Exploring Best Lossy Compression Strategy By Combining SZ with Spatiotemporal Decimation

Xin Liang, † Sheng Di,* Sihuan Li,† Dingwen Tao,‡ Zizhong Chen,† and Franck Cappello*§
*Argonne National Laboratory, IL, USA
sdi1@anl.gov, cappello@mcs.anl.gov
† University of California, Riverside, CA, USA
{xlian007,sli049}@ucr.edu, chen@cs.ucr.edu
‡The University of Alabama, AL, USA
tao@cs.ua.edu
§University of Illinois at Urbana-Champaign, IL, USA

***Abstract*—In today's extreme-scale scientific simulations, vast volumes of data are being produced such that the data cannot be accommodated by the parallel file system or the data writing/reading performance will be fairly low because of limited I/O bandwidth. In the past decade, many snapshot-based (or space-based) lossy compressors have been developed, most of which rely on the smoothness of the data in space. However, the simulation data may get more and more complicated in space over time steps, such that the compression ratios decrease significantly. In this paper, we propose a novel, hybrid lossy compression method by leveraging spatiotemporal decimation under the SZ compression model. The contribution is twofold. (1) We explore several strategies of combining the decimation method with the SZ lossy compression model in both the space dimension and time dimension during the simulation. (2) We investigate the best-fit combined strategy upon different demands based on a couple of typical real-world simulations with multiple fields. Experiments show that either the space-based SZ or time-based SZ leads to the best rate distortion. Decimation methods have very high compression rate with low rate distortion though, and SZ combined with temporal decimation is a good tradeoff.**

I. Introduction

Today's high-performance computing (HPC) applications may produce extremely large volumes of data, resulting in huge storage challenges and I/O performance issues for scientific research. The Hardware/Hybrid Accelerated Cosmology Code (HACC) [1], for example, may produce 60 PB of data to store during one simulation because of up to 3.5 trillion particles to simulate. Such a large amount of data cannot be accommodated even on the IBM Blue Gene/Q "Mira" system—one of today's most powerful supercomputers—because it provides at most 26 PB of storage space to users.

Many state-of-the-art data compressors have been designed to reduce the data size. Lossless compressors such as GZIP [2] and Zstandard [3] cannot work effectively on the compression of floating-point data values, however, because of the random ending mantissa bits in each floating-point value. Accordingly, error-bounded lossy compressors have been thought of as a promising solution that can significantly reduce the data size while controlling the compression errors on demand. Many existing in situ lossy compressors, however, are designed based on individual snapshots, heavily depending on the smoothness of the data in space. These compressors may suffer from low compression ratios since the data in each snapshot may get more and more complicated over the simulation time steps.

In this paper, we explore several strategies by combining time-based compression (such as the decimation method) with space-based compression in terms of the SZ compression model [4], in order to optimize the error-bounded, in situ lossy compression quality for scientific simulations. We choose the SZ compression model because (1) SZ has been one of the best error-bounded lossy compressors based on multiple independent assessments [4], [5] and (2) the SZ compression model allows customization of a specific prediction approach in terms of the simulation data features in order to improve the compression quality. Our contributions are as follows.

- We explore three new data prediction methods based on the time dimension during the simulation. The first is a time-based prediction method for each data point in one snapshot based on its corresponding data value at the previous time step. The second prediction method leverages the downsampling approach and uses a tricubic technique to fit the missing data values in space. The third method involves decimating the snapshots every K time steps and reconstructing the missing snapshots using interpolation method during the decompression.
- We carefully explore the best strategy from among the combinations in terms of the two spatial predictors (SZ's built-in spatial data prediction and Downsampling+tricubic method) and two temporal predictors (prestep data prediction and the decimation method), using two well-known real-world simulation datasets (Hurricane Isabel simulation and molecular dynamics simulation). We also explore the best-fit configuration setting for each strategy.

