2s
→ tMC4xMC4xL3NoYXJ1L3NwYWNrL2xtb2QvbGludXgtY2VudG9
→ zNy140DZfNjQvaW50ZWwvMTcuMC40L2ludGVsLW1waS8yMDE
→ 3LjMtZGZwaHE2ay5sdWEiLfsiZnVsbE5hbWUiXT0iaW50Zw
→ tbXBpLzIwMTcuMy1kZnBocTZrIixbImxvYWRPcmRlcJdPTI
→ scHJvcFQ9e30sWyJzdGF0dXMiXT0iYWN0aXZ1IixbInVzZXJ
→ OYW1lI09ImludGVsLW1waSIsfSx9LG1wYXRoQT17Ii9ibHV
→ lcy9ncGZzL2hvbWUvc29mdHdhcmUvc3BhY2stMC4xMC4x
INTEL_LICENSE_FILE=28518@xxxx.xxxx.xxx.xxx
SHELL=/bin/bash
TERM=xterm-256color
HISTSIZE=1000
SSH_CLIENT=xxx.xxx.xxx.xxx 58670 22
TMPDIR=/scratch
LMOD_SYSTEM_DEFAULT_MODULES=StdEnv
LIBRARY_PATH=/blues/gpfs/home/software/spack-0.10.1/
→ opt/spack/linux-centos7-x86_64/intel-17.0.4/inte
→ 1-mkl-2017.3.196-v7uuj6zmthzln35n2hb7i5u5ybncv5e
→ v/compilers_and_libraries_2017.4.196/linux/tbb/1
→ ib/intel64_lin/gcc4.7:/blues/gpfs/home/software/
→ spack-0.10.1/opt/spack/linux-centos7-x86_64/inte
→ 1-17.0.4/intel-mkl-2017.3.196-v7uuj6zmthzln35n2h
→ b7i5u5ybncv5ev/compilers_and_libraries_2017.4.19
→ 6/linux/compiler/lib/intel64_lin:/blues/gpfs/hom
→ e/software/spack-0.10.1/opt/spack/linux-centos7-
→ x86_64/intel-17.0.4/intel-mkl-2017.3.196-v7uuj6z
→ mthzln35n2hb7i5u5ybncv5ev/compilers_and_librarie
→ s_2017.4.196/linux/mkl/lib/intel64_lin:/blues/gp
→ fs/home/software/spack-0.10.1/opt/spack/linux-ce
→ ntos7-x86_64/intel-17.0.4/intel-mkl-2017.3.196-v
→ 7uuj6zmthzln35n2hb7i5u5ybncv5ev/lib:/blues/gpfs/
→ home/software/spack-0.10.1/opt/spack/linux-cento
→ s7-x86_64/gcc-4.8.5/intel-17.0.4-74uvhjiulyqgvsm
→ ywifbbuo46v5n42xc/tbb/lib/intel64/gcc4.4:/blues/
→ gpfs/home/software/spack-0.10.1/opt/spack/linux-
→ centos7-x86_64/gcc-4.8.5/intel-17.0.4-74uvhjiuly
→ qgvsmwyifbbuo46v5n42xc/lib/intel64
```

```

LMOD_PKG=/blues/gpfs/home/software/spack-0.10.1/opt/
↳ spack/linux-centos7-x86_64/gcc-4.8.5/lmod-7.4.9-
↳ ic63herzfgw5u3na5mdtvp3nwxy6oj2z/lmod/lmod
LMOD_VERSION=7.4.9
SSH_TTY=/dev/pts/0
MIC_LD_LIBRARY_PATH=/blues/gpfs/home/software/spack-
↳ 0.10.1/opt/spack/linux-centos7-x86_64/intel-17.0-
↳ .4/intel-mkl-2017.3.196-v7uuj6zmthzln35n2hb7i5u5-
↳ ybncv5ev/compilers_and_libraries_2017.4.196/linux-
↳ x/mkl/lib/intel64_lin_mic:/blues/gpfs/home/softw-
↳ are/spack-0.10.1/opt/spack/linux-centos7-x86_64/
↳ intel-17.0.4/intel-mkl-2017.3.196-v7uuj6zmthzln3-
↳ 5n2hb7i5u5ybncv5ev/compilers_and_libraries_2017.-
↳ 4.196/linux/compiler/lib/intel64_lin_mic:/blues/
↳ gpfs/home/software/spack-0.10.1/opt/spack/linux-
↳ centos7-x86_64/intel-17.0.4/intel-mkl-2017.3.196-
↳ -v7uuj6zmthzln35n2hb7i5u5ybncv5ev/compilers_and_l-
↳ ibraries_2017.4.196/linux/tbb/lib/intel64_lin_mi-
↳ c:/blues/gpfs/home/software/spack-0.10.1/opt/spa-
↳ ck/linux-centos7-x86_64/gcc-4.8.5/intel-17.0.4-7-
↳ 4uvhjiulyqgvsmywifbbuo46v5n42xc/tbb/lib/mic:/blu-
↳ es/gpfs/home/software/spack-0.10.1/opt/spack/lin-
↳ ux-centos7-x86_64/gcc-4.8.5/intel-17.0.4-74uvhji-
↳ ulyqgvsmywifbbuo46v5n42xc/lib/mic
USER=USER

LS_COLORS=rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:p
↳ i=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38
↳ ;5;11:cd=48;5;232;38;5;3:or=48;5;232;38;5;9:mi=0
↳ 5;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;1
↳ 1;38;5;16:ca=48;5;196;38;5;226:tw=48;5;10;38;5;1
↳ 6:ow=48;5;10;38;5;21:st=48;5;21;38;5;15:ex=38;5;
↳ 34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9:*.arj=
↳ 38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*
↳ 1zh=38;5;9:*.lzma=38;5;9:*.tlz=38;5;9:*.txz=38;5
↳ ;9:*.tzo=38;5;9:*.tz=38;5;9:*.zip=38;5;9:*.z=38
↳ ;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38
↳ ;5;9:*.lz=38;5;9:*.lzo=38;5;9:*.xz=38;5;9:*.bz2=
↳ 38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*
↳ tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=38;5;9
↳ :*.war=38;5;9:*.ear=38;5;9:*.sar=38;5;9:*.rar=38
↳ ;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cp
↳ io=38;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*
↳ .jpg=38;5;13:*.jpeg=38;5;13:*.gif=38;5;13:*.bmp=
↳ 38;5;13:*.pbm=38;5;13:*.pgm=38;5;13:*.ppm=38;5;1
↳ 3:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.ti
↳ f=38;5;13:*.tiff=38;5;13:*.png=38;5;13:*.svg=38;
↳ 5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:
↳ *.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v
↳ =38;5;13:*.mkv=38;5;13:*.webm=38;5;13:*.ogm=38;5
↳ ;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*
↳ .vob=38;5;13:*.qt=38;5;13:*.nuv=38;5;13:*.wmv=38
↳ ;5;13:*.ASF=38;5;13:*.rm=38;5;13:*.rmvb=38;5;13:
↳ *.flc=38;5;13:*.avi=38;5;13:*.fli=38;5;13:*.flv=
↳ 38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:
↳ *.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38;5;13:*.emf=
↳ 38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;1
↳ 3:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.f1a
↳ c=38;5;45:*.mid=38;5;45:*.midi=38;5;45:*.mka=38;
↳ 5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*
↳ .ra=38;5;45:*.wav=38;5;45:*.axa=38;5;45:*.oga=38
↳ ;5;45:*.spx=38;5;45:*.xspf=38;5;45:

```

Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model

