Understanding Performance-Quality Trade-offs in Scientific Visualization Workflows with Lossy Compression

Jieyang Chen, David Pugmire, Matthew Wolf, Nicholas Thompson, Jeremy Logan,
Kshitij Mehta, Lipeng Wan, Jong Youl Choi, Ben Whitney, Scott Klasky
Oak Ridge National Laboratory
Oak Ridge, TN, USA
{chenj3, pugmire, wolfmd, thompsonna, loganjs, mehtakv, wanl, choij, whitneybe, klasky}@ornl.gov

***Abstract*—The cost of I/O is a significant challenge on current supercomputers, and the trend is likely to continue into the foreseeable future. This challenge is amplified in scientific visualization because of the requirement to consume large amounts of data before processing can begin. Lossy compression has become an important technique in reducing the cost of performing I/O. In this paper we consider the implications of using compressed data for visualization within a scientific workflow. We use visualization operations on simulation data that is reduced using three different state-of-the-art compression techniques. We study the storage efficiency and preservation of visualization features on the resulting compressed data, and draw comparisons between the three techniques used. Our contributions can help inform both scientists and researchers in the use and design of compression techniques for preservation of important visualization details.**

***Index Terms*—reduction, visualization, adios, sz, zfp, mgard**

I. Introduction

Due to the growing imbalance between the compute and I/O performance of leadership class HPC systems, I/O has become one of the major bottlenecks in scientific computing workflows. For example, the Summit supercomputer [1] at Oak Ridge National Laboratory is 7 times faster than its predecessor, Titan. However, the maximum I/O bandwidth is only 2.5x faster than that on Titan. This imbalance makes it challenging to load data required for computation, and to store large outputs. This imbalance is expected to grow in the next generation of exascale computing systems.

Visualization remains the fundamental mechanism by which scientists extract insight from simulations. To provide this capability, a number of state-of-the-art visualization tools have been developed, including VisIt [2], ParaView [3], and VTK [4]. In order to take advantage of the increase in heterogeneous concurrency, tools such as VTK-m [5] have been developed. Visualization, in general, is I/O bound [6], and so methods for *in situ* processing of data are attractive. *In situ* processing methods are varied, and range from direct resource sharing with the simulation to allocating a dedicated set of resources (*staging*) where simulation data are transferred over the high-speed network [7]–[12].

Scientific simulations will often use a data workflow to manage and process results. These workflows use a combination of *in situ* processing, staging and traditional file-based I/O. As compute capacity continues to grow, simulations will use these larger supercomputers to solve larger and more complex problems which continue to stress the I/O bandwidth on foreseeable HPC systems. Additionally, with the growth in concurrency, I/O bandwidth per core is also significantly reduced, further exposing the I/O cost as shown in Figure 1.

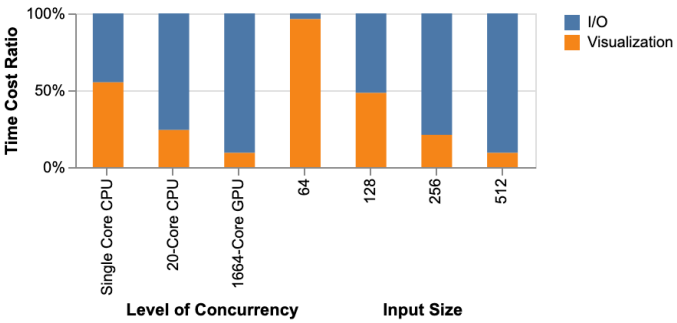


Fig. 1. Measuring the time cost ratio of I/O (file read and write) and visualization related computations (Marching cubes and rendering) in a visualization workflow when visualizing a single image using different levels of computing concurrency with input data size fixed at 512 (left three bars) and different input data sizes with the level of computing concurrency fixed to a 1664-Core GPU (right four bars). Results show that increasing either level of computing concurrency or input data size would make I/O more expensive in visualization workflows.

