Unbalanced Parallel I/O: An Often-Neglected Side Effect of Lossy Scientific Data Compression

Xinying Wang*, Lipeng Wan†, Jieyang Chen†, Qian Gong†, Ben Whitney†, Jinzhen Wang‡,
Ana Gainaru†, Qing Liu‡, Norbert Podhorszki†, Dongfang Zhao*, Feng Yan*, and Scott Klasky†
*University of Nevada, Reno
Email: xinyingw@nevada.unr.edu, dzhao@unr.edu, fyan@unr.edu
†Oak Ridge National Laboratory, Oak Ridge, TN 37830
Email: {wanl, chenj3, gongq, whitneybe, gainarua, pnorbert, klasky}@ornl.gov
‡New Jersey Institute of Technology, Newark, NJ 07102
Email: {jw447, qing.liu}@njit.edu

***Abstract*—Lossy compression techniques have demonstrated promising results in significantly reducing the scientific data size while guaranteeing the compression error bounds. However, one important yet often neglected side effect of lossy scientific data compression is its impact on the performance of parallel I/O. Our key observation is that the compressed data size is often highly skewed across processes in lossy scientific compression. To understand this behavior, we conduct extensive experiments where we apply three lossy compressors MGARD, ZFP, and SZ, which are specifically designed and optimized for scientific data, to three real-world scientific applications Gray-Scott simulation, WarpX, and XGC. Our analysis result demonstrates that the size of the compressed data is always skewed even if the original data is evenly decomposed among processes. Such skewness widely exists in different scientific applications using different compressors as long as the information density of the data varies across processes. We then systematically study how this side effect of lossy scientific data compression impacts the performance of parallel I/O. We observe that the skewness in the sizes of the compressed data often leads to I/O imbalance, which can significantly reduce the efficiency of I/O bandwidth utilization if not properly handled. In addition, writing data concurrently to a single shared file through MPI-IO library is more sensitive to the unbalanced I/O loads. Therefore, we believe our research community should pay more attention to the unbalanced parallel I/O caused by lossy scientific data compression.**

I. Introduction

As several exascale supercomputers are anticipated to be operational in the next a few years, scientific applications running on those machines are projected to generate massive amount of data at enormous velocity. For example, nowadays the X-point Gyrokinetic Code (XGC) [1] developed by the Princeton Plasma Physics Laboratory can easily produce more than 1PB of data per day when running on the OLCF's Summit supercomputer. As a conservative estimate, the data rate will increase to 10PB per day if running on an exascale supercomputer. Although data storage technologies have also improved tremendously over the years, absorbing data generated at such high rates is almost an impossible mission for most of the data storage systems built under rational budget. To address this critical and challenging issue, multiple lossy compression techniques that are specifically designed and optimized for the data generated by scientific applications have been proposed in recent years. Promising results from existing studies [2]–[5] demonstrate that these lossy compression techniques can significantly reduce the size of scientific data while guaranteeing that the compression error is within certain bound.

Since the data size can be effectively reduced by lossy compression, it is natural for scientists to expect the overhead caused by writing or reading their data can also be reduced accordingly. For example, if the size of the data is reduced by 100X, it is reasonable for scientists to believe that their data can be written out about 100X faster. This is true when their codes are run at small scale and only small amount of data is written out using a few processes. However, for data-intensive scientific applications running with massive parallelism on high-performance computing systems, the I/O performance does not simply depend on the size of the data. It also depends on how efficient the concurrent I/O bandwidth can be utilized. Ideally, the maximal concurrent I/O throughput is achieved if all the processes write out or read in the same amount of data. This is why when scientists configure their simulations, they tend to divide the global simulation space into regions of equal size and assign one region to each process to ensure that the I/O loads from all processes are balanced. If the I/O loads among processes are unbalanced, meaning that some of the processes need to write out or read in much more data than others, the overall I/O throughput would decrease as the I/O time is determined by the slowest process that finishes the I/O. Unfortunately, when we apply lossy compression to the data on each process, this I/O imbalance issue often occurs. Specifically, as the data assigned to each process after domain decomposition are often heterogeneous in nature, the information density of these data portions usually varies. Therefore, applying lossy compression to such heterogeneous domain decomposed data would inevitably result in significantly different compressed data sizes across the parallel processes and cause the imbalance of the parallel I/O loads.