```

LD_LIBRARY_PATH=/blues/gpfs/home/software/spack-0.10]
↳ .1/opt/spack/linux-centos7-x86_64/intel-17.0.4/i]
↳ intel-mkl-2017.3.196-v7uuj6zmthzln35n2hb7i5u5ybncv
↳ cv5ev/compilers_and_libraries_2017.4.196/linux/tb]
↳ b/lib/intel64_lin/gcc4.7:/blues/gpfs/home/softwa
↳ re/spack-0.10.1/opt/spack/linux-centos7-x86_64/i]
↳ intel-17.0.4/intel-mkl-2017.3.196-v7uuj6zmthzln35
↳ n2hb7i5u5ybncv5ev/compilers_and_libraries_2017.4
↳ .196/linux/compiler/lib/intel64_lin:/blues/gpfs/
↳ home/software/spack-0.10.1/opt/spack/linux-cento
↳ s7-x86_64/intel-17.0.4/intel-mkl-2017.3.196-v7uu
↳ j6zmthzln35n2hb7i5u5ybncv5ev/compilers_and_libra
↳ ries_2017.4.196/linux/mkl/lib/intel64_lin:/blues
↳ /gpfs/home/software/spack-0.10.1/opt/spack/linux
↳ -centos7-x86_64/intel-17.0.4/intel-mkl-2017.3.196
↳ -v7uuj6zmthzln35n2hb7i5u5ybncv5ev/lib:/blues/gpfs
↳ /home/software/spack-0.10.1/opt/spack/linux-cent
↳ os7-x86_64/intel-17.0.4/intel-mpi-2017.3-dfphq6k
↳ avje2olnichisvjjndtridrok/compilers_and_librarie
↳ s_2017.4.196/linux/mpi/mic/lib:/blues/gpfs/home/
↳ software/spack-0.10.1/opt/spack/linux-centos7-x8
↳ 6_64/intel-17.0.4/intel-mpi-2017.3-dfphq6kavje2o
↳ lnichisvjjndtridrok/compilers_and_libraries_2017
↳ .4.196/linux/mpi/intel64/lib:/blues/gpfs/home/so
↳ ftware/spack-0.10.1/opt/spack/linux-centos7-x86_
↳ 64/gcc-4.8.5/intel-17.0.4-74uvhjiulyqgvsmywifbbu
↳ o46v5n42xc/tbb/lib/intel64/gcc4.4:/blues/gpfs/ho
↳ me/software/spack-0.10.1/opt/spack/linux-centos7
↳ -x86_64/gcc-4.8.5/intel-17.0.4-74uvhjiulyqgvsmywi
↳ fbbuo46v5n42xc/lib/intel64
LMOD_MPI_NAME=intel-mpi
MIC_LIBRARY_PATH=/blues/gpfs/home/software/spack-0.1
↳ 0.1/opt/spack/linux-centos7-x86_64/intel-17.0.4/
↳ intel-mkl-2017.3.196-v7uuj6zmthzln35n2hb7i5u5yb
↳ cv5ev/compilers_and_libraries_2017.4.196/linux/m
↳ k1/lib/intel64_lin_mic:/blues/gpfs/home/software
↳ /spack-0.10.1/opt/spack/linux-centos7-x86_64/int
↳ el-17.0.4/intel-mkl-2017.3.196-v7uuj6zmthzln35n2
↳ hb7i5u5ybncv5ev/compilers_and_libraries_2017.4.1
↳ 96/linux/compiler/lib/intel64_lin_mic:/blues/gpf
↳ s/home/software/spack-0.10.1/opt/spack/linux-cen
↳ tos7-x86_64/intel-17.0.4/intel-mkl-2017.3.196-v7
↳ uuoj6zmthzln35n2hb7i5u5ybncv5ev/compilers_and_li
↳ raries_2017.4.196/linux/tbb/lib/intel64_lin_mic:
↳ /blues/gpfs/home/software/spack-0.10.1/opt/spack_
↳ /linux-centos7-x86_64/gcc-4.8.5/intel-17.0.4-74u
↳ vhjiulyqgvsmywifbbuo46v5n42xc/tbb/lib/mic:/blues
↳ /gpfs/home/software/spack-0.10.1/opt/spack/linux
↳ -centos7-x86_64/gcc-4.8.5/intel-17.0.4-74uvhjiuly
↳ qgvsmywifbbuo46v5n42xc/lib/mic
CPATH=/blues/gpfs/home/software/spack-0.10.1/opt/spa
↳ ck/linux-centos7-x86_64/intel-17.0.4/intel-mkl-2
↳ 017.3.196-v7uuj6zmthzln35n2hb7i5u5ybncv5ev/compi
↳ lers_and_libraries_2017.4.196/linux/mkl/include:
↳ /blues/gpfs/home/software/spack-0.10.1/opt/spack_
↳ /linux-centos7-x86_64/gcc-4.8.5/intel-17.0.4-74u
↳ vhjiulyqgvsmywifbbuo46v5n42xc/tbb/include
(ModuleTable004=_L3NoYXJ1L3NwYWNrL2xtb2QvbGludXgtY2V
↳ udG9zNy140DZfNjQvaW50ZWwtbXBpLzIwMTcuMy1kZnBocTz
↳ rL2ludGVsLzE3LjAuNCIsIi9ibHVlcY9ncGZzL2hvbwUvc29
↳ mdHdhcmUvc3BhY2stMC4xMC4xL3NoYXJ1L3NwYWNrL2xtb2Q
↳ vbGludXgtY2VudG9zNy140DZfNjQvaW50ZWwtbXBpLzIwMTcu
↳ iL2JsdWVzL2dwZmMvaG9tZS9zb2Z0d2FyZS9zcGFjay0wLjE
↳ wLjEvc2hhcmUvc3BhY2svbG1vZC9saW51eC1jZW50b3M3LXg
↳ 4N182NC9Db3J1iwiL3NvZnQvYmVi3AvbW9kdWx1ZmlsZX
↳ iLCIvb3B0L2NyYXkvBw9kdWx1ZmlsZXMiLCIvb3B0L2NyYXK
↳ vY3JheXB1L2R1ZmF1bHQvbW9kdWx1ZmlsZXMiLCIvc29mdHd
↳ hcmUvY2VudG9zNy140DZfNjQvaW50ZWwtbXBpLzIwMTcuMy1k
LMOD_FAMILY_MPI_VERSION=2017.3-dfphq6k
NLSPATH=/blues/gpfs/home/software/spack-0.10.1/opt/s
↳ pack/linux-centos7-x86_64/intel-17.0.4/intel-mkl
↳ -2017.3.196-v7uuj6zmthzln35n2hb7i5u5ybncv5ev/comp
↳ ilers_and_libraries_2017.4.196/linux/mkl/lib/int
↳ el64_lin/locale/%l_%t/%N:/blues/gpfs/home/softwa
↳ re/spack-0.10.1/opt/spack/linux-centos7-x86_64/g
↳ cc-4.8.5/intel-17.0.4-74uvhjiulyqgvsmywifbbuo46v
↳ 5n42xc/compilers_and_libraries_2017.4.193/linux/
↳ compiler/lib/intel64/locale/%l_%t/%N
PATH=/blues/gpfs/home/software/spack-0.10.1/opt/spac
↳ k/linux-centos7-x86_64/intel-17.0.4/intel-mkl-20
↳ 17.3.196-v7uuj6zmthzln35n2hb7i5u5ybncv5ev/bin:/b
↳ lues/gpfs/home/software/spack-0.10.1/opt/spack/l
↳ inux-centos7-x86_64/intel-17.0.4/intel-mpi-2017.
↳ 3-dfphq6kavje2olnichisvjjndtridrok/compilers_and
↳ _libraries_2017.4.196/linux/mpi/intel64/bin:/blu
↳ es/gpfs/home/software/spack-0.10.1/opt/spack/lin
↳ ux-centos7-x86_64/gcc-4.8.5/intel-17.0.4-74uvhji
↳ ulyqgvsmywifbbuo46v5n42xc/bin:/usr/local/bin:/us
↳ r/bin:/usr/local/sbin:/usr/sbin:/soft/xxxx/xxxx/
↳ bin
MAIL=/var/spool/mail/USER
(ModuleTable001=_X01vZHVzZVRhYmx1Xz17WyJNVHZlcnNpb24
↳ iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbn1LFsiY19zaG9
↳ ydFRpbWUiXT1mYWxzSxkZXBoaFQ9e30sZmFtaWx5PXTbImN
↳ vbXpbGVyIl09ImludGVsIixbIm1waSJdPSJpbnR1bC1tcGk
↳ iLH0sbVQ9e1N0ZEVudj17WyJmbiJdPSIvYmx1ZXMyZ3Bmcy9
↳ ob211L3NvZnR3YXJ1L3NwYWNrLTauMS9zaGFyZS9zcGF
↳ jay9sbW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L0NvcmUvU3R
↳ kRW52Lmx1YSIsWyJmdWxsTmFtZSJdPSJTdGRFbnYiLfsibG9
↳ hZE9yZGVyIl09NCxwcm9wVD17fSxbInN0YXR1cyJdPSJhY3R
↳ pdmUiLfsidXN1ck5hbWUiXT0iU3RkRW52Iix9LGladGVsPxt
↳ bImZuIl09Ii9ibHVlcY9ncGZzL2hvbwUvc29mdHdcmUv
_=/usr/bin/env
LMOD_SETTARG_CMD=:
PWD=/home/USER/codes/Author-Kit

```

