Exploring Data Corruption Inside SZ

Ruiwen Shan and Jon C. Calhoun,
Holcombe Department of Electrical and Computer Engineering - Clemson University, Clemson, SC, USA
rshan@clemson.edu, jonccal@clemson.edu

***Abstract*—Due to the increasing scale of scientific research, scientists need to collect massive amounts of data to solve complex scientific problems. The exponential growth of data poses significant challenges to high-performance computing (HPC) systems in terms of their computational ability, storage capacity, and transmission bandwidth. Data reduction techniques such as data compression have become one of the most promising solutions to these problems. Error-bounded lossy compression is now commonly utilized in HPC systems to substantially reduce data volume while precisely maintaining data accuracy. However, the majority of research was done on improving compression efficiency, such as compression ratio, and insufficient attention is paid to the security of the compression process.**

In this paper, we concentrate on the impact of corruption on error-bounded lossy compressor SZ, including corruption due to transient failures of hardware and corruption injected by malicious users. We analyze and quantify the influence of this corruption on compressed datasets by simulating the corruption errors that occur in the regression coefficient values and computation during compression using four failure models. The results demonstrate that SZ's prediction-based design makes it sensitive to corruption of the regression coefficients. A single bit-flip in the regression coefficients can result in noticeable error propagation, in some cases, the compression ratio fluctuates up to 0.28%, but peak signal-to-noise ratio(PSNR) drops to negative levels.

***Index Terms*—High-performance computing, Lossy compression, Data security, Silent Data Corruption**

I. Introduction

During the past few decades, the difficulty of conducting scientific research solely through theory and experiment has gradually increased. It is common for scientists to utilize numerical algorithms to simulate the real world in order to solve challenging problems. High-performance computing (HPC) has become a crucial tool of scientific study due to its robust computational capabilities. As a vital component of science and technological innovation, HPC is widely used in many fields including weather forecasting, high-energy physics, life sciences, etc. However, several challenges have emerged with the widespread adoption of HPC systems in various fields. Scientific data generated by large-scale systems is primarily floating-point data and can be extremely large, sometimes exceeding petabytes in size. For example, the last Earth System Model for the Coupled Model Intercomparison Project Phase (CMIP) 5 collected model data from roughly 30 institutions and constructed a database of approximately 1.6 petabytes to complete the climate simulation [1]. The newer model CMIP6 is anticipated to generate 20 to 40 petabytes of data, mainly depending on the number of the resolution of the models [2]. Such a large dataset imposes a non-negligible burden on the system, such as insufficient storage space, limited I/O bandwidth, and more frequent silent data corruptions (SDC) [3]. In fact, it is almost impossible to store raw data directly because of restrictions associated with bandwidth and storage capacity [4]. One typical instance is the Linear Coherent Light Source (LCLS), a free-electron laser light source facility at SLAC National Accelerator Laboratory. The LCLS produces raw data at a rate of 250 GB/s and requires thousands of discs to sustain the data rate [5].

Data compression is a promising method of data reduction that can substantially alleviate issues related to I/O bottlenecks and limited storage space in HPC systems. Compared to lossless compression, error-bounded lossy compression (EBLC) is often preferred to manage large-scale data as it offers higher compression ratios(CRs) and accurate error control [6]. In fact, there is a common acceptance of information loss in practical applications. For example, when using the Particle-in-Cell (PIC) method to simulate particle acceleration in plasma, scientists use numerical plasma containing heavy macroparticles to reduce the amount of injected macro particles in order to lower the complexity of the problem [7]. The cutting-edge error-bounded lossy compressors, such as SZ [8]–[10] and ZFP [11], are capable of achieving CRs of 10–1000$\times$ when working with scientific datasets.

As EBLC data increasingly becomes common place, new issues arise with respect to its reliability and security. Prior work shows that lossy compressed data is vulnerable to transient data corruption, where a single bit-flip in a compressed data file can make it unreadable or induce error, rendering it unusable for scientific purposes [12]. While this study looked are corruption in the compressed file, the computation required to compress the data is still vulnerable [13], [14].

