2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00116

# A Feature-Driven Fixed-Ratio Lossy Compression Framework for Real-World Scientific Datasets

Md Hasanur Rahman*, Sheng Di†, Kai Zhao‡, Robert Underwood†, Guanpeng Li*, Franck Cappello†
*Computer Science Department, University of Iowa, IA, USA
†Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
‡Computer Science Department, University of Alabama at Birmingham, Birmingham, AL, USA
mdhasanur-rahman@uiowa.edu, sdi1@anl.gov, kzhao@uab.edu,
runderwood@anl.gov, guanpeng-li@uiowa.edu, cappello@mcs.anl.gov

***Abstract*—Today's scientific applications and advanced instruments are producing extremely large volumes of data everyday, so that error-controlled lossy compression has become a critical technique to the scientific data storage and management. Existing lossy scientific data compressors, however, are designed mainly based on error-control driven mechanism, which cannot be efficiently applied in the fixed-ratio use-case, where a desired compression ratio needs to be reached because of the restricted data processing/management resources such as limited memory/storage capacity and network bandwidth. To address this gap, we propose a low-cost compressor-agnostic feature-driven fixed-ratio lossy compression framework (FXRZ). The key contributions are three-fold. (1) We perform an in-depth analysis of the correlation between diverse data features and compression ratios based on a wide range of application datasets, which is a fundamental work for our framework. (2) We propose a series of optimization strategies that can enable the framework to reach a fairly high accuracy in identifying the expected error configuration with very low computational cost. (3) We comprehensively evaluate our framework using 4 state-of-the-art error-controlled lossy compressors on 10 different snapshots and simulation configuration-based real-world scientific datasets from 4 different applications across different domains. Our experiment shows that FXRZ outperforms the state-of-the-art related work by 108×. The experiments with 4,096 cores on a supercomputer show a performance gain of 1.18∼8.71× than the related work in overall parallel data dumping.**

***Index Terms*—Lossy Compression, Scientific Data Management, Machine Learning, Data Features**

## I. Introduction

With ever-advancing scientific research, how to efficiently access and manage science data is critical to many today's scientific projects across different domains. In practice, scientific application users often use dedicated libraries or data formats such as HDF5 [1], ADIOS2 [2] and NetCDF [3] to store, manage and transfer the scientific data for efficient post-hoc analysis. These libraries or data formats are highly preferred by scientific users because of the high efficiency in multi-objective data query, management and storage/access. Multiple database systems or drivers (such as RESTful HDF5 [4] and HDF5 ODBC driver [5]) have also been developed to support such scientific data formats.

Corresponding author: Sheng Di, MCS Division, Argonne National Laboratory, 9700 Cass Avenue, Lemont, IL 60439, USA

Unlike the traditional data management, a grand challenge for the scientific data management is its vast volume of data to manage, store and transfer. In fact, extremely large volume of data are generated by today's scientific applications or instruments during their simulations or data acquisition. For example, Cosmological simulations such as Nyx [6] can produce hundreds of petabytes of data for each run. Such vast amount of generated data poses a significant challenge to store, process or transfer because of limited memory capacity, storage space, and network/I/O bandwidth [7], [8]. Thus, dramatically compressing such vast amount of data to better utilize available compute resources is a key focus at today's scientific data management.

Significant efforts have been put in resolving the big scientific data issue by data compression techniques. On the one hand, multiple libraries/toolkits (such as H5Z filter [9] and pNetCDF-SZ [10]) have been developed to enable the scientific data libraries/formats (HDF5, ADIOS2) to support transparent lossless and lossy compression for users. On the other hand, error-bounded data compressors [11]–[13] have been developed for scientific datasets in the past decade, as not only can it obtain high compression ratios but also it can provide high fidelity based on user-specified error bound.

To optimize data management, storage, or transfer performance, an efficient 'fixed-ratio compression mechanism' is critical in practice. Specifically, the systems or management tools often need to perform the data compression with the data size under control. For instance, users often need to archive, store, or transfer large amount of data locally or remotely [14], thus they need to control the compressed data size strictly according to the limited available resources such as memory capacity, I/O bandwidth and storage capacity [15], [16]. Existing error-bounded lossy compressors [12], [17]–[19] offer multiple types of error controls such as absolute error bound, relative error bound and peak signal-to-noise ratio (PSNR) to compress data, but unfortunately cannot compress data based on a target compression ratio required by users. FRaZ [20] is the first attempt working on the *generic* fixed-ratio lossy compression framework, but it suffers from very high computational cost (one order of magnitude or more) compared with the compression time, because of its expensive trial-and-error based iterative search method. Thus, it is

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00116

challenging to apply FRaZ in online in-situ use-cases.