Timely feedback from visualization is critical for monitoring and understanding simulation results, and the I/O costs can impede its usability. Although reducing image resolution can effectively reduce visualization time, information and features can be lost, and it does not address the fundamental bottleneck of access to data.

Lossy compression has been an active area of research to mitigate pressure on I/O systems. These techniques aim to reduce the size of the data and preserve the information content as controlled by a set of user-specified parameters. This is very attractive for visualization as it effectively reduces the data access bottleneck and gives users the flexibility to control the trade-off between quality and cost. While compression techniques are an active area of research, the impact of

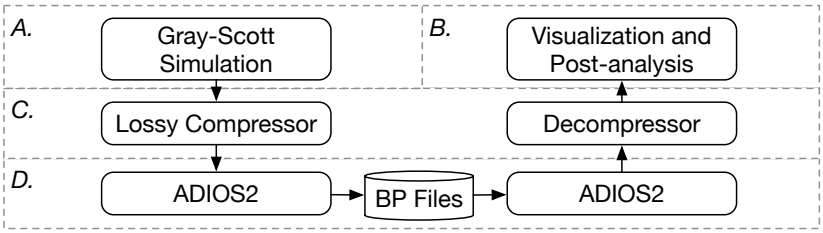


Fig. 2. Our testing visualization workflow used in this work. The four components (A-D) in the above figure are discussed in Section III A-D

lossy compression within a visualization workflow is not well studied. In this paper we study the impact of lossy compression on a representative scientific workflow. Our focus is on storage efficiency and visualization quality. First, storage efficiency is critical as access to data, either from disk in a post processing scenario, or a staging scenario where data are transferred over the network between memory regions, is a significant bottleneck for visualization operations. Second, the quality of visualization results, which are often composed of a number of operations on the data, is critical to scientific understanding of the nature of the simulation.

Specifically, the contributions of this paper include:

- Study the impact of different lossy compressors, along with parameters on visualization and workflow performance.
- Identification of useful metrics for evaluating the quality feature and information preservation using visualization on lossy data.
- Study the impact of different lossy compressors on visualization in different computing environments i.e., single-core CPU, multi-core CPU, and GPU.

II. Background

We evaluate three popular lossy compression algorithms: `MGARD`, `SZ`, and `ZFP`.

`MGARD`, i.e., the multigrid adaptive reduction of data (`MGARD`) [13], [14] is a recently developed technique for reduction of scientific data. The algorithm draws on the ideas of multigrid methods, which offers a high degree of flexibility and guaranteed provision on loss incurred by the reduction of the data. The loss is measured in a problem-specific Besov norm, and allows preservation of derived quantities, e.g., energy spectra.

`SZ` [15]–[18] is a lossy compressor based on polynomial interpolation and extrapolation. It offers three modes of error bound settings: absolute (`SZ-ABS`), relative (`SZ-REL`), and point-wise relative (`SZ-PWR`).

`ZFP` [19] is a general purpose compressor for floating-point arrays, similar in character to a block discrete cosine transform, that supports fine-grained read and write access enabled by fixed-rate compression.

A. Data Exchange Methods in Scientific Visualization Workflows

Scientific visualization workflows must exchange data between numerical solvers and visualization applications. We use the Adaptable Input/Output System (ADIOS) [20] to manage this exchange. ADIOS is a high-performance data transfer and management framework designed for extreme-scale scientific data. It uses the `BP` self-describing file format for representing data. It uses a publisher-subscriber framework for data exchange, where processes subscribe to a data stream that is typically generated by a running simulation. Its performance is mainly bound by the I/O bandwidth of the filesystem.