There are two main scenarios that may lead to corruption while running lossy compressors: the active case and the passive case. The *active* case refers to malicious attackers intentionally injecting faults into the compressor itself by modifying code or injecting corruption into actively used data buffers. In the *passive* case, silent errors occur as a result of hardware issues, such as unstable internal states resulting from a cosmic ray [13]. In either case, as long as the user does not perceive the presence of errors, the compressed dataset deviates silently. For a prediction-based lossy compressor such as SZ, the corruption is propagated as new values are predicted or recovered during decompression. In the most severe case, other researchers may use corrupted data for subsequent data analysis and interfere with critical studies.

Although several approaches have been developed for en-

suring data integrity for SZ compressors, such as block-wise error detection [14] and partial encryption during compression [15], these schemes limit the level of protection to key data structures and computations. Thus, the correctness of the compression parameters is neglected. Moreover, general purposes methods such as ARC do not protect compressed data while it is being generated [12].

To provide a better understanding of the sensitivity of data lossy compressed with SZ, we explore the impact of active and passive corruption in areas left unprotected by previous methods. Our main contributions are summarized as follows:

- We carefully analyze the workflow of SZ and set up two coefficient-centric failure scenarios during the compression process: system-level computation errors and code-level corruption injection.
- We implement four fault injection schemes to simulate coefficient-targeted bit-flip errors and malicious attacks then quantify the impact of these corruptions on compressed datasets. The results show that even a single bit-flip in regression coefficients can compromise the correctness of the entire dataset, allowing errors to dominate the dataset and rendering the decompressed data completely invalid for post hoc analysis.

The rest of the paper is organized as follows. In Section II, we analyze the cause and location of the corruption in SZ. Next, we present our fault inject methods and use cases in Section III. Section IV evaluates our fault models and quantifies the severity of the unprotected data regions. Finally, we discuss related work in Section V and conclude the paper in Section VI.

II. Analysis of SZ

A. SZ Lossy Compression

SZ is one of the most advanced error-bounded lossy compressors available in the field of scientific data compression today [8]–[10]. There are four major steps involved in SZ compression: 1) *Data prediction.* SZ divides the entire dataset into fixed-size blocks and then selects a more appropriate prediction function to predict future data values within each block in accordance with the results of the sampling method. We note that when using certain predictors, such as liner regression, additional data needs to be saved. In the case of linear regression, the regression coefficients are compressed separately. 2) *Linear-scale quantization.* With the user-specified error bound (eb), SZ quantifies the difference between the predicted value and the original data point as an integer (quantization index). 3) *Variable-length encoding.* Encodes the quantization indices using Huffman encoding. 4) *Lossless compression.* Enhance the compression ratio(CR) by using lossless compression.

B. Corruption Cause and Timing

The robustness of SZ against corruption is limited, as it is a prediction-based lossy compressor. In other words, if a deviation occurs in the prediction of one value, the propagation of this error in the subsequent compression process could lead to catastrophic results. Generally, these initial deviations are caused by two factors:

- **Malicious attack.** HPC systems process scientifically critical data as well as privacy information [16], which makes them vulnerable to malicious users. In this case, users with ulterior motives can massively alter or invalidate correct data by injecting corruption directly into the binary executable or running memory state, thereby yielding inaccurate results.
- **System malfunctions.** A typical example is bit-flip [13]. This type of error could have a direct impact on the logic of the program and the result of the running process if it occurs in an important place. In the case of SZ, a bit-flip in one value could cause deviations in the next prediction step, ultimately leading to a reduced CR, loss in PSNR, or errors beyond the range specified by the user. Because the chance of multi-bit errors is low even in HPC systems, we only consider single bit-flips [17].