The rest of the paper is organized as follows. In Section II, we discuss the related work. In Section III, we first review the classic SZ compression model and then present our main idea by combining the time-based compression and space-based compression together in this model. In Section IV, we present our evaluation results in terms of critical metrics such as

compression ratio and rate distortion. In Section V we discuss our conclusions and present ideas for future work.

II. Related Work

Although many error-controlled lossy compressors [4], [6]–[10] have been developed to reduce the amount of data produced during a simulation, they are designed based on the analysis of the spatial correlation of the data, significantly relying on the smoothness of the data in space. However, the data values in some datasets (such as N-body simulation data) may not be consecutive in a snapshot and hence will definitely lead to relatively low compression ratios, as confirmed in our experiment shown later.

In order to address the limitations of space-based compression, some compressors have been developed by leveraging the relatively high smoothness of data in the time dimension. A typical example is NUMARCK [11]. This approach has two critical drawbacks. (1) NUMARCK adopts only the previous step value in the compression during the simulation and stores the original data periodically (i.e., every K time steps), which results in relatively low compression ratios. (2) NUMARCK cannot respect error bound because it predicts data values for any snapshot based on the original data in its previous snapshot. This leads to unexpected error propagation during the decompression because the data used in the decompression have to be **decompressed** values instead of **original** values. By comparison, our compressor strictly respects the compression error bound set by users. We also explore more prediction methods by combining different strategies, such as space-based prediction, time-based prediction, and decimation.

III. Exploring the Best Strategy with Both Spatial Prediction and Temporal Prediction in SZ

In this section, we explore the best strategy for the combination of space-based compression and time-based compression. In what follows, we first review the SZ compression model and then describe several new strategies based on the model.

A. Classic SZ Compression Model

In our prior work, we proposed a prediction-based error-controlled lossy compression model, called SZ, that involves four critical steps: (1) data prediction, (2) linear-scaling quantization, (3) variable-length encoding (Huffman encoding), and (4) lossless compression (Gzip/Zstd). In the first step, the whole dataset is scanned, and each data point is predicted by a Lorenzo [12] predictor based on its neighbor values at the backward directions. In order to guarantee that the compression error always is controlled within the user-set error bound, the neighbor values used in the data prediction must be the decompressed values (i.e., the same as the values used in the decompression) instead of the original data values; otherwise the compression errors would be propagated without controls. In the second step, we compute the difference between the predicted value and original value for each data point and perform a linear-scaling quantization to convert the difference value to an integer number based on the user-set error bound. After converting all the floating-point values to integer numbers in step 2, we adopt a customized Huffman encoding algorithm and a lossless compressor (Zstd) to reduce the data size significantly. Since the classic SZ compressor supports only a space-based prediction method, which may suffer from low compression ratios especially for a relatively complicated dataset with low data smoothness in space, we integrate the time-based compression mechanism based on the SZ model for improving the prediction accuracy.

B. Classic Decimation Compression Model

The decimation compression model (i.e., perform decimation during compression and perform interpolation to reconstruct the missing data during the decompression) has been widely used in the visualization community. It can be categorized into two classes in terms of compressed dimension.

1) Decimation in space: This method often applies uniform sampling during compression. In the course of decompression, the missing data is interpolated according to the sampled data. Although trilinear interpolation is a good choice in terms of interpolation rate, tricubic interpolation can provide better quality at the cost of higher time in most cases. In our experiments, we compare the performance and quality of both trilinear and tricubic interpolation for space-based decimation.

2) Decimation in time: This decimation technique samples the temporal snapshots every K time steps during the simulation and reconstructs the missing snapshots using an interpolation method. Because of the generally low variance of scientific simulation data along the time dimension, we adopt linear interpolation for time-based decimation in this paper, which provides a high compression rate and costs low memory with a comparable compression quality.

C. New Compression Strategies by Combining the SZ Model and Decimation Model

In this section, we explore a new compression scheme for scientific simulation with multiple snapshots by combining the classic SZ compression model with the decimation model. Since there are two options (SZ compression and decimation) for both space-based compression and time-based compression, the combination comprises four strategies.