To verify the aforementioned conjecture and characterize the imbalance in compressed data size, we apply three widely used lossy compressors MGARD [2], [3], [6], [7], ZFP [4], and SZ [5] to the datasets generated by three real world scientific applications: Gray-Scott simulation, WarpX [8] and XGC [1],

[9]. We aim to answer the following questions.

- If the original data is evenly decomposed and assigned to each parallel process, would the compressed data still be imbalanced among the processes?
- Would imbalance exist in compressed data regardless of which compression method is used?
- How imbalanced can the compressed data sizes be?
- How would the imbalance of the compressed data sizes affect the overall I/O performance?

Our analysis shows that the compressed data is always skewed even if the original data is evenly decomposed and such skewness exists in all the compressed data from different compressors. The difference in compressed data can be more than 10X and Weibull distributions can be used to fit the sizes of the compressed data on each process. During the write out or read in process of scientific applications, the overall write or read performance is usually bounded by the slowest process. Therefore, the skewness in compressed data can potentially degrade the parallel I/O performance if not handled carefully. To demonstrate such negative impacts caused by unbalanced parallel I/O due to skewed compression data, we conduct experiments on OLCF's Andes cluster [10]. In our experiments, we launch tests to simulate the highly skewed data among processes caused by lossy compression and measure the write performance when different write patterns are used. From the results, we observe that the I/O imbalance caused by lossy compression can significantly reduce the efficiency of I/O bandwidth utilization if the processes that own high information density data blocks happen to run on the same compute node. Moreover, witing data concurrently to a single shared file through MPI-IO library is more sensitive to the unbalanced I/O loads.

In summary, the paper makes the following contributions. First, we characterize the skewness of three real world compressed scientific data by studying three widely used data compressors. Second, we analyze the potential impacts of the skewed compressed data on parallel I/O performance. Finally, we compare the write time and write throughput under different use scenarios.

II. Analysis of Lossy Compressed Data Size

Scientific codes are often run with a large number of processes in parallel on high-performance computing systems to achieve satisfactory speedups. Particularly, during the execution of these codes, each process is often assigned a single or multiple portions of the data to operate on, which is known as domain decomposition or "data parallelism". For instance, when scientists launch an MPI-based particle-in-cell simulation code, each MPI process is assigned to simulate the movements of particles in certain areas of the entire simulation region. Since the movements of particles are driven by complex physics models, the information density of the data that each process operates on is usually quite different. In some areas, the large distribution of the particles leads to high information density, while in others the data is sparse due to the lack of physics phenomena. This type of *heterogeneity* in domain-decomposed data, is commonly observed in many other parallel scientific applications [9].

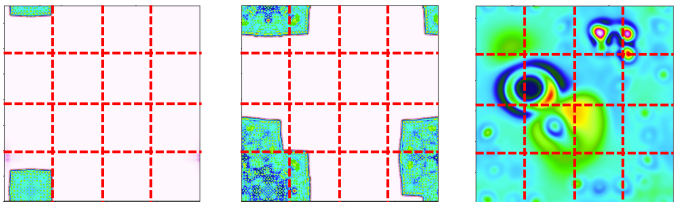

Fig. 1: Different simulation steps of domain decomposition in parallel scientific applications.

Fig. 1 shows an example of the domain decomposition in a parallel Gray-Scott simulation [11]. Each of these three sub-figures illustrates the concentration of a chemical species at different simulation steps. The simulation is run by 16 MPI processes and each of them operates $\frac{1}{16}$ of the entire simulation region. In each sub-figure, we use red dashed lines to differentiate the areas each process owns. The first sub-figure shows that the data on most of the processes is sparse. In the second sub-figure, the information density of data increases and demonstrates more differences across processes. In the third sub-figure, the data on each process has rich information content, but their characteristics are quite dissimilar. According to the information theory [12], the size of the compressed data is strongly correlated with the information content which is defined by the entropy in the original data. If we apply lossy compression to the data on each process, our conjecture is that the sizes of the data after compression would be significantly different from each other in many real world scenarios since the data on each process has quite different information density.