SZ compresses the predictor coefficients in a separate pipeline to increase the CR. Specifically, SZ uses the adjacent coefficients of the same type to predict the current coefficients, quantizing the difference before applying Huffman encoding. Prior research develops strategies to detect and correct errors of critical data structures and computations in SZ but claims that errors in the coefficients have little impact on the CR and the accuracy of the compressor, as they only occupy a small portion of the overall memory and the same coefficients are used for compression and decompression [14]. However, our results show that the correctness of these parameters also merits concern, as their accuracy directly affects the overall lossy compressor performance. Consider the following two scenarios regarding the timing of errors:

- ***Timing 1:*** Corruption occurs while quantizing the coefficients. SZ verifies the decompressed data remains within a predictable range. In this case, if the decompressed data becomes unpredictable, the coefficients are saved, and an increasing amount of unpredictable data could degrade the compressibility of the entire dataset. However, if the data remain inside the predictable range, inaccurate quantization values leads SZ to use different coefficients for compression and decompression. As a result, the decompressed dataset is not valid.
- ***Timing 2:*** Corruption leads to deviation in the quantization index. In this case, corruption appear after the double-checking process has been conducted based on the correct quantization value. Theoretically, this type of fault does not significantly affect the CR, however, incorrect coefficients can cause serious deviations from the intended result during decompression.

The ramifications of corruption vary depending on when and where it occurs. The next section provides a detailed discussion of the significance of when and where.

III. Corruption Modeling and Injection Methods

The most common technique for determining how corruption impacts a specific application is fault injection [18].

The quantization process of SZ for regression coefficients is presented in Algorithm 1. For it, we design two scenarios for corruption injection based on a careful study of the compression process of SZ:

- ***Scenario 1:*** Red color in Algorithm 1. Flip one bit in a random regression coefficient. A random value is generated to determine which regression coefficient to select. This scenario represents the most common method of transient fault injection.
- ***Scenario 2:*** Orange color in Algorithm 1. Flip a particular bit of each coefficient to simulate injecting errors that modify code and transient errors in loop computation.

These two scenarios combined with the timing of the two error occurrences discussed in Section II-B yield four schemes for corruption injection:

1) ***S1+T1:*** The quantization index of a random coefficient is flipped by one bit during the quantization process. There is still a possibility of avoiding additional corruption after the bit is flipped, as the decompressed coefficient's value has not yet been calculated and categorized as predictable or not. In the case that the decompressed value exceeds the quantization range after the bit flip, the unpredictable coefficient is stored directly. This injection scheme has a negligible effect on the overall algorithm. Conversely, if the coefficient remains within the quantization range after the flip, the accuracy of the subsequent coefficients are compromised. Additionally, the location of the erroneous coefficient within the quantization array determines the extent to which it impacts the overall compression process (further ahead causes more corruption).
2) ***S1+T2:*** Flip one bit at a random coefficient before storing the correct quantization index. SZ has completed the prediction verification process at this point, therefore it cannot detect any changes to the index. Thus, the coefficient used during decompression differs from the one used when compressing. As a result, the value of the decompressed data points corresponding to this coefficient can become unusable.
3) ***S2+T1:*** The indexes of all regression coefficients suffer a single bit-flip during quantization. In this case, attackers manipulates the code to modify statement(s) in the for loop that processes the coefficients. Although some of the coefficients may be rectified (see S1+T1), the probability of correcting them all is negligible.
4) ***S2+T2:*** Flip one bit in all regression coefficients before storing quantization indexes. This scenario yields the worst outcomes. On the one hand, there is no follow-up check or protection mechanism to remedy this situation, which conceivably introduces substantial corruption with the compressed data file. On the other hand, the user may not be aware of this situation, resulting in the decompressed dataset misleading research directions or even interrupting the research process.