1) SZST – SZ compression in space and time: This strategy applies the space-based prediction and time-based prediction alternatively during the simulation in the compression step. Specifically, it compresses the data by the classic space-based SZ compression every K time steps and adopts the time-based SZ compression (i.e., pre-step-based prediction + quantization and encoders) for other time steps.

2) SZSDT – SZ compression in space and decimation in time: This strategy compresses the data via space-based SZ for the snapshots selected every K time steps and drops the other snapshots during the compression. During the decompression, it reconstructs the selected snapshots via the space-based SZ and reconstructs the missing snapshots via linear interpolation.

3) DSSZT – decimation in space and SZ compression in time: This strategy compresses the data via space-based decimation for the first snapshot every K consecutive time steps and adopts time-based SZ compression for the remaining K-1 snapshots in each period. Note that it has to decompress the data for the first snapshot in each period to provide the initial prediction for time-based SZ compression used by the next snapshot during compression, an action that may lead to a high overhead on compression rate.

4) DST – decimation in space and time: This strategy applies the space-based decimation on the first snapshot selected every K consecutive time steps and drops the other snapshots during the compression. During decompression, it reconstructs the selected snapshots by a spatial interpolation method (either tricubic interpolation or trilinear interpolation) and reconstructs the missing snapshots via linear interpolation in the time dimension.

We present the pros and cons of the four strategies in Table I in terms of their compression principles on four metrics: guaranteeing error bound (GEB), compression quality (CQ), compression rate/speed (CS), and decompression rate/speed (DS). In the table, F, S, TL, and TC are short for fast, slow, trilinear interpolation, and tricubic interpolation, respectively. Based on the compression principle, DST should be extremely fast in compression (denoted as EXF in the table) because it drops many data points during the compression, but it would suffer from very low decompression rate because of expensive interpolation operations in the decompression. DSSZT and DSSZT can respect the error bound only when the snapshots are compressed by SZ (either in the space or the time dimension), so the percentages of the error-bounded snapshots for them are $\frac{1}{K}\times 100\%$ and $\frac{K-1}{K}\times 100\%$, respectively.

TABLE I
PROPERTIES OF THE FOUR COMPRESSION STRATEGIES

	SZST	**SZSDT**	**DSSZT**	**DST**
GEB	Y	Y: $\frac{1}{K}$,N: $\frac{K-1}{K}$	Y: $\frac{K-1}{K}$,N: $\frac{1}{K}$	N
CQ	high	low	low	lowest
CS	F	EXF	F	EXF
DS	F	S	TL:F, TC:S	TL:F, TC:S

IV. EVALUATION OF COMPRESSION QUALITY

In this section, we evaluate the four strategies discussed earlier, with the best settings for each of them.

A. Experimental Setup

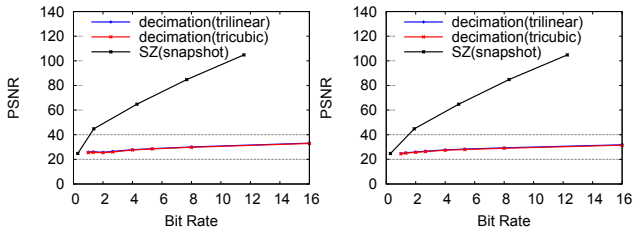
We conduct our experiments on the Bebop cluster at Argonne National Laboratory (Intel Xeon E5-2695 v4 processors, 2.10 GHz, 45M cache). We evaluate the EXAALT data and Hurricane Isabel data for demonstration purposes. Detailed information about the datasets is presented in Table II.

TABLE II
SIMULATION FIELDS USED IN THE EVALUATION

Application	**# Fields**	**# Snapshots**	**Dimensions**	**Examples**
EXAALT	3	83	1,077,290	x, y, z
Hurricane	13	48	100×500×500	CLOUDf48, Uf48 ···

B. Compression Performance for Space-Based Compression

We first explore the compression performance of different space-based compression methods by setting the time-compression interval to 1. The rate distortion of space-based methods is shown in Fig. 1 and Fig. 3 for EXAALT and Hurricane Isabel, respectively, for some fields (x for EXAALT, QCLOUD, and CLOUD and U for Hurricane) and the overall performance because of space limitation.