B. Gray-Scott Simulation

As a representative example of the type of workflows encountered in scientific visualization, we use a numerical solution of the Gray-Scott model of reaction diffusion [21]

$$\frac{\partial U}{\partial t} = r_u \nabla^2 U - UV^2 + f(1-U)$$
$$\frac{\partial V}{\partial t} = r_v \nabla^2 V + UV^2 - (f+k)V$$

If U and V are the concentrations of two chemical species, then this models the reaction

$$U + 2V \rightarrow 3V, V \rightarrow P$$

where P is an inert species. It is chosen because it is a hyperbolic nonlinear coupled system of PDEs which captures many of the challenges present in cutting-edge scientific simulations.

III. Design of the Testing Visualization Workflow

Our workflow consists of four components: a scientific simulation code, data compressors/decompressors, data management, and visualization/post-analyser. This is summarized in Figure 2.

A. Simulation Code

The Gray-Scott simulation serves the role of data producer in our workflow. The simulation has two settings: the spatial resolution of simulation grid N, which creates a uniform three dimensional grid of size N^3, and a random noise parameter, η, which perturbs the initial conditions of the problem. In our experiments we use a grid resolution of $N = 512$, and a noise parameter of $\eta = 0.5$ to increase the amount of meaningful data (features) in all of our workflow examples.

TABLE I
PERFORMANCE AND QUALITY METRICS USED IN THIS WORK

	Metrics	Description	Tool Used
Performance	Compression ratio	Defined by the ratio of the space needed to store original data vs. compressed data	Our post-analyser
	Time cost of workflow	Including time cost for I/O (file read and write) using ADIOS2 and visualization related computation (Marching cubes and rendering) using VTK-h	Our post-analyser
Quality	PSNR	Defined as $10\log_{10}(\frac{R^2}{\text{MSE}})$ where MSE represents mean square error and R is the maximum fluctuation in the input	Z-Checker
	Relative L^∞ error	Defined as $\frac{\text{Max Absolute Error}}{\text{Max Input}}$	Our post-analyser
	Relative error of iso-surface area	Visualization specific quality metrics	VisIt
	Relative error of number of connected components		

B. Visualization and Analysis

We use three different tools for visualization and analysis of the compressed simulation outputs: VisIt, a production visualization tool, an iso-surface and rendering application based on the VTK-m library for portable performance on single-/multicore CPU and GPUs, and Z-Checker [22] for data analysis. We use the visualization application to measure the time cost for each visualization step, as well as evaluation of the visualization quality. As a measure of quality, we consider the surface area of the computed iso-surfaces as well as the number of connected components (features). For convenience, we use VisIt to compute the surface area and connected component count from the output of the VTK-m application. Finally, we build post-analysis tools to capture two kinds of data quality metrics: peak signal-to-noise ratio (PSNR) and relative L^∞ error. **Table I** summarizes the performance and quality metrics used in this work.

C. Data Compressors/Decompressors

We use three different lossy compression techniques on the output data from the Gray-Scott simulation. For I/O, we are using ADIOS, which has read and write support for each of the compression techniques. The post processing tools use the ADIOS library to read and decompress the data for visualization and analysis. We discuss ADIOS in greater detail in §III-D.

Each of the three compression techniques has parameter(s) that allow users to specify an error bound. Each compression technique will use this error bound to find an acceptable reduced representation of the data. The definition of error bound differs from compressor to compressor, and we refer the interested reader to the works cited in §VI.

To help facilitate this work and without having to run simulation code against each combination of compressor and error bound, we extract the compression and decompression process as explicitly separate operations in between simulation code/visualization application and ADIOS. It does not impact validity of time measuring work, but can make our testing more efficient and accurate.

D. Data Management System

Data sharing between the Gray-Scott simulation code and visualization applications is done through ADIOS, which is widely used data management systems that offers state-of-the-art performance. In our workflow, we use ADIOS to perform file-based data sharing in our testing workflow.

IV. EXPERIMENTS

A. Experimental Environment

Our visualization workflow was run on a workstation whose hardware specification is shown in **Table II**.