Algorithm 1: Quantization process of SZ3 [10].

```
Input: Regression coefficients coef, error-bound for coefficients
       eb_rc, predicted value from the adjacent coefficient p
Output: Quantization index array for regression coefficients
for c in coef do
    diff ← c − p;
    quant_bin ← diff/eb_rc;
    S1: Single bit-flip in quant_bin.
    if quant_bin < bin_capacity then
        Calculate the decompressed value D
        if |D − c| > eb_rc then
            S2: Single bit-flip in quant_bin.
            Save quant_bin;
        end
    else
        Save c as unpredictable;
    end
end
```

IV. Evaluation Results

A. Experimental setup

We conduct a series of experiments to comprehensively explore how corruptions in regression coefficients affect compressed datasets. We perform various experiments on Clemson University's Palmetto Cluster, where each node consists of two 2.4GHz Intel Xeon Gold 6148 processors and 376GB of RAM.

For our experiments, to ensure the results are applicable to datasets of various data types and features, we test on datasets from SDRBench [19]. In particular, we use *CLOUDf48* from the Hurricane Isabel simulation, *dark_matter_density* (to be called *Nyx* for simplicity) from Nyx [20], *T* from atmospheric modeling code SCALE-LetKF [21], and *diffusivity* (to be called *D*) from Miranda [22]. Table I details properties of each dataset.

TABLE I: Attributes of the datasets used in experiments.

Dataset	Type	Dimensions	Size	Description
CLOUDf48	float	100×500×500	95.37MB	Cloud moisture mixing ratio
Nyx	float	512×512×512	527MB	Dark matter density
T	float	98×1200×1200	61MB	Temperature
D	double	256×384×384	288MB	Rayleigh-Taylor simulation

We utilize SZ3 [23] with absolute error bound mode since it is the most recent version of SZ. In the configuration file, we select Lorenzo and Linear regression as prediction methods and implement all four error injection methods proposed in Section III. We compile all of our code with GCC-8.5.0.

Before presents results, we note one special case when injecting. A bit-flip in the most significant bit of the index value generates a negative value. The absolute value is used for all subsequent calculations as part of the original quantization process. Flipping this bit results in segmentation faults. We note that all runs flipping this bit resulted in a segmentation fault. A bit-flip in the most significant bit results in a negative quantization index, while SZ only considers positive indexes when compressing data. Consequently, in all of our experiments, we exclude injections into this bit position. Table II indicates the baseline CR and PSNR for each dataset during our experiment. All data presented in Section IV are the

outcomes of 1000 rounds of random corruption injection for each dataset.

TABLE II: Baseline CR and PSNR with eb=1E-3

Dataset	CR	PSNR
CLOUDf48	2824.4598	36.4495
Nyx	2.7233	147.9646
T	8.4107	107.2911
D	130.3103	79.6729

B. Method 1: S1+T1

Figure 1 summarizes the CR changes for our first method. Error bars indicate the standard deviation. Among the four datasets, *CLOUDf48* shows the most fluctuation in the compression ratio(CR). However, it still maintains a CR of greater than 99.99%. There are also some volatile trends observed in *D*'s CR, but it is considerably smaller than that of *CLOUDf48*. In general, *Nyx* and *T*'s CRs are stable at 99.998% and 100.005%, respectively, of their original values. There is a positive correlation between the original CR of the dataset and the degree to which it is affected after fault injection. That is, the higher the original CR, the more vulnerable it is to corruptions. According to the result in Table II, *CLOUDf48* has a high compressibility and, consequently, more regression coefficients. On the one hand, the large number of coefficients makes it more sensitive to the presence of corruptions in the coefficients. On the other hand, the location of the coefficients conduct bit-flips becomes one of the criteria that affect the CR. To *Nyx* and *T*, which bit of which coefficient is flipped has little impact in terms of the overall CR. For datasets where the majority of the data points are unpredictable, such as *Nyx*, a bit-flip in one coefficient can have a limited impact.