To verify our conjecture, we conduct characterization studies by applying different lossy compressors to datasets generated by three real world scientific applications. The three lossy compressors we select for our study are MGARD [2], [3], [6], [7], ZFP [4], and SZ [5]. These lossy compressors are specifically designed and optimized for compressing scientific data and their effectiveness are verified by existing works [13]. The datasets we use in our study are generated by three parallel scientific codes. Gray-Scott is a 3D 7-point stencil code to simulate Gray-Scott reaction diffusion model. It's a simple system simulating the time evolution of the spatial distributions of two interacting chemical concentrations. And parameters for the simulation such as the diffusion coefficient can be easily adjusted by the user. WarpX [8] is an advanced multi-platform electromagnetic Particle-In-Cell code. It supports many features including Perfectly-Matched Layers (PML), mesh refinement, and the boosted-frame technique. In addition, WarpX includes load balancing capabilities to achieve better performance. XGC [1], [9] is a gyrokinetic particle-in-cell code, which specializes in the simulation of the edge region of magnetically confined thermonuclear fusion plasma. The simulation domain can include the magnetic separatrix, magnetic axis, and the biased material wall. All of

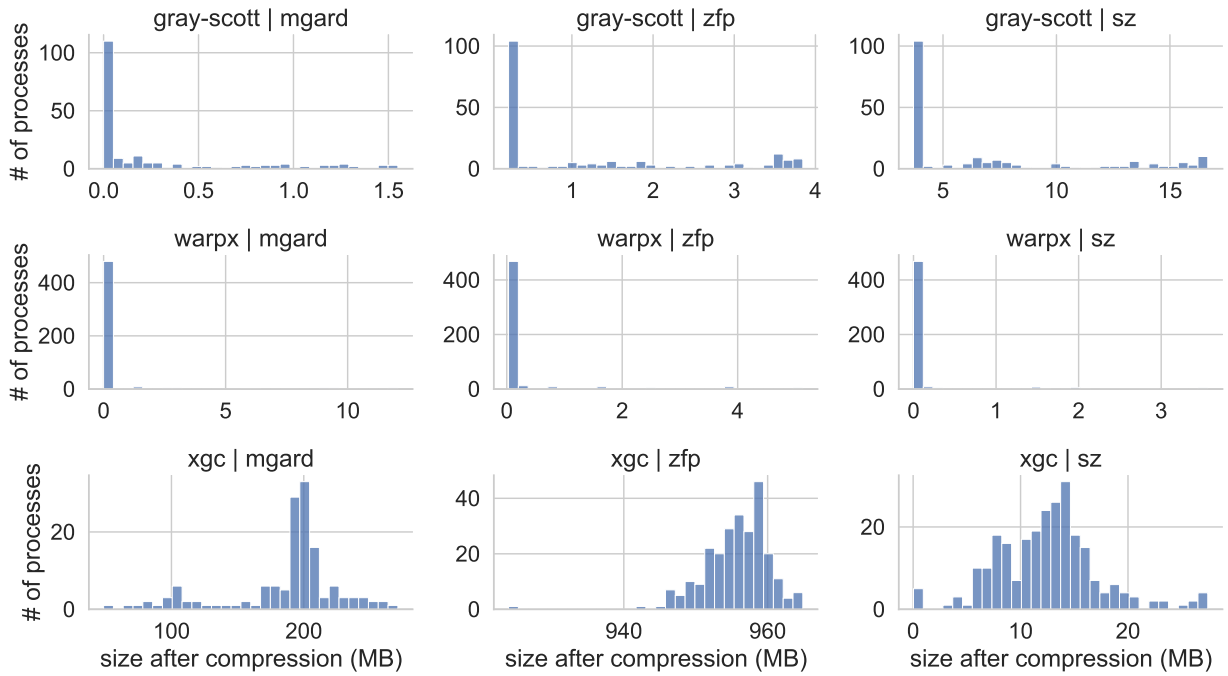

Fig. 2: Distribution of data sizes among processes after applying different lossy compressors to three scientific datasets.