TABLE II
WORKSTATION HARDWARE SPECIFICATION

CPU: 20-core Intel Xeon E5-2687W v3 @ 3.1 GHz
RAM: 32 GB
GPU: Two Nvidia Quadro M4000 GPUs with 8 GB RAM in each GPU
Disk: Seagate 4TB SATA III (max sustainable bandwidth: 182 MB/s for read and 179 MB/s for write)

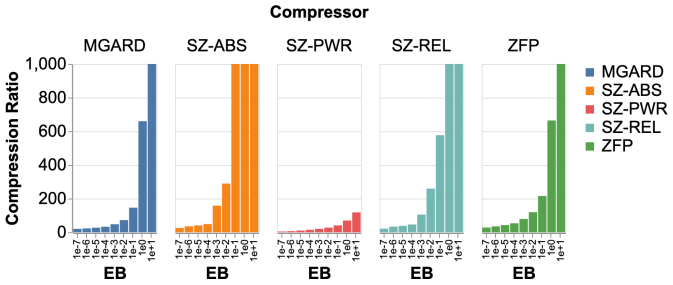
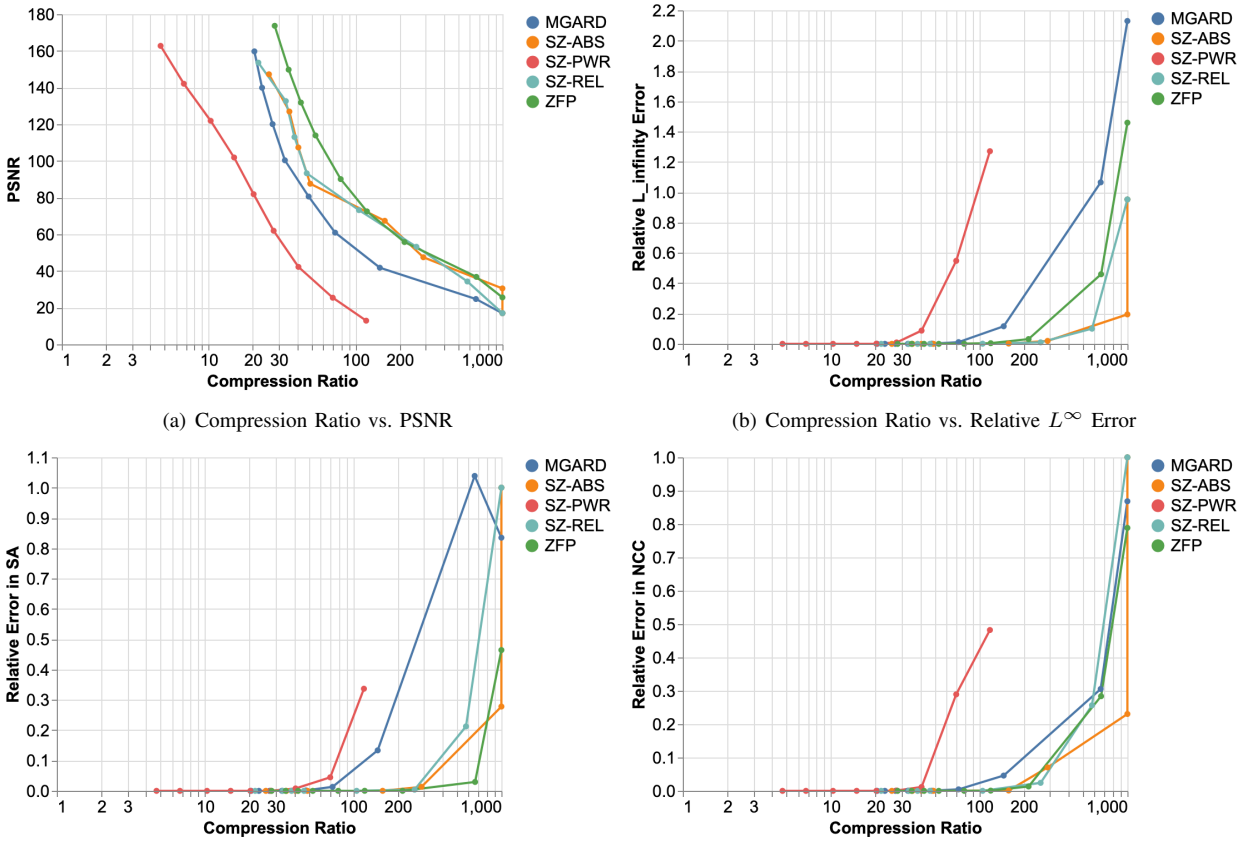