All of the datasets retain 99.99% of the corruption free baseline PSNR, which means a single bit-flip in the quantization of the regression coefficients, such as the occurrence of a computational error, does not cause a degradation in image quality. Additionally, compression against *T* over-preserves the user-set eb after fault injection. Overly strict eb generally causes a reduction in CR and results in a larger compressed dataset. However, in our case, the CR does not decrease significantly as the range of eb is narrowed by only 1E-5.

TABLE III: Method 1 PSNR and Max Absolute with eb=1E-3

Dataset	PSNR	Max Absolute error
CLOUDf48	36.4495 (100%)	1E-3
Nyx	147.9644 (99.99%)	1E-3
T	107.2910 (99.99%)	0.00099
D	79.6729 (100%)	1E-3

C. Method 2: S1+T2

In this case, we inject corruption in one random bit of a random coefficient. Figure 2 and Figure 3 show the result for CR and PSNR, respectively. The red line in each figure represents the original CR and PSNR.

We conclude that Method 2 has the greatest impact on *CLOUDf48*. However, even in the worst case, the CR is reduced by only 0.28%. *Nyx* shows the smallest change, with

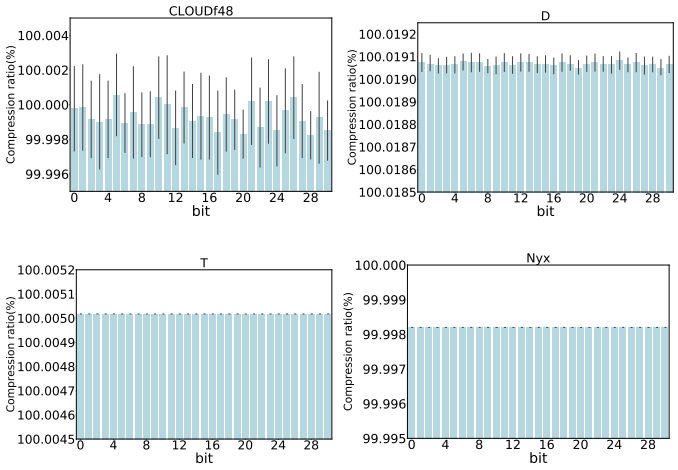

Fig. 1: Percentage of the original Compression Ratio for Method 1 compared to the corruption free baseline.

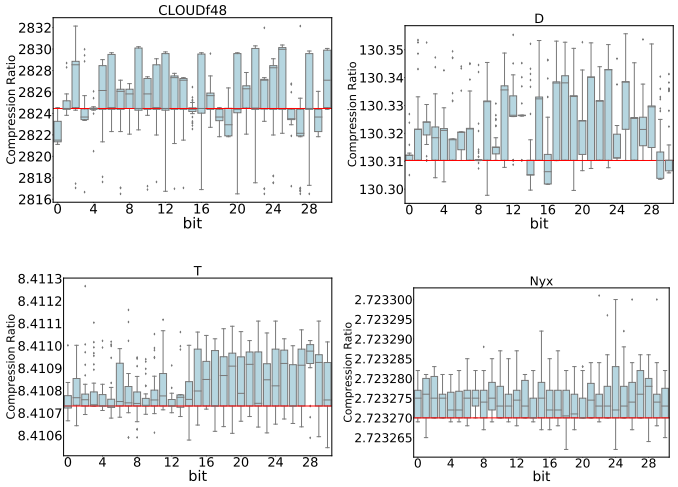

Fig. 2: Compression Ratio for Method 2. Red line indicates Baseline's compression ratio.

the CR only changing by 0.0011% in the worst case. This is consistent with the properties of the dataset, where *CLOUDf48* has a high compressibility and *Nyx* is hard to compress. In terms of PSNR, an corruption in the coefficients results in a PSNR lower than the original value. In general, the higher the bit where a flip occurs, the greater the adverse effect on PSNR. However, flipping which bit is not the only factor that influences the compression result. In this scheme, we also randomly choose the coefficients to perform bit-flipping, so the position of this coefficient in the overall coefficient array also needs to be taken into account. For dataset *T*, the PSNR value cannot be calculated after flipping the bit above the 16th bit. This is due to the introduction of an excessive number of errors to the coefficients and the floating-point exceptions yield invalid values such as NAN when calculating the offset between the decompressed data point and the original data.