these datasets are written out through ADIOS2 [14], a library that manages the parallel I/O and stores the data in a self-describing format. The metadata of this self-describing format contains rich information about how the dataset is generated, such as which area each data block belongs to in the global array, which MPI process produces which data blocks, etc. By leveraging such information, we can launch a job to read in the exact data blocks produced by each MPI process, and compress them using different lossy compressors. Then we study how different the sizes of the compressed data can be among all the processes. Since different compressors might adopt different error metrics, when we choose the error bounds for each compressor, we try our best to make the peak signal-to-noise ratio (PSNR) of the entire compressed data produced by each compressors fall into a similar range. One thing we would like to emphasize here is that our results cannot be used to indicate which lossy compressor is better in terms of compression ratio since the PSNR of the compressed data from each compressor are not exactly the same. In this study, we only focus on how skew the distribution of the data sizes among processes can be after lossy compression.

For the dataset generated by the Gray-Scott simulation, the entire data is evenly decomposed and assigned to 192 processes. The size of the original data each process owns is 18MB. As shown in Fig. 2, after the lossy compression, the sizes of the data on most processes are reduced to less than 0.5MB. However, no matter which lossy compressor is used, the compressed data on a few processes are significantly larger due to their high information density. For example, when SZ is used, the compressed data owned by 5% of the processes are on average more than 10X larger than other processes. As a result, the distribution of data sizes among processes after lossy compression is highly skewed.

When we use the dataset generated by the WarpX simulation for our experiments, the sizes of the compressed data on almost all of the processes are similarly small since the entire dataset is very sparse. However, there are still several processes that have much larger data after compression. The compressed data on these processes can be hundreds of times larger than the minimal data size among all the processes. When we apply different lossy compressors to the data generated by the XGC simulation, the sizes of the compressed data on each process are also dispersed in a wide range. From all these results, we can see that for parallel scientific applications, even the original data is evenly decomposed and assigned to each process, the sizes of the compressed data on each process can be significantly different from each other due to the nonuniform distribution of the information density in the datasets.

III. Impact of Unbalanced Parallel I/O Loads

When parallel scientific codes write out or read in their data, the overall write or read performance usually depends on the slowest process. This is because a synchronization is often needed among processes before each computation step and no process can start the computation of the next step until the slowest process finishes its I/O. Therefore, scientists tend to decompose the data into chunks of the same size and evenly assign them to among processes, so that all the processes can finish the data writing or reading at about the same time to diminish the "straggler" effect. However, based on the observations we obtain in Section II, the amount of data each process

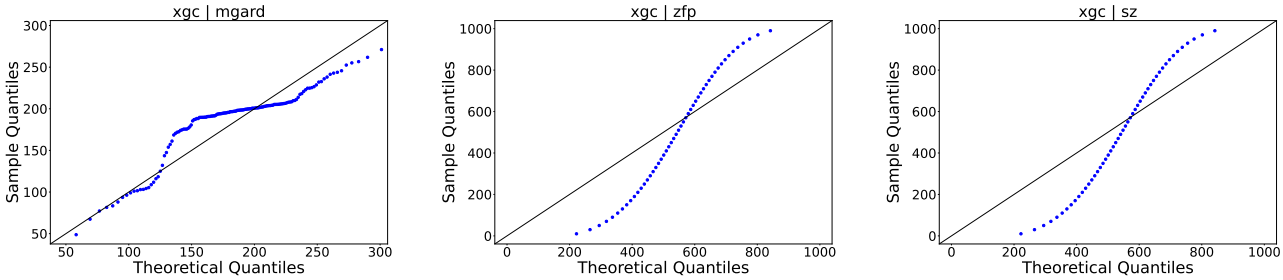

Fig. 3: Q-Q plot of using Weibull distribution to fit the XGC data with different lossy compressors.