Fig. 3. Comparison on Compression Ratio

The Gray-Scott simulation implementation used in this work is available at [23], along with the current latest VTK-m (version 1.4), Visit (version 3.0.1) and ADIOS 2 are built with GCC 7.4.0. VTK-m is also built with OpenMp 7.4.0 and CUDA 10.1.168. To enable compression, ADIOS is linked with the latest available release of `MGARD` (version 3), `SZ` (version 2.1.6), and `ZFP` (version 0.5.5) compression libraries.

B. Experimental Design

We run Gray-Scott on a grid of size 512^3 for 10,000 time steps and write the values of U and V every 1000 time steps. As we notice that the simulation enter states with complex data patterns only after 3000 time steps, we only evaluate the



(a) Compression Ratio vs. PSNR

(b) Compression Ratio vs. Relative L^∞ Error

(c) Compression Ratio vs. Relative Error in Surface Area (SA)

(d) Compression Ratio vs. Relative Error in Number of Connected Components (NCC)

Fig. 4. Compression Ratio vs. Quality

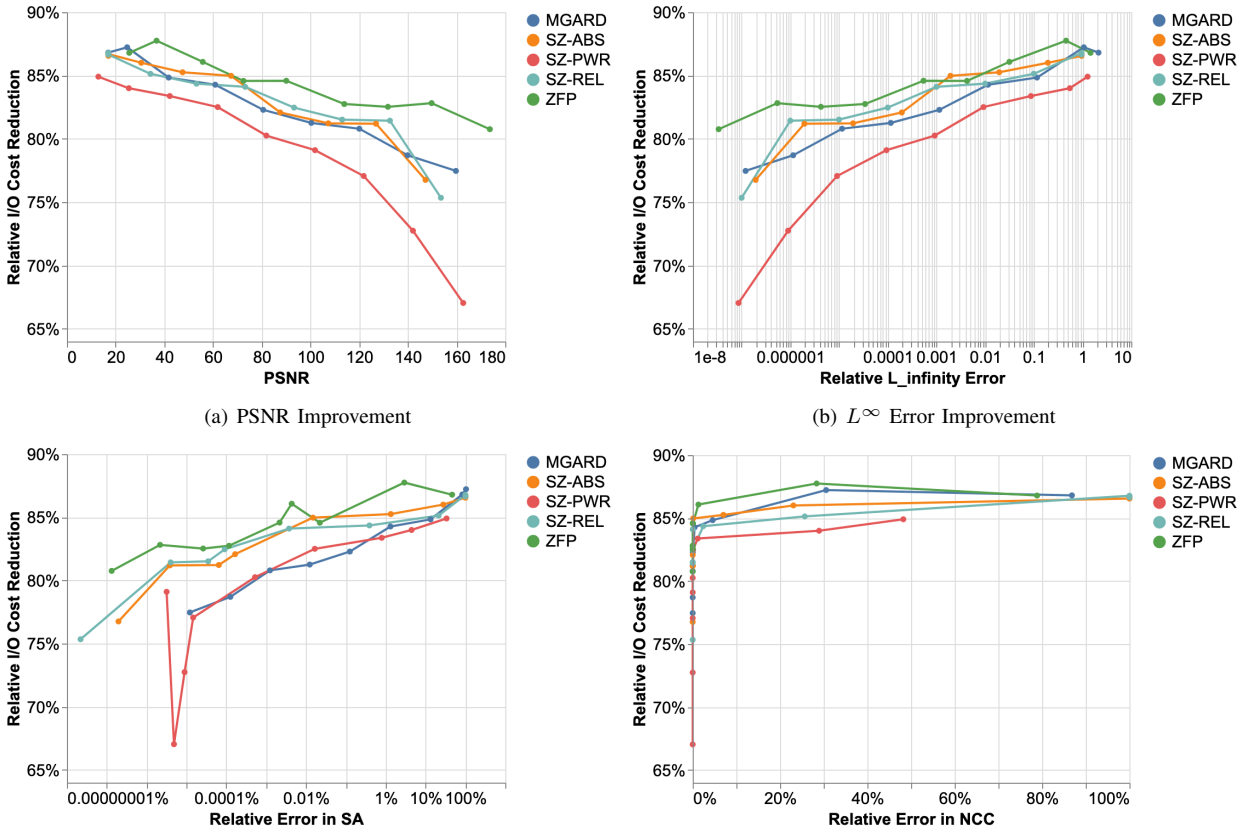
data from timesteps 3000 to 10,000. We compress, visualize, and analyze the variable V. As stated previously, we evaluate compression using the `SZ`, `ZFP`, and `MGARD` compression libraries. The default settings of each compressor are used except for the error bound. The error bound ranges are set so that they reach both minimum possible compression and maximum possible compression where the compressed data are severely distorted i.e., the relative errors of surface area or number of connected components reach or exceed 30%. Iso value for the surface generation is set to 0.1. We show results of running VTK-m under three configurations: single-core CPU, 20-core CPU, and GPU. The time taken to perform post-analysis as shown in **Table I** is not counted towards the time taken to run the workflow.