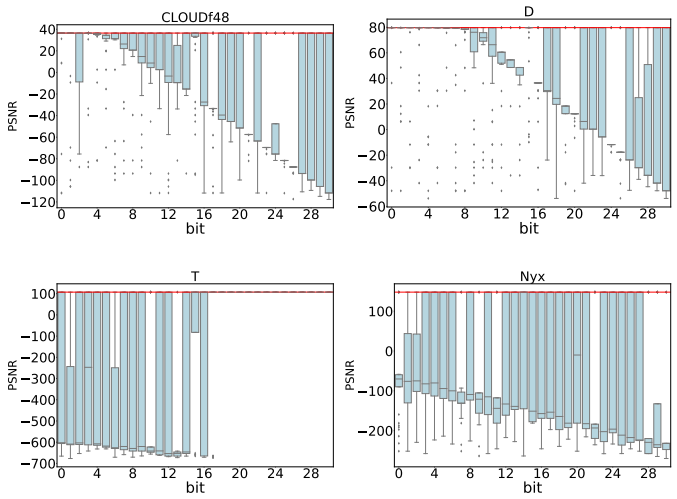

Fig. 3: PSNR for different datasets after applying method 2.

The results indicate that CRs fluctuate up to a maximum of 0.2% while the PSNR values decrease significantly, sometimes reaching negative levels. The user cannot determine whether the compression is successful based on the file size if an attacker decides to compromise the data by injecting one corruption. Only after decompression does the user realize that the dataset is dominated by errors and is no longer usable.

D. Method 3: S2+T1

Table IV shows the average metics obtained from 1000 corruption injection trials for each dataset. From Table IV, we observe that among the four datasets, only *Nyx* still adheres to the user-set error bound (eb=1E-3) after the coefficients are corrupted. *Nyx* respects the error bound because of its own properties. It is a dataset primarily composed of unpredictable data, which is stored losslessly. Thus, it originally has a low CR of 2.72×, which is far below the other three datasets. Less predictable data reduces the total number of regression coefficients as well as the likelihood that introducing too much error when the coefficient changes. Dataset *T* is over-preserved yielding a PSNR that is 0.294% better than the baseline. *CLOUDf48* performs better than *D*. However, it still decreases to below 30 dB, exhibiting poorer image quality.Changes in coefficients lead to a serious distortion in dataset *D*, which exceeds more than $10^5\times$ the original error bound, ruining the fidelity of the decompressed dataset. We use "-" to represent a negative PSNR here. It means that the decompressed dataset has too much noise and is not informative with respect to the original one. This result is consistent with the max absolute error for each decompressed dataset.

Investigating the impact on CR, *Nyx* and *T* retain more than 99% of the original CR, while the CR of *CLOUDf48* is severely damaged. The reason is that *CLOUDf48* contains over 99% predictable data when the error bound is 1E-3. Moreover, the efficiency of compressing coefficients is also important. Our experiments indicate that in this case, even if

TABLE IV: Compression Ratio and PSNR of Method 3 compared to the Baseline with eb=1E-3.

Dataset	CR(%)	PSNR(%)	Max Absolute error
CLOUDf48	6.655	76.167	0.0038
Nyx	99.041	99.998	1E-3
T	100	100.294	0.00099
D	52.492	–	120

only the least significant bit of all coefficients of *CLOUDf48* is corrupted, these coefficients enter the unpredictable branch during double-checking. It is also apparent from this result that the original coefficients of *CLOUDf48* are concentrated at the edge of bin_{max}, which allows for a high CR during the subsequent encoding and lossless compression process. As a result of Method 3, the compressor now has to save all regression coefficients as unpredictable, resulting in a precipitous drop in the CR.