TABLE I: Experiments for understanding the impact of unbalanced parallel I/O Loads.

Experiment	Description
uncompressed-equal	Each process writes the same amount of uncompressed data.
compressed-random	The sizes of the compressed data owned by each process are randomly generated based on certain probability distribution.
compressed-clustered	The per-process compressed data sizes are randomly generated based on the same probability distribution, but they are sorted and assigned to each process in ascending order so that the largest sizes are always assigned to a few processes running on the same compute node.
compressed-equal	The per-process compressed data sizes are randomly generated based on the same probability distribution, but the total size of the compressed data is equally divided by the number of processes and each process writes the same amount of compressed data.

writes out or reads in can be significantly different after lossy compression even if the original decomposed data is evenly distributed across processes, see Fig. 2. Therefore, applying lossy compression can potentially leads to imbalanced I/O loads across processes and thus cause "straggler" effect. In this section, we study how the imbalance of I/O loads caused by lossy compression affects the overall I/O performance of parallel scientific applications.

We conduct all our I/O tests on OLCF's Andes cluster. Andes is a 704-compute node commodity-type Linux cluster. Each of these compute nodes has 32 AMD EPYC 7302 cores and 256GB memory. Andes mounts the same center-wide GPFS file system as Summit, which offers more than 3GB/s per node I/O bandwidth. Since after decomposition the existing datasets we have are relatively small compared to the memory size on each compute node, we notice that the caching effect becomes a dominant factor when measuring the write performance. The measured performance numbers do not reflect the actual I/O throughput. In order to mitigate the caching effects and fully saturate the I/O bandwidth, instead of using the existing datasets, we developed a code which can synthetically generate data with arbitrarily large sizes for each process.

The impact of unbalanced parallel I/O might be different if the data is written in different patterns. There are three common patterns for writing data in parallel: N-N, N-1-collective, and N-1-independent. "N-N" denotes the code is executed by N processes and each process writes its data to a separate file independently. I/O libraries such as ADIOS2 adopts this pattern. "N-1-collective" denotes N processes write out their data to a single shared file using collective MPI-IO functions, while "N-1-independent" represents the data is written out to a single shared file using independent MPI-IO functions. These two patterns are used by I/O libraries such as HDF5, PnetCDF, etc. To understand which write pattern is more sensitive to the unbalanced parallel I/O, we measure the performance of writing data in all these three patterns in our experiments.

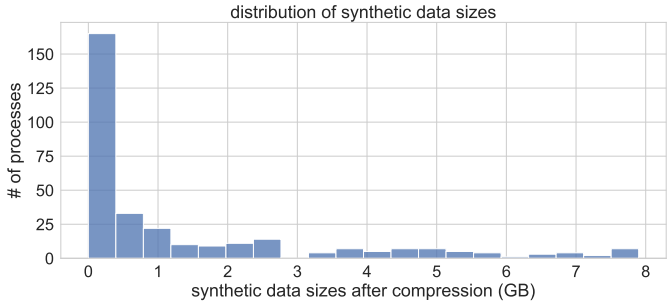

Fig. 4: Distribution of the synthetically generated data sizes among processes to mimic the effect of lossy compression.

TABLE II: Experiments for understanding the impact of unbalanced parallel I/O Loads.

Experiment	Description
uncompressed-equal	Each process writes the same amount of uncompressed data.
compressed-random	The sizes of the compressed data owned by each process are randomly generated based on certain probability distribution.
compressed-clustered	The per-process compressed data sizes are randomly generated based on the same probability distribution, but they are sorted and assigned to each process in ascending order so that the largest sizes are always assigned to a few processes running on the same compute node.
compressed-equal	The per-process compressed data sizes are randomly generated based on the same probability distribution, but the total size of the compressed data is equally divided by the number of processes and each process writes the same amount of compressed data.