C. Error Bound vs. Compression Ratio

First, we study how the compression ratio changes as we set different error bounds. The error bounds are defined differently in different compressors or different compressing modes, so the compression ratio of different cases are not comparable even if they have the same error bound. Thus, the results are not used to compare different compressors but to show the possible compression ratios using different error bounds. As shown in **Fig. 3**, all compressors exhibit a wide range of compression ratios. `MGARD` and `ZFP` give better compression ratios when the error bounds are relative tight. On the other hand, `SZ` gives better compression ratios when the error bounds are relative loose.

D. Compression Ratio vs. Quality

Comparing the compression ratio only shows how much of the original data is reduced but does not show how much useful information is preserved or lost. So, in this subsection we compare the compression quality of each compressor and show how that is related to the compression ratio as shown in **Fig. 3**. Here we use four quality metrics mentioned in **Table I**. **Fig. 4**(a) show that `MGARD`, `ZFP`, `SZ-PWR` give good PSNR for low compression ratios (less than 30x). The PSNR drops similarly for all compressors as the compression ratio increases except for `MGARD`, which drops faster. Finally, they all converge to 20-40 as the compression ratio grows beyond 200x. **Fig. 4(b)** shows that the relative L^∞ error stays close to 0 when the compression ratio is less than 20x. Then it increases fast after around 45x. It grows faster for `MGARD` as compared to other compressors. **Fig. 4(c) - (d)** shows that with 35x compression ratio the loss on surface area is almost indistinguishable compared with uncompressed data. The number of connected components is less sensitive



(a) PSNR Improvement

(b) L^∞ Error Improvement

(c) Relative Error in Surface Area (SA)

(d) Relative Error in # of Connected Components (NCC)