In this paper, we propose a low-cost compressor-agnostic feature-driven fixed-ratio lossy compression framework, which faces a series of challenges to resolve. (1) For given datasets, we need to extract effective data features that can exploit diverse data characteristics such as data smoothness, data distribution, special pattern to project data compressibility. (2) Developing an efficient compressor-agnostic framework is *non-trivial* because there are many state-of-the-art error-controlled lossy compressors which exhibit largely different compression qualities with each other because of their distinct design principles. It is very challenging to develop a compressor-agnostic framework to characterize the correlation between various data features and diverse compressors' qualities. (3) State-of-the-art lossy compressors permit to input an error bound setting to obtain a compression ratio but not the other way around, and different lossy compressors demonstrate distinct compressibilities for the same dataset. Brute-force search through the exhaustive range of error bound settings would be very expensive, in order to obtain a compression ratio that is very close to the target compression ratio.

To address the above challenges, we propose an efficient compressor-agnostic **F**eature-driven fi**x**ed-**r**atio lossy **compression** framework (called **FXRZ**), which can determine the best-qualified error bound setting based on a user-specified target compression ratio, by efficiently extracting/analyzing data features at runtime. The key contributions are three-fold:

- To the best of our knowledge, *we are the first to propose a **low-cost feature-driven compressor-agnostic** lossy compression framework*, by leveraging the correlation between the data features and lossy compressor's quality to **efficiently** estimate the qualified error bound setting for the target compression ratio.
- We carefully investigate diverse data features and identify their effectiveness to the projection of compression ratios.
- We propose a series of optimization strategies ( e.g., *a novel technique* to adjust target compression ratio to project the data compressibility more accurately) that significantly improves the model accuracy based on target compression ratio with very little computational cost.

We evaluate our proposed FXRZ framework using variety of simulations and snapshot datasets from 4 real-world scientific applications across different domains. We perform the experiments based on multiple state-of-the-art lossy compressors such as SZ, ZFP, FPZIP, and MGARD+ and also compare our solution with another state-of-the-art related work – FRaZ [20]. Experiments show that our solution has a high accuracy (with only 8.24% estimation error on average) in estimating the required error bound setting based on a target compression ratio. FXRZ outperforms the FRaZ by 108× with the comparable accuracy. On a supercomputer, the overall parallel data dumping under FXRZ is substantially faster than that of FRaZ with a performance gain of 1.18∼8.71×.

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III, we propose and formulate our research problem. In Section IV, we detail our framework FXRZ. In Section V, we present the evaluation results with an in-depth analysis. Finally, we conclude and envision the future work in Section VI.

## II. Related Work

**Existing error-controlled lossy compressors.** There have been many error-controlled lossy compressors developed for scientific datasets, such as SZ [11], [18], [21], ZFP [12], FPZIP [19], MGARD+ [13]. Their designs are driven by the error-control model, so the fixed-ratio compression cannot be carried out directly or efficiently. The key reason is that lossy compressors often leverage some sophisticated entropy encoder or dictionary encoder such that the final compression ratios are very hard to estimate accurately. SZ, for instance, depends on Huffman encoding and dictionary encoding (Zstd [22]). To the best of our knowledge, ZFP is the only lossy compressor supporting fixed-ratio compression. However, ZFP's fixed-ratio mode (a.k.a., fixed-rate) suffers from much lower compression ratio (e.g., ∼2× lower compression ratio at the same data distortion level) compared with its fixed-accuracy mode, which has been validated by prior studies [20].

**Existing compression ratio estimation methods.** The fixed-ratio compression problem can be transformed to the compression ratio estimation in some sense, thus we also investigate the related works about lossy compression ratio estimation. In fact, if the compression ratio can be estimated efficiently based on any given error-control method (e.g., error bound), fixed-ratio compression can be realized by a search algorithm (e.g., binary search) with a searching cost to a certain extent. Lu et al. [23] explored how to estimate the compression ratio based on a given error bound for SZ and ZFP in particular, by leveraging these two compressors' characteristics obtained from their empirical studies. Tao et al. [24] developed a method to estimate the compression ratio for SZ and ZFP based on peak signal-to-noise ratio (PSNR), which is a more commonly used metric in the compression community. Liang et al. [25] proposed a hybrid lossy compression framework by integrating ZFP as one predictor in the SZ compression framework, which can improve the overall compression quality in turn. In their framework, one critical step is selecting the better data predictor (either SZ or ZFP) at runtime based on the estimated compression ratios for the two compressors. All the above compression ratio estimation methods, however, rely on the in-depth investigation of the specific lossy compressors' working principles. This is a serious drawback in that they cannot be adaptive to any new lossy compressor with emerging techniques.

**Existing generic fixed-ratio lossy compression framework.** The only related work in this category is FRaZ [20] – a generic fixed-ratio compression framework. FRaZ searches the best-fit configuration based on a given target compression ratio for any lossy compressor, however, it suffers from very high search cost (one order of magnitude or higher compared with compression time). The key reason is that it needs to run the lossy compressors on the full dataset iteratively in order to get a high estimation accuracy. As such, FRaZ is only

suitable for the offline analysis but not for the real-time usage. By comparison, our solution FXRZ is a low-cost fixed-ratio lossy compression framework, which can obtain the required configuration without running the lossy compressor at all. So, it can be applied to real-time usecases, such as fixing ratio for data transfer at runtime.