First of all, we measure the overall performance of writing out the uncompressed data in different patterns as the baseline. In this experiment, we assume the uncompressed data on each process is 8GB. We lunch a job on Andes with 10 compute nodes and 32 processes per node, and let each process write out 8GB randomly generated data to the file system using different patterns. This experiment is denoted by "uncompressed-equal" in Table II.

Secondly, we randomly generate the sizes of compressed data owned by each process based on a Weibull distribution. The reason for choosing the Weibull distribution is that its probability density function usually has a long tail, which is similar to those shown in Fig. 2. To verify it, we use Weibull distribution to fit the real XGC data and show Q-Q plot in Figure 3. From the plot, we can see that most points perfectly lie on $y = x$, which suggests Wellbull distribution is a good representation. Since the size of the compressed data cannot be less than or equal to zero or greater than or equal to the original data size, we make sure only the valid random numbers are selected. Fig. 4 shows the sizes of the compressed data synthetically generated for 320 processes using a Weibull distribution whose shape parameter is 0.3 and scale parameter is 5.0. These sizes are randomly assigned to each process and each process writes out certain amount of data based on the assigned size. This experiment is denoted by "compressed-random" in Table II.

Thirdly, as shown in Fig. 1, the information density of each process' data demonstrates spacial locality. Data blocks that have similar information density are likely to be assigned to processes running on the same compute node or compute nodes that are close to each other in the HPC system's interconnect topology. E.g., in the bottom left corner of the second sub-figure in Fig. 1, data on those three processes all have dense information content. It is also possible that those three processes run on the same compute node as they need to share the I/O bandwidth of that particular compute node. If the sizes of the compressed data on those three processes are much larger than other processes, the I/O on that compute node would become a bottleneck. To mimic this scenario, we sort the sizes of the compressed data synthetically generated in the "uncompressed-random" experiment. We then assign them to each process in ascending order, so that the largest sizes are always assigned to a few processes running on the same compute node to trigger the bandwidth contention. This experiment is denoted by "compressed-clustered" in Table II.

Finally, we measure the write performance when the sizes of the compressed data on each process are all the same. Although this scenario rarely occurs in practice, we believe the performance numbers can be useful for us to understand how much write time the lossy compression can save ideally. In order to have a fair comparison with numbers measured in other experiments, we also use the sizes of the compressed data synthetically generated in the previous two experiments. We sum up all these synthetic sizes and calculate the mean of them. Then we let each process write out data with the size of that mean value. This experiment is denoted by "compressed-equal" in Table II.

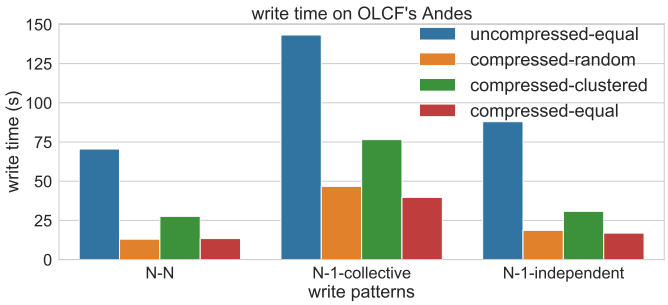

Fig. 5: The overall write time.

The overall write time measured in these four experiments are shown in Fig. 5. As we can see, adopting lossy compression does reduce the overall write time. The total size of the original data is $8 \times 32 \times 10 = 2560$GB, while the total size of the compressed data is 441GB. Ideally, if the sizes of the compressed data on each process are all the same ("compressed-equal"), we expect the write time is reduced by roughly 6 times. However, for other more realistic scenarios, the amount of write time reduced are less than the ideal case. As expected, "compressed-clustered" reduces the least amount write time. As we mentioned above, if processes running on the same compute node all have much larger data compared to other processes after lossy compression, I/O bandwidth contention on that compute node is likely to happen which makes the I/O on that particular node much slower than other nodes. Even those processes run on different compute nodes but those nodes are close to each other in the network topology, bandwidth contention might also happen on the routers shared by those nodes.