```

_LMFILES_=~/blues/gpfs/home/software/spack-0.10.1/sha_
↳ re/spack/lmod/linux-centos7-x86_64/Core/intel/17_
↳ .0.4-74uvhji.lua:/blues/gpfs/home/software/spack_
↳ -0.10.1/share/spack/lmod/linux-centos7-x86_64/int_
↳ el/17.0.4/intel-mpi/2017.3-dfphq6k.lua:/blues/gp_
↳ fs/home/software/spack-0.10.1/share/spack/lmod/l_
↳ inux-centos7-x86_64/intel/17.0.4/intel-mkl/2017.j_
↳ 3.196-v7uuj6z.lua:/blues/gpfs/home/software/spac_
↳ k-0.10.1/share/spack/lmod/linux-centos7-x86_64/C_
↳ ore/StdEnv.lua
MODULEPATH=/blues/gpfs/home/software/spack-0.10.1/sh_
↳ are/spack/lmod/linux-centos7-x86_64/intel-mpi/20_
↳ 17.3-dfphq6k/intel/17.0.4:/blues/gpfs/home/softw_
↳ are/spack-0.10.1/share/spack/lmod/linux-centos7-
↳ x86_64/intel/17.0.4:/blues/gpfs/home/software/sp_
↳ ack-0.10.1/share/spack/lmod/linux-centos7-x86_64_
↳ /Core:/soft/xxxx/modulefiles:/opt/cray/modulefil_
↳ es:/opt/cray/craype/default/modulefiles:/softwar_
↳ e/centos7/spack/share/spack/lmod/linux-centos7-x_
↳ 86_64/Core:/software/centos7/modulefiles
LOADEDMODULES=intel/17.0.4-74uvhji:intel-mpi/2017.3-
↳ dfphq6k:intel-mkl/2017.3.196-v7uuj6z:StdEnv
_ModuleTable_Sz_=5
LMOD_CMD=/blues/gpfs/home/software/spack-0.10.1/opt/
↳ spack/linux-centos7-x86_64/gcc-4.8.5/lmod-7.4.9-
↳ ic63herzfgw5u3na5mdtvp3nwxy6oj2z/lmod/lmod/libex_
↳ ec/lmod
_ModuleTable005_=L2xpbnV4LWN1bnRvczcteDg2XzY0L0NvcmU_
↳ iLCIvc29mdHdhcmUvY2VudG9zNy9tb2R1bGVmaWxlcyIsfSx_
↳ bInN5c3R1bUJhc2VNUEFUSCJdPSIvYmx1ZXMvZ3Bmcy9ob21_
↳ lL3NvZnR3YXJ1l3NwYWNrLTauMTAuMS9zaGFyZs9zCGFjay9_
↳ sbW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L0NvcmU6L3NvZnQ_
↳ vYmVib3AvbW9kdWx1ZmlsZXM6L29wdC9jcmF5L21vZHVsZWZ_
↳ pbGVzOi9vcHQvY3JheS9jcmF5cGUvZGvmyXVsdC9tb2R1bGV_
↳ maWxlczovc29mdHdhcmUvY2VudG9zNy9zcGFjay9zaGFyZS9_
↳ zcGFjay9sbW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L0NvcmU_
↳ 6L3NvZnR3YXJ1l2N1bnRvczcvbW9kdWx1ZmlsZXMiLH0=
HISTCONTROL=ignoredups
HOME=/home/USER
SHLVL=2
_ModuleTable002_=c3BhY2stMC4xMC4xL3NoYXJ1l3NwYWNrL2x_
↳ tb2QvbGludXgtY2VudG9zNy140DZfNjQvQ29yZS9pbnR1bC8_
↳ xNy4wLjQtNzR1dmhqaS5sdWEiLFsiZnVsbE5hbWUiXT0iaW5_
↳ 0ZwvMTcuMC40LTc0dXzoamkiLFsibG9hZE9yZGVyIl09MSx_
↳ wcm9wVD17fSxbInN0YXR1cyJdPSJhY3RpdmUiLFsidXN1ck5_
↳ hbWUiXT0iaW50ZWwvMTcuMC40LTc0dXzoamkiLH0sWyJpbnR_
↳ 1bC1ta2wiXT17WjYmbiJdPSIvYmx1ZXMvZ3Bmcy9ob211L3N_
↳ vZnR3YXJ1l3NwYWNrLTauMTAuMS9zaGFyZs9zcgFjay9sbW9_
↳ kL2xpbnV4LWN1bnRvczcteDg2XzY0L21udGVsLzE3LjAuNC9_
↳ pbnR1bC1ta2wvMjAxNy4zLjE5Ni12N3V1ajZ6Lmx1YSIsWj_
↳ mdWxsTmFtZSJdPSJpbnR1bC1ta2wvMjAxNy4zLjE5Ni12
BASH_ENV=/blues/gpfs/home/software/spack-0.10.1/opt/
↳ spack/linux-centos7-x86_64/gcc-4.8.5/lmod-7.4.9-
↳ ic63herzfgw5u3na5mdtvp3nwxy6oj2z/lmod/lmod/init/
↳ bash
LOGNAME=USER