Fig. 5. I/O cost reduction against four quality metrics

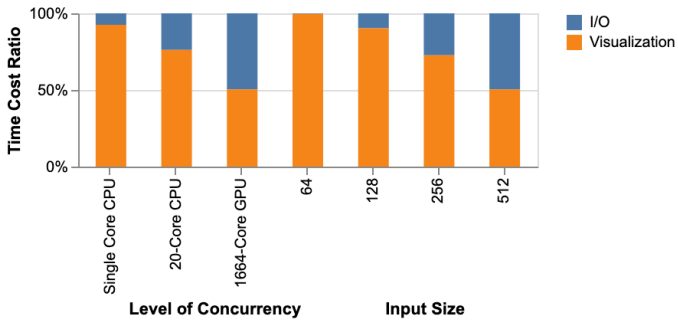


Fig. 6. Time cost ratio when lossy compression is applied while keeping at most 1% change to surface area. The best lossy compressor and error bound that leads to the most I/O reduction and meets feature loss requirement is chosen for each case.

to errors, and it only shows considerable accuracy loss after compressing data towards 40x. Also, less than 20% distortion on surface area and the number of connected components is achievable with at least 100x compression ratio if proper compressors and error bounds are set.

E. Reduction of Time Cost for I/O

An important benefit of data compression is reduction in I/O cost. We show how I/O cost is reduced given different levels of loss metrics. As shown in **Fig. 5**, out test cases reduced 66% - 87% of the I/O cost that includes both file read and write time. We can see a clear trade-off that shows that more loss of information or features can bring greater reduction in the I/O cost. We also notice some oscillation due to the variability of the I/O bandwidth and contention.

F. Best Reduction of Time Cost for I/O Given Different Quality Loss Requirements

Finally, we show how the time cost ratio changes as mentioned in **Fig. 1**. As in **Fig. 1**, **Fig. 6** shows the time cost ratios when visualizing a single image using different levels of computing concurrency with input data size fixed at 512 (left three bars) and different input data sizes with the level of computing concurrency fixed to a 1664-Core GPU (right four bars). Different from the tests in **Fig. 1**, we apply lossy compression. The I/O cost in time cost ratio of each case is obtained by choosing the compressor and error bound that gives the maximum I/O reduction while meets our quality loss requirement i.e., less than 1% change in surface area. Compared with **Fig. 1**, we can see that the time cost ratio for I/O has considerable dropped, which shows great potentials of using lossy compression to reduce I/O in visualization workflows.

V. Discussion and Future Work

In this work we have sought to understand the compression ratios and visualization and analysis quality of data output from lossy compressors. We have conducted experiments using several state-of-the-art lossy compression techniques and evaluated them against these two metrics. Our experiments showed that lossy compression can achieve significant compression ratios that can mitigate pressure on the I/O system. This bodes well for workflows using in situ staging techniques where the compressed data are moved over the network. Our experiments highlight the importance of feature preservation under visualization and analysis operations. This provides insight into the types of compression to use and the appropriate parameter settings. We feel that these, and findings through future studies, can be helpful in guiding the research efforts in lossy data compression.

In the future we plan to extend these tests to large-scale distributed simulations. Tests at great levels of concurrency and larger data would allow us to better understand characteristics in production. In particular, at scale, the time component becomes more meaningful as costs of compression and decompression can be amortized over larger concurrency. We are particularly interested in in situ workflows where data staging can further reduce the I/O bottleneck for visualization as the file system is entirely avoided. This will also present the opportunity to study more complex workflows and understand the impacts of lossy compression on both cost and quality. We are also interested in studying different classes of simulations and understanding the relationship between feature preservation under lossy compression.

VI. Related Work

The impacts that lossy compression make on visualization have been studied in many recent works. [14] shows `MGARD`'s compression impact on the visualization results of the velocity at a particulate time step of the pseudo-spectral simulation of forced isotropic turbulence. They obtain visually indistinguishable visualization results from reconstructed data that was compressed in 4-fold by setting the error bound 10^{-2}. [13] applies `MGARD` lossy compression on High Reynolds number channel flow data and Community Earth System Model atmosphere data. They report indistinguishable visualization results when error bound is less or equal to 10^{-1}. We obtain similar conclusion that setting the error bound to be 10^{-2} and 10^{-1} for `MGARD` give 0.1% and 1% surface area error that is visually indistinguishable. [15] studies impact on visualization when applying `SZ` and `ZFP` using dark matter density field data from NYX simulation and the CLOUDf field data in Hurricane-ISABELA. They show that `SZ` preserves more details compared with `ZFP` while keeping similar compression ratios. [19] shows the impacts on volumetric ray tracer when integrated with `ZFP`. They show that `ZFP` brings indistinguishable visualization impact when keeping compression ratio equal or less than 4bpd (bits-per-double).