(a) x (b) overall

Fig. 1. Rate distortion of space-based compression on EXAALT

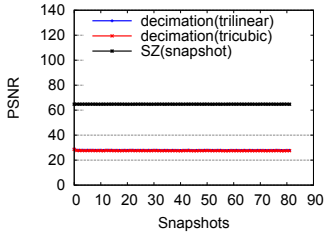
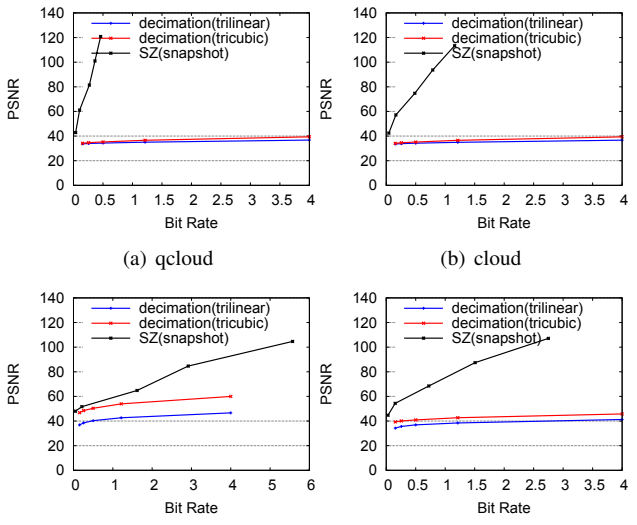


Fig. 2. Multistep PSNR of space-based compression on EXAALT with compression ratio 8 (EXAALT field x)



(a) qcloud (b) cloud

(c) U (d) overall

Fig. 3. Rate distortion of space-based compression on Hurricane Isabel

From Fig. 1, we see that the snapshot-based SZ has dominant performance over the decimation-based methods. Also, in this dataset the linear interpolation has almost the same quality as does the tricubic interpolation. Therefore, it is preferred because of its faster decompression as shown in Table III. On the other hand, the tricubic interpolation has a better compression ratio than does the trilinear interpolation for some fields such as U, at a cost of lower compression rate. Thus, we used tricubic interpolation for these fields and trilinear interpolation for all others in the Hurricane Isabel simulation. Table III and Table IV present the compression rate and de-

compression rate of different space-based compression strategies based on EXAALT data and Hurricane Isabel data, respectively, with the compression ratios of 4, 8, and 16 for EXAALT data and 8, 25, and 64 for Hurricane Isabel data. We see that the spatial-decimation-based compression strategies run very fast (5∼40 GB/s) during the compression, because they omit many data points in each snapshot for reducing the data size. The compression rate and decompression rate are stable for SZ. Decimation+tricubic has a fairly low decompression rate (only 1∼13MB/s) because of its expensive tricubic interpolation. we also observe that the compression/decompression rate of SZ increases with the compression ratio because higher compression ratios indicate easier compression cases for SZ. In addition, high compression ratio also leads to high compression rate for the decimation because of fewer points to sample in the compression.

TABLE III
COMPRESSION/DECOMPRESSION RATE OF SPACED-BASED METHODS ON EXAALT (MB/S)

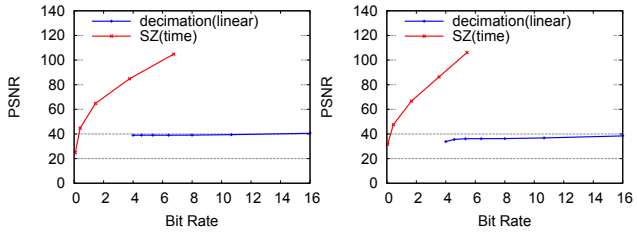
Method	CR≈4	CR≈8	CR≈16
decimation+trilinear (comp)	5083.44	9331.32	13668.97
decimation+tricubic (comp)	5186.57	8859.47	12075.77
SZ_snapshot (comp)	68.01	72.99	76.84
decimation+trilinear (decomp)	246.713	247.564	244.298
decimation+tricubic (decomp)	1.0646	2.08551	4.00
SZ_snapshot (decomp)	43.59	78.04	167.25