SSH_CONNECTION=xxx.xxx.xxx.xxx 58670 xxx.xxx.xxx.xxx
↳ 22
CLASSPATH=/blues/gpfs/home/software/spack-0.10.1/opt_
↳ /spack/linux-centos7-x86_64/intel-17.0.4/intel-m_
↳ pi-2017.3-dfphq6kavje2olnichisvjnjndtridrok/compi_
↳ lers_and_libraries_2017.4.196/linux/mpi/intel64/
↳ lib/mpi.jar
MODULESHOME=/blues/gpfs/home/software/spack-0.10.1/o_
↳ pt/spack/linux-centos7-x86_64/gcc-4.8.5/lmod-7.4_
↳ .9-ic63herzfgw5u3na5mdtvp3nwxy6oj2z/lmod/lmod
LESSOPEN=||/usr/bin/lesspipe.sh %s
__Init_Default_Modules=1
LMOD_MPI_VERSION=2017.3-dfphq6k
LMOD_FULL_SETTARG_SUPPORT=no
LMOD_FAMILY_COMPILER=intel
CMAKE_PREFIX_PATH=/blues/gpfs/home/software/spack-0.1_
↳ 10.1/opt/spack/linux-centos7-x86_64/intel-17.0.4_
↳ /intel-mkl-2017.3.196-v7uuj6zmthzln35n2hb7i5u5yb_
↳ ncv5ev:/blues/gpfs/home/software/spack-0.10.1/op_
↳ t/spack/linux-centos7-x86_64/gcc-4.8.5/intel-17.0.4_
↳ 0.4-74uvhjiulyqgvsmwyifbbuo46v5n42xc
LMOD_DIR=/blues/gpfs/home/software/spack-0.10.1/opt/
↳ spack/linux-centos7-x86_64/gcc-4.8.5/lmod-7.4.9-
↳ ic63herzfgw5u3na5mdtvp3nwxy6oj2z/lmod/lmod/libex_
↳ ec
LMOD_FAMILY_MPI=intel-mpi
I_MPI_ROOT=/blues/gpfs/home/software/spack-0.10.1/op_
↳ t/spack/linux-centos7-x86_64/intel-17.0.4/intel-m_
↳ pi-2017.3-dfphq6kavje2olnichisvjnjndtridrok/comp_
↳ ilers_and_libraries_2017.4.196/linux/mpi
BASH_FUNC_module()={ eval $($LMOD_CMD bash "$@")_
↳ && eval ${${LMOD_SETTARG_CMD}:-} -s sh)
}
BASH_FUNC_m1()={ eval $($LMOD_DIR/m1_cmd "$@")_
}
+ lsb_release -a
collect_environment.sh: line 10: lsb_release: command
↳ not found
+ uname -a
Linux bdw-0159 3.10.0-957.5.1.el7.x86_64 #1 SMP Fri
↳ Feb 1 14:54:57 UTC 2019 x86_64 x86_64 x86_64
↳ GNU/Linux
+ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 36
On-line CPU(s) list: 0-35
Thread(s) per core: 1
Core(s) per socket: 18
Socket(s): 2
NUMA node(s): 2
Vendor ID: GenuineIntel
CPU family: 6
Model: 79

```

Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model

```

Model name: Intel(R) Xeon(R) CPU E5-2695 v4
          ↳ @ 2.10GHz
Stepping: 1
CPU MHz: 2407.617
CPU max MHz: 3300.0000
CPU min MHz: 1200.0000
BogoMIPS: 4153.34
Virtualization: VT-x
L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 46080K
NUMA node0 CPU(s): 0-17
NUMA node1 CPU(s): 18-35
Flags: fpu vme de pse tsc msr pae mce
      ↳ cx8 apic sep mtrr pge mca cmov pat pse36 clflush
      ↳ dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
      ↳ pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
      ↳ bts rep_good nopl xtopology nonstop_tsc
      ↳ aperfmpf eagerfpu pni pclmulqdq dtes64 monitor
      ↳ ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr
      ↳ pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt
      ↳ tsc_deadline_timer aes xsave avx f16c rdrand
      ↳ lahf_lm abm 3dnowprefetch epb cat_13 cdp_13
      ↳ intel_ppin intel_pt ssbd ibrs ibpb stibp
      ↳ tpr_shadow vnmi flexpriority ept vpid fsgsbase
      ↳ tsc_adjust bmi1 hle avx2 smp bmi2 erms invpcid
      ↳ rtm cqmq_rdt_a rdseed adx smap xsaveopt cqmq_llc
      ↳ cqmq_occup_llc cqmq_mbmbm_total cqmq_mbmbm_local dtherm
      ↳ ida arat pln pts spec_ctrl intel_stibp flush_l1d
+ cat /proc/meminfo
MemTotal: 131911416 kB
MemFree: 125514056 kB
MemAvailable: 125112424 kB
Buffers: 0 kB
Cached: 3356524 kB
SwapCached: 0 kB
Active: 501512 kB
Inactive: 167684 kB
Active(anon): 472016 kB
Inactive(anon): 130848 kB
Active(file): 29496 kB
Inactive(file): 36836 kB
Unevictable: 4207912 kB
Mlocked: 1048576 kB
SwapTotal: 0 kB
SwapFree: 0 kB
Dirty: 0 kB
Writeback: 0 kB
AnonPages: 1520700 kB
Mapped: 141116 kB
Shmem: 130856 kB
Slab: 387844 kB
SReclaimable: 202376 kB
SUnreclaim: 185468 kB
KernelStack: 22368 kB
PageTables: 10264 kB
NFS_Unstable: 0 kB
Bounce: 0 kB
WritebackTmp: 0 kB
CommitLimit: 65955708 kB
Committed_AS: 1880648 kB
VmallocTotal: 34359738367 kB
VmallocUsed: 710948 kB
VmallocChunk: 34291711180 kB
HardwareCorrupted: 0 kB
AnonHugePages: 1161216 kB
CmaTotal: 0 kB
CmaFree: 0 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
DirectMap4k: 357752 kB
DirectMap2M: 23662592 kB
DirectMap1G: 112197632 kB
+ inxi -F -c0
collect_environment.sh: line 14: inxi: command not
      ↳ found
+ lsblk -a
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
loop0 7:0 0 0 1 loop
loop1 7:1 0 0 1 loop
loop2 7:2 0 0 1 loop
loop3 7:3 0 0 1 loop
loop4 7:4 0 0 1 loop
loop5 7:5 0 0 1 loop
loop6 7:6 0 0 1 loop
loop7 7:7 0 0 1 loop
loop8 7:8 0 0 1 loop
loop9 7:9 0 0 1 loop
loop10 7:10 0 0 1 loop
loop11 7:11 0 0 1 loop
loop12 7:12 0 0 1 loop
loop13 7:13 0 0 1 loop
loop14 7:14 0 0 1 loop
loop15 7:15 0 0 1 loop
loop16 7:16 0 0 1 loop
loop17 7:17 0 0 1 loop
loop18 7:18 0 0 1 loop
loop19 7:19 0 0 1 loop
loop20 7:20 0 0 1 loop
loop21 7:21 0 0 1 loop
loop22 7:22 0 0 1 loop
loop23 7:23 0 0 1 loop
loop24 7:24 0 0 1 loop
loop25 7:25 0 0 1 loop
loop26 7:26 0 0 1 loop
loop27 7:27 0 0 1 loop
loop28 7:28 0 0 1 loop
loop29 7:29 0 0 1 loop

```