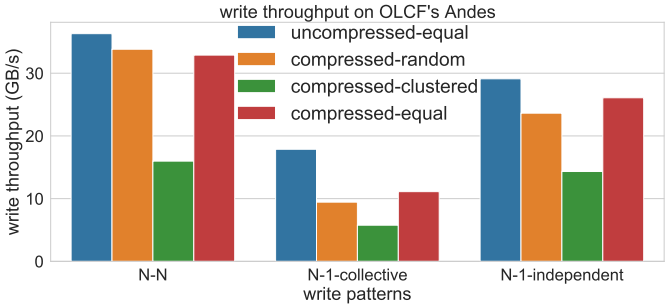

Fig. 6: The overall write throughput.

The write throughput for different scenarios are shown in Fig. 6. Apparently, "compressed-clustered" achieves the lowest write throughput no matter which write pattern is used. "compressed-random" achieves almost the same throughput as "compressed-equal" when each process writes its own data to a separate file. This is because in the "compressed-random" experiment, although the data sizes are very different across processes, the total data size of the 32 processes on each compute node does not show significant imbalance due to the random assignment of synthetic data sizes. If all the processes

write data to a single shared file, "compressed-equal" always outperforms "compressed-random". This indicates that writing data to a single shared file is more sensitive to the unbalanced parallel I/O loads.

IV. Related Work

As the advancement of computational power has greatly out paced capacity and bandwidth of I/O systems over the last decade, storing the whole scientific data has become infeasible as it will be prohibitory expensive. To reduce the cost of I/O and speed up scientific computations,using compression is a promising direction. Namely, scientific data are first reduced using lossless or lossy compressor before transferring through the I/O systems. Lossless compressors [15]–[19] offer the capability of compressing data and preserving bitwise identical information content in decompressed data. As scientific data become increasingly large with the advancement of scientific simulations and experiments, relative low compression ratios obtained though lossless compression can no longer satisfy both the time and resource constrains in modern scientific computing. As not every bit of the scientific data necessarily contributes to the useful information in data, lossy compression is known as a more favorable approach for greatly reducing the cost of I/O. Especially, to make sure important information contents are not lost during compression, several lossy compressors for scientific data have been proposed with guaranteed error control. For example, SZ [5] lossy compressor is built based on using multiple prediction methods. ZFP [4] lossy compressor is built based on block transformation. MGARD [2], [3], [6], [7] lossy compressor is built based on multilevel decomposition.

Based on compression techniques, many works has been focusing on applying compression to reduce the cost of I/O. For example, [20] proposes to use lossless compressors such as LZO and BZip2 to reduce the amount of data transferred over the network. They build I/O Forwarding Scalability Layer in the communication libraries so that the compression and decompression are transparent to users' applications. [21] proposes to use both lossless and lossy compressors to reduce the data movement cost between scientific simulation code and in-situ analytics. Based on their evaluation, they propose an adaptive compression service for the in-situ analytics middle-ware. [22] focuses on applying transparent compression between the computing and the storage systems. They build on-line decision system that can predict whether to compress at runtime, allowing guaranteed QoS for the I/O systems. [23] proposes to adaptively applying compression at highly-compressible regions and perform direct I/O on less-compressible regions to optimize the overall I/O performance.

As lossy compression are continuing to evolve for achieving higher compression ratios, it is anticipated that there will be higher disparity in terms of compressed sizes across regions in scientific data. This could lead to even more unblanaced I/O workload, which results inefficient I/O operations. However, very few existing studies have identified such issue or proposed solutions to improve the I/O performance.

V. Conclusion and Future Work

In this paper, we focus on an often neglected side effect of lossy scientific data compression: unbalanced parallel I/O. We conduct a comprehensive study by applying three commonly used lossy compressors MGARD, ZFP, and SZ to data generated by three real-world scientific applications Gray-Scott simulation, WarpX, and XGC. Our study quantifies the data skewness of lossy compressed scientific data across processes due to heterogeneous information density. Further experiments on write performance demonstrates how such data skewness can cause unbalanced parallel I/O and thus impact the parallel I/O performance.