TABLE IV
COMPRESSION/DECOMPRESSION RATE OF SPACED-BASED METHODS ON HURRICANE ISABEL (MB/S)

Method	CR≈8	CR≈25	CR≈64
decimation+trilinear (comp)	7715.51	19359.67	36560.61
decimation+tricubic (comp)	7729.18	19207.16	39469.09
SZ_snapshot (comp)	77.88	79.67	83.85
decimation+trilinear (decomp)	33.35	34.28	33.23
decimation+tricubic (decomp)	2.05	6.15	12.67
SZ_snapshot (decomp)	55.83	78.13	121.34

C. Compression Performance for Time-Based Compression

We also investigate the compression performance for the time-based compression strategies by adopting lossless compression on the selected snapshots to compress (i.e., setting the error bound to 1 for the space-based SZ and the sampling distance to 1 for spatial decimation). We adopt linear interpolation considering the overhead on both speed and memory.



(a) EXAALT (b) Hurricane Isabel

Fig. 4. Overall Rate distortion of time-based compression

Figure 4 presents the rate distortion of different strategies based on EXAALT data and Hurricane Isabel data. We observe that the time-based SZ has a much higher compression quality than does the time-based decimation method, which suffers from very low PSNR because of its uncontrolled error during the compression.

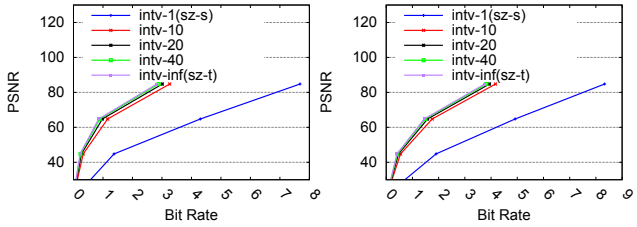
We present the compression/decompression rate for the time-based strategies based on EXAALT and Hurricane Isabel in Table V. It shows that time-based decimation always has a higher compression/decompression rate because of less operations in both compression and decompression.

TABLE V
COMPRESSION/DECOMPRESSION RATE OF TIME-BASED STRATEGIES ON EXAALT AND HURRICANE ISABEL (MB/S)

	EXAALT		Hurricane	
Method	CR≈5	CR≈8	CR≈5	CR≈8
decimation+linear (comp)	inf	inf	inf	inf
SZ_snapshot (comp)	69.11	74.85	77.88	79.67
decimation+linear (decomp)	816.13	729.88	894.11	820.41
SZ_snapshot (decomp)	50.82	80.43	55.83	78.13

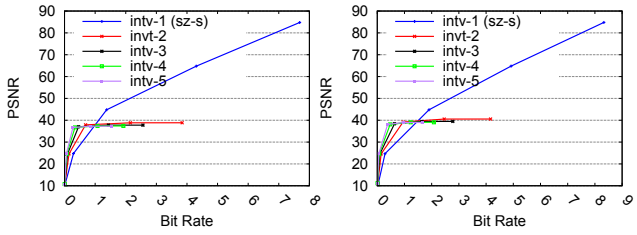
D. Compression performance of the Combined Strategies

Figure 5, Figure 6, Figure 7 present the rate distortion of compressing EXAALT data by using the three combined compressors SZST, SZSDT, and DSSZT, respectively. DST is omitted because its low PSNR (usually <40). Comparing all these results, we conclude that the time-based SZ is the best choice. Its PSNR can reach up to about 100 with a bit-rate of 6, while the PSNRs of other solutions are up to 70.