References

[1] J. Wells, B. Bland, J. Nichols, J. Hack, F. Foertter, G. Hagen, T. Maier, M. Ashfaq, B. Messer, and S. Parete-Koon, "Announcing supercomputer summit," ORNL (Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States)), Tech. Rep., 2016.

[2] H. Childs, "Visit: An end-user tool for visualizing and analyzing very large data," 2012.

[3] J. Ahrens, B. Geveci, and C. Law, "Paraview: An end-user tool for large data visualization," *The visualization handbook*, vol. 717, 2005.

[4] W. J. Schroeder, L. S. Avila, and W. Hoffman, "Visualizing with vtk: a tutorial," *IEEE Computer graphics and applications*, vol. 20, no. 5, pp. 20–27, 2000.

[5] K. Moreland, C. Sewell, W. Usher, L.-t. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K.-L. Ma, H. Childs *et al.*, "Vtk-m: Accelerating the visualization toolkit for massively threaded architectures," *IEEE computer graphics and applications*, vol. 36, no. 3, pp. 48–58, 2016.

[6] H. Childs, D. Pugmire, S. Ahern, B. Whitlock, M. Howison, G. Weber, and E. W. Bethel, "Extreme scaling of production visualization software on diverse architectures," *IEEE Computer Graphics and Applications*, vol. 30, pp. 22–31, 05 2010.

[7] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "Datastager: scalable data staging services for petascale applications," *Cluster Computing*, vol. 13, no. 3, pp. 277–290, 2010.

[8] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky, "Just in time: adding value to the io pipelines of high performance applications with jitstaging," in *Proceedings of the 20th international symposium on High performance distributed computing*. ACM, 2011, pp. 27–36.

[9] F. Zheng, H. Zou, G. Eisenhauer, K. Schwan, M. Wolf, J. Dayal, T.-A. Nguyen, J. Cao, H. Abbasi, S. Klasky *et al.*, "Flexio: I/o middleware for location-flexible scientific data analytics," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. IEEE, 2013, pp. 320–331.

[10] F. Zheng, H. Yu, C. Hantas, M. Wolf, G. Eisenhauer, K. Schwan, H. Abbasi, and S. Klasky, "Goldrush: resource efficient in situ scientific data analytics using fine-grained interference aware execution," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 78.

[11] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci *et al.*, "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 49.

[12] Y. Tian, S. Klasky, W. Yu, H. Abbasi, B. Wang, N. Podhorszki, R. Grout, and M. Wolf, "A system-aware optimized data organization for efficient scientific analytics," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*. ACM, 2012, pp. 125–126.

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

[14] ——, "Multilevel techniques for compression and reduction of scientific datathe univariate case," *Computing and Visualization in Science*, vol. 19, no. 5-6, pp. 65–76, 2018.

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

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

[17] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE, 2016, pp. 730–739.

[18] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 179–189.

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

[20] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield *et al.*, "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 7, pp. 1453–1473, 2014.

[21] J. E. Pearson, "Complex patterns in a simple system," *Science*, vol. 261, no. 5118, pp. 189–192, 1993.

[22] D. Tao, S. Di, H. Guo, Z. Chen, and F. Cappello, "Z-checker: A framework for assessing lossy compression of scientific data," *The International Journal of High Performance Computing Applications*, vol. 33, no. 2, pp. 285–303, 2019.

[23] "Gray-Scott Simulation Code," https://github.com/pnorbert/adiosvm/tree/master/Tutorial/gray-scott, [Online; accessed 2019].