E. Method 4: S2+T2

Table V shows the results for Method 4. While all datasets are successfully compressed, none of them maintain accuracy within the user-specified error-bound. Nonetheless, the CR of all datasets is generally preserved in this case. The reason is that we do not recalculate and correct the bin in which the decompressed data is located. Thus, it does not affect the coefficient quantization as no data that was predictable enters the unpredictable range. In summary, there is no noticeable change in the CR when utilizing this fault injection strategy. The tiny differences in the CR result from the subsequent processes (encoding and lossless) performed on the error-injected index. However, the size of regression coefficients is relatively small compared to the whole dataset and has only a minor impact on the overall CR.

TABLE V: Compression Ratio and PSNR of Method 4 compared to the Baseline with eb=1E-3.

Dataset	CR(%)	PSNR	Max Absolute error
CLOUDf48	100.023	-54.605494	3.8
Nyx	100.001	-206.188119	1.40E+17
T	99.99	-607.718888	8.60E+34
D	100	0.526726	19

From an attackers perspective, the compression is successful based on CR, but the data is useless. Scientists would never know their data is corrupted until they investigate. The results are consistent with our assumptions in Section III, as this time we inject the error after the double-checking mechanism, and there is no method in SZ for detecting or correcting this type of error. In this situation, the compressor incorrectly believes it is still processing the data correctly but in fact it has been tampered with. This is risky due to the large number of errors added to the data while decompressing, and the error magnitude greatly exceeded the user's tolerance. Besides, three datasets exhibit negative PSNRs, whereas *D*'s PSNR is very close to 0 indicating invalue decompressed data.

V. Related Work

With the popularity of HPC systems and the exponential growth of scientific data, many lossy compressors [8]–[11] have been developed over the past few years. Despite this, few studies have been conducted to examine how corruption occurs during lossy compression and its repercussions.

According to Li et al., [14], lossy compressor SZ does not exhibit resiliency to silent errors during data compression and transmission. They analyze the possible errors that could arise during each stage of SZ and the associated consequences, then design an independent block-wise model based on the SZ's framework as well as a set of strategies to detect and correct these errors. However, their paper only accounts for computational errors when targeting the regression coefficients. They state that since the coefficients used in compression and decompression are identical, computational errors do not affect the validity of the decompressed data. Besides, regression coefficients only occupy a small volume of memory, thus no additional precautions are necessary. In contrast, this paper concentrates on the corruptions of the valid regression coefficients, analyzing and qualifying the impact of employing different coefficients during the compression and decompression process, which is one of the cases that has not been covered in their paper.

LCFI [24] is an error injection tool designed to understand the error propagation in lossy compressors on HPC applications, this tool can inject a fault at any given time and location. Their experiments investigate the effect of precision loss on various HPC benchmarks by injecting faults into the program to simulate the errors induced by lossy compression. A series of Data-analytic Based Fault Tolerance methods (DBFT) based on end-to-end SDC detection methods are proposed by Li et al. [3]. The core idea of these approaches is to compare the predicted data and decompressed data(or snapshot data) to check whether the latter meets the user-set error bound. These detection methods are able to detect potential SDC occurring at each time step. ARC [12] protects compressed data only after it has been fully compressed which leaves the compression/decompression operations vulnerable.

VI. Conclusion

The massive volume of data in HPC systems poses problems of storage space and I/O bottlenecks. While error-bounded lossy compression is a reliable and efficient way to reduce the size of datasets, it is also vulnerable to faults and attack. In this paper, the relationship between the error in regression coefficients and the correctness of compressed datasets is investigated by several fault injection methods. The results demonstrate that even though the prediction coefficients occupy only a small fraction of the total data, once errors occur, the compressor cannot accurately control the introduced errors within user-specified error bounds, and metrics such as compression ratio can still be dramatically affected. Malevolent users can invalidate compressed datasets with a minor adjustment. Moreover, datasets can retain their compression ratio while the accuracy is fully compromised.