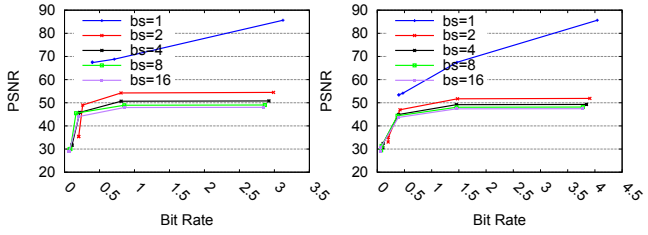
(a) x (b) overall

Fig. 5. Rate distortion of SZST on EXAALT



(a) x (b) overall

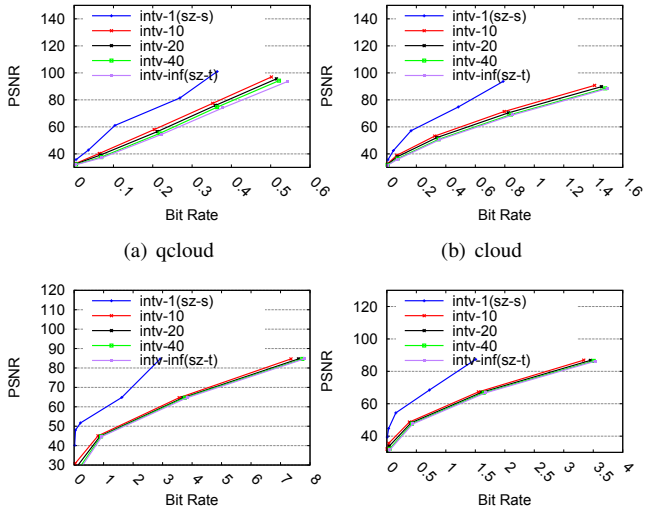
Fig. 6. Rate distortion of SZSDT on EXAALT



(a) x (b) overall

Fig. 7. Rate distortion of DSSZT on EXAALT

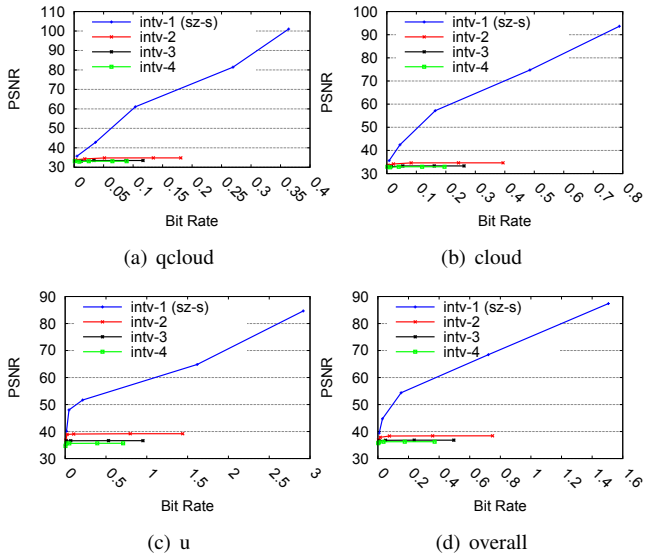
Figure 8, Figure 9, and Figure 10 demonstrate the PSNR based on Hurricane Isabel simulation data. Similar to the compression results based on EXAALT data, SZSDT and DST both suffer from very low PSNR if the time-based decimation is adopted in the compression (i.e., temporal decimation interval > 1), because it does not respect the error bound strictly. Specifically, the larger the decimation interval is, the worse the rate distortion is for the decimation. Comparing these three figures, we also observe that the best compression strategy for the Hurricane Isabel simulation is the snapshot-based SZ, The reason the snapshot-based SZ is better than the time-based SZ on this dataset is that the former has much higher prediction accuracy in space than in time on this dataset. This inspires us to develop an adaptive solution to suit different datasets dynamically, which will be our future work.