```

loop30 7:30 0      1 loop
loop31 7:31 0      1 loop
+ lsscsi -s
collect_environment.sh: line 16: lsscsi: command not
→  found
+ module list
++ /blues/gpfs/home/software/spack-0.10.1/opt/spack/
→  linux-centos7-x86_64/gcc-4.8.5/lmod-7.4.9-ic63he
→  rfgw5u3na5mdtvp3nwxy6oj2z/lmod/lmod/libexec/lmo
→  d bash
→  list

```

Currently Loaded Modules:

```

1) intel/17.0.4-74uvhji 2)
→  intel-mpi/2017.3-dfphq6k 3)
→  intel-mkl/2017.3.196-v7uuj6z 4) StdEnv

```

```

+ eval
→  'MODULEPATH="/blues/gpfs/home/software/spack-0.1
→  .1/share/spack/lmod/linux-centos7-x86_64/intel-
→  mpi/2017.3-dfphq6k/intel/17.0.4:/blues/gpfs/home
→  /software/spack-0.10.1/share/spack/lmod/linux-ce
→  ntos7-x86_64/intel/17.0.4:/blues/gpfs/home/softw
→  are/spack-0.10.1/share/spack/lmod/linux-centos7-
→  x86_64/Core:/soft/xxxx/modulefiles:/opt/cray/mod
→  ulefiles:/opt/cray/craype/default/modulefiles:/s
→  oftware/centos7/spack/share/spack/lmod/linux-cen
→  tos7-x86_64/Core:/software/centos7/modulefiles";'
export 'MODULEPATH; ' '_ModuleTable001_="X01vZHVsZVRh
→  Ymx1Xz17WjJNvHZlcnNpb24iXT0zLFsiY19yZWJ1aWxkVGlt
→  ZSJdPWZhbnHN1LFsiY19zaG9ydFRpbWUiXT1mYWxzSxkZXB0
→  aFQ9e30sZmFtaWx5PXtbImNvbXBpbGvyIl09ImludGVsIixb
→  Im1waSJdPSJpbnR1bC1tcGkiLH0sbVQ9e1N0ZEVuji7WjyJm
→  biJdPSIVymx1ZXMvZ3Bmcy9ob211L3NvZnR3YXJ1L3NwYWNr
→  LTauMTAuMS9zaGFyZS9zcGFjay9sbW9kL2xpbnV4LWN1bnRv
→  czcteDg2XzY0L0NvcmlUvU3RkRW52Lmx1YSIsWyJmdWxsTmFt
→  ZSJdPSJTdGRFbnYiLFsibG9hZE9yZGvyIl09NCxwcm9wVD17
→  fSxbInN0YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0i
→  U3RkRW52Iix9LGladGVsPxtbImZuIl09Ii9ibHV1cy9ncGzz
→  L2hvbWUvc29mdHdcmUv";
export '_ModuleTable001_;"'_ModuleTable002_="c3BhY2
→  stMC4xMC4xL3NoYXJ1L3NwYWNrl2xtb2QvbGludXgtY2VudG
→  9zNy140DZfNjQvQ29yZS9pbnR1bC8xNy4wLjQtNzR1dmhqaS
→  5sdWEiLFsiZnVsbE5hbWUiXT0iaW50ZWwvMTcuMC40LTc0dX
→  ZoamkiLFsibG9hZE9yZGvyIl09MSxwcm9wVD17fSxbInN0YX
→  R1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0iaW50ZWwvMT
→  cuMC40LTc0dXZoamkiLH0sWyJpbnR1bC1ta2wiXT17WjyJmbi
→  JdPSIVymx1ZXMvZ3Bmcy9ob211L3NvZnR3YXJ1L3NwYWNrLT
→  AuMTAuMS9zaGFyZS9zcGFjay9sbW9kL2xpbnV4LWN1bnRvcz
→  cteDg2XzY0L21udGVsLzE3LjAuNC9pbnR1bC1ta2wvMjAxNy
→  4zLjE5Ni12N3V1ajZ6Lmx1YSIsWyJmdWxsTmFtZSJdPSJpbn
→  R1bC1ta2wvMjAxNy4zLjE5Ni12";'

```