In our future work, we aim to use information theory to formally analyze the information density across decomposed data and design an effective approach to estimate the data skewness among parallel processes so that proper actions can be taken to mitigate the parallel I/O imbalance issue. We also plan to employ a tiered approach to group decomposed data with complimentary information density on the same node to reduce the imbalance in parallel I/O.

VI. Acknowledge

This work was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and National Science Foundation grants CAREER-2048044 and IIS-1838024. This research used resources of the Oak Ridge Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. Furthermore, the research in this project was also supported by the SIRIUS-2 ASCR research project and the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL). We thank the anonymous reviewers for their insightful comments.

References

[1] C.-S. Chang and S. Ku, "Spontaneous rotation sources in a quiescent tokamak edge plasma," *Physics of Plasmas*, vol. 15, no. 6, p. 062510, 2008.

[2] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the univariate case," *Computing and Visualization in Science*, vol. 19, no. 5-6, pp. 65–76, 2018.

[3] ——, "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.

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

[5] 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.

[6] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2146–A2171, 2019.

[7] ——, "Multilevel techniques for compression and reduction of scientific data—the unstructured case," *SIAM Journal on Scientific Computing*, vol. 42, no. 2, pp. A1402–A1427, 2020.

[8] J.-L. Vay, A. Almgren, J. Bell, L. Ge, D. Grote, M. Hogan, O. Kononenko, R. Lehe, A. Myers, C. Ng *et al.*, "Warp-x: A new exascale computing platform for beam–plasma simulations," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 909, pp. 476–479, 2018.

[9] C.-S. Chang, S. Ku, P. Diamond, Z. Lin, S. Parker, T. Hahm, and N. Samatova, "Compressed ion temperature gradient turbulence in diverted tokamak edge," *Physics of Plasmas*, vol. 16, no. 5, p. 056108, 2009.

[10] OLCF, "Andes user guide," https://docs.olcf.ornl.gov/systems/andes_user_guide.html. Visited Sep 29, 2021.

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

[12] D. Salomon and G. Motta, *Handbook of Data Compression*. Springer, 2010.

[13] F. Cappello, S. Di, S. Li, X. Liang, A. Murat Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong, "Use cases of lossy compression for floating-point data in scientific data sets," *International Journal of High-Performance Computing Applications*, vol. 33, no. 6, 2018.

[14] W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck *et al.*, "Adios 2: The adaptable input output system. a framework for high-performance data management," *SoftwareX*, vol. 12, p. 100561, 2020.

[15] J.-l. Gailly and M. Adler, "Zlib compression library," 2004.

[16] Facebook, "Zstandard lossless compressor," http://facebook.github.io/zstd/. Visited Sep 29, 2021.

[17] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[18] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[19] M. Oberhumer, "Lzo-a real-time data compression library," *http://www. oberhumer. com/opensource/lzo/*, 2008.

[20] B. Welton, D. Kimpe, J. Cope, C. M. Patrick, K. Iskra, and R. Ross, "Improving i/o forwarding throughput with data compression," in *2011 IEEE International Conference on Cluster Computing*. IEEE, 2011, pp. 438–445.

[21] H. Zou, Y. Yu, W. Tang, and H. M. Chen, "Improving i/o performance with adaptive data compression for big data applications," in *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*. IEEE, 2014, pp. 1228–1237.

[22] R. Filgueira, M. Atkinson, Y. Tanimura, and I. Kojima, "Applying selectively parallel i/o compression to parallel storage systems," in *European Conference on Parallel Processing*. Springer, 2014, pp. 282–293.

[23] E. R. Schendel, S. V. Pendse, J. Jenkins, D. A. Boyuka, Z. Gong, S. Lakshminarasimhan, Q. Liu, H. Kolla, J. Chen, S. Klasky *et al.*, "Isobar hybrid compression-i/o interleaving for large-scale parallel i/o optimization," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, 2012, pp. 61–72.