(a) qcloud (b) cloud

(c) u (d) overall

Fig. 8. Rate distortion of SZST on Hurricane Isabel

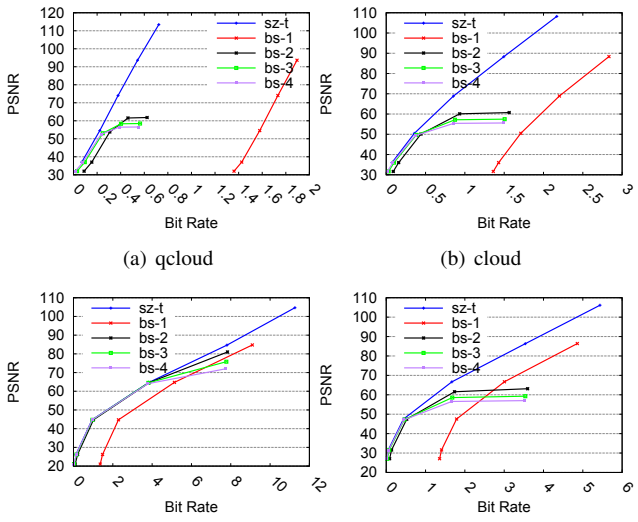


(a) qcloud (b) cloud

(c) u (d) overall

Fig. 9. Rate distortion of SZSDT on Hurricane

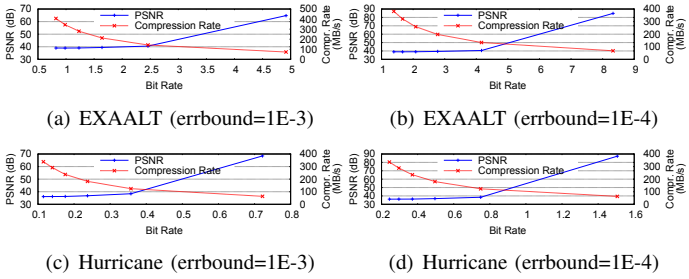
We also evaluate the compression/decompression rate of all the three combined compressors. Experiments show that DST has an extremely high compression rate (20 GB/s on EXAALT and 55 GB/s on Hurricane data), but it has a low decompression rate on the Hurricane Isabel dataset (11.75 MB/s$\sim$42 MB/s). The other three methods have different levels of tradeoff between rate distortion and compression/decompression rate. We present a typical example, SZSDT, in Fig. 11. We evaluate the PSNR and compression rate for SZSDT based on different decimation intervals. In each subfigure, the right-most point corresponds to decimation interval 1 (i.e., SZ without decimation); the second right-most point corresponds to decimation interval 2, and so on. We observe that for decimation interval $\geq$2, the compression rate differs a lot, with little changes of PSNR. As such, SZSDT with decimation interval of 6 is a good solution if the fast compressor is required, or the SZ without decimation is recommended otherwise.



(a) qcloud (b) cloud

(c) u (d) overall

Fig. 10. Rate distortion of DSSZT on Hurricane



(a) EXAALT (errbound=1E-3) (b) EXAALT (errbound=1E-4)

(c) Hurricane (errbound=1E-3) (d) Hurricane (errbound=1E-4)

Fig. 11. PSNR vs. Compression Speed with Bit-rate for SZSDT

V. Conclusion and Future Work

In this paper, we explore four novel compression strategies by combining the SZ compressor with a decimation method in both the time dimension and space dimension. Our experiments using two real-world scientific datasets show that either the space-based SZ or time-based SZ leads to the best rate distortion. DST leads to an extremely high compression rate, while suffering from low decompression rate and very large data distortion because it cannot respect user-set error bounds. SZSDT is a good tradeoff if the users want to avoid high data distortion while still demanding high compression speed.

REFERENCES

[1] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, V. Vishwanath, T. Peterka, J. Insley *et al.*, "Hacc: extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

[2] L. P. Deutsch, "Gzip file format specification version 4.3," 1996.

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

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

[5] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, and Z. Qiao, "Understanding and modeling lossy compression schemes on HPC scientific data," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2018, pp. 348–357.

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

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

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

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

[10] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Isabela for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 524–540, 2013.

[11] Z. Chen, S. W. Son, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "Numarck: machine learning algorithm for resiliency and checkpointing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 733–744.

[12] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 343–348.