```

export '_ModuleTable002_;"'_ModuleTable003_="N3V1aj
→  Z6IixbImxvYWRPcmRlcjJdPTMsCHJvcFQ9e30sWyJzdGF0dX
→  MiXT0iYWN0aXZlIixbInVzZXJOYW1lI09ImludGVsLW1rbC
→  IsfSxbImludGVsLW1waSJdPxtbImZuIl09Ii9ibHV1cy9ncG
→  ZzL2hvbWUvc29mdHdcmUvc3BhY2stMC4xMC4xL3NoYXJ1L3
→  NwYWNrl2xtb2QvbGludXgtY2VudG9zNy140DZfNjQvaW50ZW
→  wvMTcuMC40L21udGVsLW1waS8yMDE3LjMtZGZwaHE2ay5sdW
→  EiLFSiZnVsbE5hbWUiXT0iaW50ZWwbtXBpLzIwMTcuMy1kZn
→  BocTzrIixbImxvYWRPcmRlcjJdPTIschJvcFQ9e30sWyJzdG
→  F0dXMiXT0iYWN0aXZlIixbInVzZXJOYW1lI09ImludGVsLW
→  1waSIsfSx9LG1wYXRoQT17Ii9ibHV1cy9ncGzzL2hvbWUvc2
→  9mdHdcmUvc3BhY2stMC4xMC4x";'
export '_ModuleTable003_;"'_ModuleTable004_="L3NoYXJ1L3NwYWNrl2xtb2QvbGludXgt
→  Y2VudG9zNy140DZfNjQvaW50ZWwbtXBpLzIwMTcuMy1kZnB
→  ocTzrL21udGVsLzE3LjAuNCIsi9ibHV1cy9ncGzzL2hvbW
→  Uvc29mdHdcmUvc3BhY2stMC4xMC4xL3NoYXJ1L3NwYWNrl
→  2xtb2QvbGludXgtY2VudG9zNy140DZfNjQvaW50ZWwvMTcu
→  MC40Iiwl2JsdWvZL2wdZnMvaG9tZs9zb2Z0d2FyZs9zcGf
→  jay0WljEwljEvc2hdcUvc3BhY2svbG1vZC9saW51eC1jZw
→  50b3M3LXg4N182NC9db3J1IiwiL3NvZnQyVmVib3AvbW9kd
→  Wx1Zm1sZXMlLCIvb3B0L2NyYXkvB9kdWx1Zm1sZXMlLCIv
→  b3B0L2NyYXkvY3JheXB1L2R1ZmF1bHQvbW9kdWx1Zm1sZXM
→  iLCIvc29mdHdcmUvY2VudG9zNy9zcGFjay9zaGFyZs9zcG
→  Fjaj9sbW9k";'
export '_ModuleTable004_;"'_ModuleTable005_="L2xpbn
→  V4LWN1bnRvczcteDg2XzY0L0NvcmlUilcIvc29mdHdcmUvY2
→  VudG9zNy9tb2R1bGvmaWx1cyIsfSxbInN5c3R1bUJhc2VNUe
→  FUSCJdPSIVymx1ZXMvZ3Bmcy9ob211L3NvZnR3YXJ1L3NwYw
→  NrLTauMTauMS9zaGFyZs9zcGFjay9sbW9kL2xpbnV4LWN1bn
→  RvczcteDg2XzY0L0NvcmlU6L3NvZnQyVmVib3AvbW9kdWx1Zm
→  1sZXM6L29wdC9jcmF5L21vZHVszWZpbGVzO19vcQyV3JheS
→  9jcmF5cGUvZGvmyXVvsdC9tb2R1bGvmaWx1czovc29mdHdcm
→  UvY2VudG9zNy9zcGFjay9zaGFyZs9zcGFjay9sbW9kL2xpbn
→  V4LWN1bnRvczcteDg2XzY0L0NvcmlU6L3NvZnR3YXJ1L2N1bn
→  RvczcvbW9kdWx1Zm1sZXMlLH0=";
export '_ModuleTable005_;"'_ModuleTable_Sz_="5";'
export '_ModuleTable_Sz_;"'_ModuleTable_Sz_="5";'
++ MODULEPATH=/blues/gpfs/home/software/spack-0.10.1
→  /share/spack/lmod/linux-centos7-x86_64/intel-mpi
→  /2017.3-dfphq6k/intel/17.0.4:/blues/gpfs/home/so
→  ftware/spack-0.10.1/share/spack/lmod/linux-cento
→  s7-x86_64/intel/17.0.4:/blues/gpfs/home/software
→  /spack-0.10.1/share/spack/lmod/linux-centos7-x86
→  _64/Core:/soft/xxxx/modulefiles:/opt/cray/module
→  files:/opt/cray/craype/default/modulefiles:/soft
→  ware/centos7/spack/share/spack/lmod/linux-centos
→  7-x86_64/Core:/software/centos7/modulefiles
++ export MODULEPATH

```

Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model

```

++ _ModuleTable001_=X01vZHVzZVRhYmx1Xz17WyJNVHZlcnNp_
→ b24iXT0zLFsiY19yZWJ1aWxkVG1tZSJdPWZhbn1LfsiY19z_
→ aG9ydFRpbWUiXT1mYWxzSxkZXBoaFQ9e30sZmFtaWx5Pxtb_
→ ImNvbXBpGvyIl09ImludGVsIixbIm1waSJdpSjpbnR1bC1t_
→ cGkiLH0sbVQ9e1N0ZEVuj17WyJmbiJdPSIVYmx1ZXMvZ3Bm_
→ cy9ob21L3NvZnR3YXJ1L3NwYWNRlTAuMTAuMS9zaGFyZS9z_
→ cGFjay9sbW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L0NvcnUv_
→ U3RkRW52Lmx1YStsWYJmdWxsTmftZSJdPSJdTGRfbnYiLfsi_
→ bG9hZE9yZGvyIl09NCxwcm9wVD17FsxbInN0YXR1cyJdPSJh_
→ Y3RpdmUiLfsidXN1ck5hbWUiXT01U3RkRW52Iix9LGludGVs_
→ PXtbImZuIl09Ii9ibHVlc9ncGZzL2hvbWUvc29mdHdcmUv
++ export _ModuleTable001_
++ _ModuleTable002_=c3BhY2stMC4xMC4xL3NoYXJ1L3NwYWNR_
→ L2xtb2QvbGludXgtY2VudG9zNy140DZfNjQvQ29yZS9pbnR1_
→ bC8xNy4wLjQtNzR1dmhqaS5sdWEiLfsiZnVsbE5hbWUiXT0i_
→ aW50ZWwvMTcuMC40LTC0dXZoamkiLfsiC9hZE9yZGvyIl09_
→ MSxwcm9wVD17FsxbInN0YXR1cyJdPSJhY3RpdmUiLfsidXN1_
→ ck5hbWUiXT0iaW50ZWwvMTcuMC40LTC0dXZoamkiLH0sWYJp_
→ bnR1bC1ta2wiXT17WyJmbiJdPSIVYmx1ZXMvZ3Bmc9ob21L_
→ L3NvZnR3YXJ1L3NwYWNRlTAuMTAuMS9zaGFyZS9zcGFjay9s_
→ bW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L21udGVsLzE3LjAu_