Acknowledgments

This research was supported by the U.S. National Science Foundation under Grants SHF-1910197 and SHF-1943114.

References

[1] M. Stockhause and M. Lautenschlager, "Cmip6 data citation of evolving data," *Data Science Journal*, vol. 16, 2017.

[2] V. Eyring, S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, and K. E. Taylor, "Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization," *Geoscientific Model Development*, vol. 9, no. 5, pp. 1937–1958, 2016.

[3] S. Li, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Towards end-to-end sdc detection for hpc applications equipped with lossy compression," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2020, pp. 326–336.

[4] F. Cappello, S. Di, S. Li, X. Liang, A. M. 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," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.

[5] G. Marcus, Y. Ding, P. Emma, Z. Huang, J. Qiang, T. Raubenheimer, M. Venturini, and L. Wang, "High fidelity start-to-end numerical particle simulations and performance studies for lcls-ii," in *Proceedings, 37th International Free Electron Laser Conference (FEL 2015): Daejeon, Korea, August 23-28*, 2015.

[6] J. Wang, T. Liu, Q. Liu, X. He, H. Luo, and W. He, "Compression ratio modeling and estimation across error bounds for lossy compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1621–1635, 2019.

[7] A. Pukhov, "Particle-in-cell codes for plasma-based particle acceleration," *arXiv preprint arXiv:1510.01071*, 2015.

[8] 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.

[9] 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.

[10] 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.

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

[12] D. Fulp, A. Poulos, R. Underwood, and J. C. Calhoun, "Arc: An automated approach to resiliency for lossy compressed data via error correcting codes," in *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC 21. New York, NY, USA: Association for Computing Machinery, 2021, p. 57–68. [Online]. Available: https://doi.org/10.1145/3431379.3460638

[13] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson *et al.*, "Addressing failures in exascale computing," *The International Journal of High Performance Computing Applications*, vol. 28, no. 2, pp. 129–173, 2014.

[14] S. Li, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Resilient error-bounded lossy compressor for data transfer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–14.

[15] R. Shan, S. Di, J. C. Calhoun, and F. Cappello, "Exploring lightweight cryptography for efficient and secure lossy data compression," in *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2022, pp. 23–34.

[16] S. Peisert, E. Dart, W. Barnett, E. Balas, J. Cuff, R. L. Grossman, A. Berman, A. Shankar, and B. Tierney, "The medical science dmz: a network design pattern for data-intensive medical science," *Journal of the American Medical Informatics Association*, vol. 25, no. 3, pp. 267–274, 2018.

[17] H. Schirmeier, C. Borchert, and O. Spinczyk, "Avoiding pitfalls in fault-injection based comparison of program susceptibility to soft errors," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 319–330.

[18] Z. Li, H. Menon, D. Maljovec, Y. Livnat, S. Liu, K. Mohror, P.-T. Bremer, and V. Pascucci, "Spotsdc: Revealing the silent data corruption propagation in high-performance computing systems," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 10, pp. 3938–3952, 2020.

[19] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "Sdrbench: Scientific data reduction benchmark for lossy compressors," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 2716–2724.

[20] L. et al., "methods: numerical, intergalactic medium, quasars: absorption lines, large-scale structure of universe," *journal of Monthly Notices of Royal Astronomical Society*.

[21] "Scalable computing for advanced library and environment-regional model," https://scale.riken.jp/scale-rm.

[22] W. H. Cabot and A. W. Cook, "Reynolds number effects on rayleigh–taylor instability with possible implications for type ia supernovae," *Nature Physics*, vol. 2, no. 8, pp. 562–568, 2006.

[23] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao *et al.*, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, 2022.

[24] B. Shan, A. Shamji, J. Tian, G. Li, and D. Tao, "Lcfi: A fault injection tool for studying lossy compression error propagation in hpc programs," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 2708–2715.