→ NC9pbnR1bC1ta2wvMjAxNy4zLjE5Ni12N3V1ajZ6Lmx1YSts_
→ WyJmdWxsTmftZSJdPSJpbnR1bC1ta2wvMjAxNy4zLjE5Ni12
++ export _ModuleTable002_
++ _ModuleTable003_=N3V1ajZ6IixbImxvYWRPcmRlcJdPTMs_
→ cHJvcFQ9e30sWYJzdGF0dXMiXT0iYWN0aXZlIixbInVzXJ0_
→ YW11Ii109ImludGVsLW1rbCIsfSxbImldGVsLW1waSJdPxtb_
→ ImZuIl09Ii9ibHVlc9ncGZzL2hvbWUvc29mdHdcmUvc3Bh_
→ Y2stMC4xMC4xL3NoYXJ1L3NwYWNRl2xtb2QvbGludXgtY2Vu_
→ dG9zNy140DZfNjQvaW50ZWwvMTcuMC40L21udGVsLw1waS8y_
→ MDE3LjMtZGZwaHE2ay5sdWEiLfsiZnVsbE5hbWUiXT0iaW50_
→ ZWwtbXBpLzIwMTcuMy1kZnBocTzrIixbImxvYWRPcmRlcJd_
→ PTIscHJvcFQ9e30sWYJzdGF0dXMiXT0iYWN0aXZlIixbInVz_
→ ZXJOYw11Ii109ImludGVsLw1waSIsfSx9LG1wYXR0QT17Ii9i_
→ bHVlc9ncGZzL2hvbWUvc29mdHdcmUvc3BhY2stMC4xMC4x
++ export _ModuleTable003_
++ _ModuleTable004_=L3NoYXJ1L3NwYWNRl2xtb2QvbGludXgt_
→ Y2VudG9zNy140DZfNjQvaW50ZWwtbXBpLzIwMTcuMy1kZnBo_
→ cTzrL21udGVsLzE3LjAuNCIsIi9ibHVlc9ncGZzL2hvbWUv_
→ c29mdHdcmUvc3BhY2stMC4xMC4xL3NoYXJ1L3NwYWNRl2xt_
→ b2QvbGludXgtY2VudG9zNy140DZfNjQvaW50ZWwvMTcuMC40_
→ IiwiL2JsdWVzL2dwZnMvaG9tZs9zb2Z0d2FyZs9zcGFjay0w_
→ LjEwLjEvc2hhcmUvc3BhY2svbG1vZC9saW51eC1jZW50b3M3_
→ LXg4N182NC9Db3J1IiwiL3NvZnQvYmVib3AvbW9kdWx1Zmls_
→ ZXMiLCIvb3B0L2NyYXkvBw9kdWx1ZmlsZXMiLCIvb3B0L2Ny_
→ YXkvY3JheXB1L2R1ZmF1bHQvbW9kdWx1ZmlsZXMiLCIvc29m_
→ dHdcmUvY2VudG9zNy9zcGFjay9zaGFyZS9zcGFjay9sbW9k
++ export _ModuleTable004_
++ _ModuleTable005_=L2xpbnV4LWN1bnRvczcteDg2XzY0L0Nv_
→ cmUiLCIvc29mdHdcmUvY2VudG9zNy9tb2R1bGvmaWx1cyIs_
→ fSxbInN5c3R1bUJhc2VNUEFUSCJdPSIVYmx1ZXMvZ3Bmc9y_
→ b21L3NvZnR3YXJ1L3NwYWNRlTAuMTAuMS9zaGFyZS9zcGFj_
→ ay9sbW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L0NvcmU6L3Nv_
→ ZnQvYmVib3AvbW9kdWx1ZmlsZXM6L29wdC9jcmF5L21vZHVs_
→ ZWZpbGvZoI9vcHQyV3JheS9jcmF5cGUvZGvmyXVsdC9tb2R1_
→ bGvmaWx1cyovc29mdHdcmUvY2VudG9zNy9zcGFjay9zaGFy_
→ ZS9zcGFjay9sbW9kL2xpbnV4LWN1bnRvczcteDg2XzY0L0Nv_
→ cmU6L3NvZnR3YXJ1L2N1bnRvczcvbW9kdWx1ZmlsZXM1H0=
++ export _ModuleTable005_
++ _ModuleTable_Sz_=5
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ nvidia-smi
collect_environment.sh: line 18: nvidia-smi: command
→ not found
+ cat
+ lshw -short -quiet -sanitize
WARNING: you should run this program as super-user.
H/W path          Device  Class      Description
=====
                                         system    Computer
/0                         bus      Motherboard
/0/0                        memory   128GiB System
→   memory
/0/1                      processor Intel(R)
→   Xeon(R) CPU E5-2695 v4 @ 2.10GHz
/0/2                      processor Intel(R)
→   Xeon(R) CPU E5-2695 v4 @ 2.10GHz
/0/100                     bridge    Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D DMI2
/0/100/1                   bridge    Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D PCI Express Root Port 1
/0/100/2                   bridge    Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D PCI Express Root Port 2
/0/100/2/0                 ib0      network   Omni-Path HFI
→   Silicon 100 Series [discrete]
/0/100/3                   bridge    Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D PCI Express Root Port 3
/0/100/4                   generic   Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
→   0
/0/100/4.1                 generic   Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
→   1
/0/100/4.2                 generic   Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
→   2
/0/100/4.3                 generic   Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
→   3
/0/100/4.4                 generic   Xeon E7 v4/Xeon
→   E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
→   4

```

```

/0/100/4.5          generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
  ↳ 5
/0/100/4.6          generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
  ↳ 6
/0/100/4.7          generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel
  ↳ 7
/0/100/5            generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Map/VTd_Misc/System
  ↳ Management
/0/100/5.1         generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D IIO Hot Plug
/0/100/5.2         generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D IIO RAS/Control
  ↳ Status/Global Errors
/0/100/5.4         generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D I/O APIC
/0/100/11           generic      C610/X99 series
  ↳ chipset SPSR
/0/100/11.4        storage      C610/X99
  ↳ series chipset sSATA Controller [AHCI mode]
/0/100/16           communication C610/X99
  ↳ series chipset MEI Controller #1
/0/100/16.1         communication C610/X99
  ↳ series chipset MEI Controller #2
/0/100/1a           bus         C610/X99 series
  ↳ chipset USB Enhanced Host Controller #2
/0/100/1c           bridge       C610/X99 series
  ↳ chipset PCI Express Root Port #1
/0/100/1c.3          bridge       C610/X99
  ↳ series chipset PCI Express Root Port #4
/0/100/1c.3/0        display      MGA G200e
  ↳ [Pilot] ServerEngines (SEP1)
/0/100/1c.4          bridge       C610/X99
  ↳ series chipset PCI Express Root Port #5
/0/100/1c.4/0        eno1        network     I350 Gigabit
  ↳ Network Connection
/0/100/1c.4/0.1      eno2        network     I350 Gigabit
  ↳ Network Connection
/0/100/1d           bus         C610/X99 series
  ↳ chipset USB Enhanced Host Controller #1
/0/100/1f           bridge       C610/X99 series
  ↳ chipset LPC Controller
/0/100/1f.2          storage      C610/X99
  ↳ series chipset 6-Port SATA Controller [AHCI mode]
/0/100/1f.3          bus         C610/X99 series
  ↳ chipset SMBus Controller
/0/3                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 0
/0/6                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 0
/0/7                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 0
/0/8                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 1
/0/9                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 1
/0/a                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 1
/0/b                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link 0/1
/0/c                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link 0/1
/0/d                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link 0/1
/0/e                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link Debug
/0/f                generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/10               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/11               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/12               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/13               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/14               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/15               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/16               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/17               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/18               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/19               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/1a               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/1b               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/1c               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/1d               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/1e               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/1f               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/20               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/21               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent
/0/22               generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent

```

Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model

/0/23	generic	Xeon E7 v4/Xeon	/0/3a	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -		
/0/24	generic	Xeon E7 v4/Xeon	↳ Channel 1 Error		
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			/0/3b	generic	Xeon E7 v4/Xeon
/0/25	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			↳ Interface		
/0/26	generic	Xeon E7 v4/Xeon	/0/3c	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
/0/27	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			/0/3d	generic	Xeon E7 v4/Xeon
/0/28	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
↳ E5 v4/Xeon E3 v4/Xeon D R2PCIE Agent			↳ Interface		
/0/29	generic	Xeon E7 v4/Xeon	/0/3e	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D R2PCIE Agent			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
/0/2a	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D Ubox			/0/3f	generic	Xeon E7 v4/Xeon
/0/2b	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Target Address/Thermal/RAS		
↳ E5 v4/Xeon E3 v4/Xeon D Ubox			/0/40	generic	Xeon E7 v4/Xeon
/0/2c	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Target Address/Thermal/RAS		
↳ E5 v4/Xeon E3 v4/Xeon D Ubox			/0/41	generic	Xeon E7 v4/Xeon
/0/2d	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Channel Target Address		
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 0			↳ Decoder		
/0/2e	generic	Xeon E7 v4/Xeon	/0/42	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 0			↳ E5 v4/Xeon E3 v4/Xeon D Channel Target Address		
/0/2f	generic	Xeon E7 v4/Xeon	↳ Decoder		
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 1			/0/43	generic	Xeon E7 v4/Xeon
/0/30	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 1			↳ Broadcast		
/0/31	generic	Xeon E7 v4/Xeon	/0/44	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Global Broadcast		
↳ Target Address/Thermal/RAS			/0/45	generic	Xeon E7 v4/Xeon
/0/32	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ Channel 0 Thermal Control		
↳ Target Address/Thermal/RAS			/0/46	generic	Xeon E7 v4/Xeon
/0/33	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ Channel 1 Thermal Control		
↳ Channel Target Address Decoder			/0/47	generic	Xeon E7 v4/Xeon
/0/34	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ Channel 0 Error		
↳ Channel Target Address Decoder			/0/48	generic	Xeon E7 v4/Xeon
/0/35	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1			↳ Channel 1 Error		
↳ Broadcast			/0/49	generic	Xeon E7 v4/Xeon
/0/36	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Global Broadcast			↳ Interface		
/0/37	generic	Xeon E7 v4/Xeon	/0/4a	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
↳ Channel 0 Thermal Control			↳ Interface		
/0/38	generic	Xeon E7 v4/Xeon	/0/4b	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
↳ Channel 1 Thermal Control			↳ Interface		
/0/39	generic	Xeon E7 v4/Xeon	/0/4c	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
↳ Channel 0 Error			↳ Interface		
			/0/4d	generic	Xeon E7 v4/Xeon
			↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		

/0/4e	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/4f	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/50	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/51	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/52	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/53	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/4	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 0		
/0/4.1	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 1		
/0/4.2	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 2		
/0/4.3	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 3		
/0/4.4	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 4		
/0/4.5	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 5		
/0/4.6	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 6		
/0/4.7	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Crystal Beach DMA Channel		
↳ 7		
/0/5	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Map/VTd_Misc/System		
↳ Management		
/0/5.1	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D IIO Hot Plug		
/0/5.2	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D IIO RAS/Control		
↳ Status/Global Errors		
/0/5.4	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D I/O APIC		
/0/54	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 0		
/0/55	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 0		
/0/56	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 0		
/0/57	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 1		
/0/58	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 1		
/0/59	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D QPI Link 1		
/0/5a	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link 0/1		
/0/5b	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link 0/1		
/0/5c	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link 0/1		
/0/5d	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D R3 QPI Link Debug		
/0/5e	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/5f	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/60	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/61	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/62	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/63	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/64	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/65	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/66	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/67	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/68	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/69	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/6a	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/6b	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/6c	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/6d	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/6e	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/6f	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/70	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/71	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		
/0/72	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent		

Significantly Improving Lossy Compression Quality Based on an Optimized Hybrid Prediction Model

/0/73	generic	Xeon E7 v4/Xeon	/0/8a	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
/0/74	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			/0/8b	generic	Xeon E7 v4/Xeon
/0/75	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			↳ Interface		
/0/76	generic	Xeon E7 v4/Xeon	/0/8c	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Caching Agent			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
/0/77	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D R2PCIe Agent			/0/8d	generic	Xeon E7 v4/Xeon
/0/78	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1		
↳ E5 v4/Xeon E3 v4/Xeon D R2PCIe Agent			↳ Interface		
/0/79	generic	Xeon E7 v4/Xeon	/0/8e	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Ubox			↳ E5 v4/Xeon E3 v4/Xeon D Target Address/Thermal/RAS		
/0/7a	generic	Xeon E7 v4/Xeon	/0/8f	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Ubox			↳ E5 v4/Xeon E3 v4/Xeon D Target Address/Thermal/RAS		
/0/7b	generic	Xeon E7 v4/Xeon	/0/90	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Ubox			↳ E5 v4/Xeon E3 v4/Xeon D Channel Target Address		
/0/7c	generic	Xeon E7 v4/Xeon	↳ Decoder		
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 0			/0/91	generic	Xeon E7 v4/Xeon
/0/7d	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D Channel Target Address		
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 0			↳ Decoder		
/0/7e	generic	Xeon E7 v4/Xeon	/0/92	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 1			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
/0/7f	generic	Xeon E7 v4/Xeon	↳ Broadcast		
↳ E5 v4/Xeon E3 v4/Xeon D Home Agent 1			/0/93	generic	Xeon E7 v4/Xeon
/0/80	generic	Xeon E7 v4/Xeon	↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Global Broadcast		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/94	generic	Xeon E7 v4/Xeon
↳ Target Address/Thermal/RAS			↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
/0/81	generic	Xeon E7 v4/Xeon	↳ Channel 0 Thermal Control		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/95	generic	Xeon E7 v4/Xeon
↳ Target Address/Thermal/RAS			↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
/0/82	generic	Xeon E7 v4/Xeon	↳ Channel 1 Thermal Control		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/96	generic	Xeon E7 v4/Xeon
↳ Channel Target Address Decoder			↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
/0/83	generic	Xeon E7 v4/Xeon	↳ Channel 0 Error		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/97	generic	Xeon E7 v4/Xeon
↳ Channel Target Address Decoder			↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 1 -		
/0/84	generic	Xeon E7 v4/Xeon	↳ Channel 1 Error		
↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 0/1			/0/98	generic	Xeon E7 v4/Xeon
↳ Broadcast			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
/0/85	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Global Broadcast			/0/99	generic	Xeon E7 v4/Xeon
/0/86	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/9a	generic	Xeon E7 v4/Xeon
↳ Channel 0 Thermal Control			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
/0/87	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/9b	generic	Xeon E7 v4/Xeon
↳ Channel 1 Thermal Control			↳ E5 v4/Xeon E3 v4/Xeon D DDRIO Channel 2/3		
/0/88	generic	Xeon E7 v4/Xeon	↳ Interface		
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			/0/9c	generic	Xeon E7 v4/Xeon
↳ Channel 0 Error			↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
/0/89	generic	Xeon E7 v4/Xeon	/0/9d	generic	Xeon E7 v4/Xeon
↳ E5 v4/Xeon E3 v4/Xeon D Memory Controller 0 -			↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit		
↳ Channel 1 Error					

```
/0/9e           generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit
/0/9f           generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit
/0/a0           generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit
/0/a1           generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit
/0/a2           generic      Xeon E7 v4/Xeon
  ↳ E5 v4/Xeon E3 v4/Xeon D Power Control Unit
/0/a3           system      PnP device
  ↳ PNP0b00
/0/a4           system      PnP device
  ↳ PNP0c02
/0/a5           communication PnP device
  ↳ PNP0501
WARNING: output may be incomplete or inaccurate, you
  ↳ should run this program as super-user.
```