## III. Problem Formulation and Use-cases

### A. Problem Formulation

Recall that any error-controlled lossy compressor is driven by a given error bound/configuration setting but not the other way around – cannot be executed based on a given compression ratio. Our framework FXRZ aims to fill this gap. Given a multi-dimensional scientific dataset *D*, an error-controlled lossy compressor *C* (SZ, ZFP, FPZIP or MGARD+), and a target compression ratio (denoted *TCR*), our framework FXRZ extracts the key data features from the dataset *D* and estimates an error bound setting *Ep*, under which the measured compression ratio (denoted *MCR*) would be close enough to the target compression ratio (*TCR*). Finally, we compare *MCR* with *TCR* to verify the accuracy of FXRZ. Therefore, for a given *D*, a *C* and a *TCR*, our research objective is to minimize the difference between *MCR* and *TCR* – that is $\min(|MCR\text{-}TCR|)$ – with very low performance overhead.

Such a fixed-ratio feature-driven compression framework can benefit many scientific users in practice. In fact, each scientific application package nowadays (such as Nyx [26], QMCPack [27], RTM [28]) is generally serving many users worldwide. That is, the training triggered by one user is expected to benefit many other users in the similar domain.

### B. Discussion of Use-Cases

Scientific data management is composed of multiple complicated phases in the entire data acquisition and analysis, also depending on user's diverse use-cases. In what follows, we describe several real-world use-cases regarding our FXRZ.

- *Preserving best data quality based on restricted data transfer bandwidth*. In the materials science, the advanced instruments such as LCLS-II [29] and APS-U [30] generate an extremely large volume of ptychography data (up to 250 GB/s in the raw data acquisition [31]), and these data need to be transferred to the data server through a relatively low-bandwidth I/O or network. Accordingly, the materials scientists have specific requirements on the minimum compression ratios (generally 10+ [31]) for the qualified lossy compression methods.
- *Preserving best data quality based on limited storage space*. Many scientific applications such as cosmology research [26], [32] may produce vast amount of data (from hundreds of TBs to dozens of PBs for each run) during the simulations. However, supercomputing users always have limited storage spaces on a supercomputer (e.g., 10TB for a regular user from ANL Theta [33] and 50TB for a regular user from ORNL Summit [15], [34]), so that they have to perform a lossy compression with a minimum compression ratio to store such a large amount of data in practice.
- *Preserving best data quality based on limited memory capacity*. Our proposed low-cost feature-driven fixed-ratio framework can also benefit the in-memory data processing at runtime. Quite a few scientific applications require a fairly large memory capacity to deal with large problem size. Quantum computing simulations, for example, may require up to 32EB of memory [35] when running a simulation with 61 qubits on a supercomputer. Considering the restricted memory capacity, the users have to compress the data in memory and reconstruct them when needed during the simulation, in order to avoid the out-of-memory crash.

## IV. Feature-driven Fixed-ratio Error-controlled Lossy Compression Framework (FXRZ)

In this section, we describe FXRZ (available at https://github.com/hasanur-rahman/FXRZ). We first present the design overview and then describe how we resolve a series of challenges and optimize the performance.

### A. Design Overview

The fundamental design idea is to explore the key data features from the real-world scientific datasets and apply effective augmentation technique to quickly generate ample compression results (without running the compressor) and then adopt an ML model to estimate the best-qualified error configuration for a target compression ratio (at runtime).

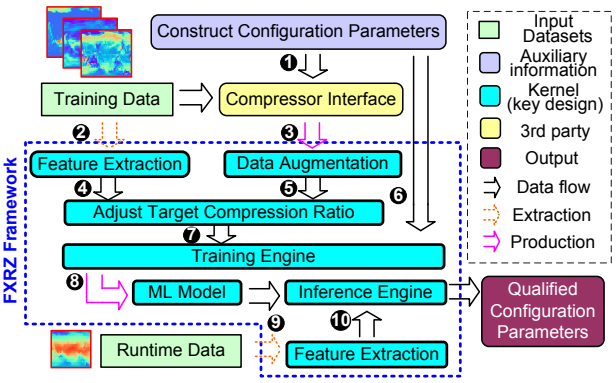



Fig. 1: Design Architecture of Our Framework (FXRZ)

The FXRZ is composed of several key modules, as shown in Fig. 1. The feature extraction module is developed for extracting the feature vector, which is used to construct the ML model during the training stage (❷) and also to decide the best-qualified error bound setting during the inference stage (❾). The data augmentation step is to augment the compression results which are generated by running different lossy compressors (❶ and ❸). Optimization such as adjustment in target compression ratio is applied in ❹. The training engine is the kernel module that takes over the ML training work based on the three parts of information: extracted feature vector (❹), augmented compression results (❺) and

corresponding configuration parameters (❻), and optimizes the error bound estimation based on our dynamic target ratio adjustment method (❼). After generating the well-trained ML model (❽), the inference engine will be launched to decide the best-qualified error configuration setting to reach the target compression ratio in terms of the runtime-extracted feature vector (❿).

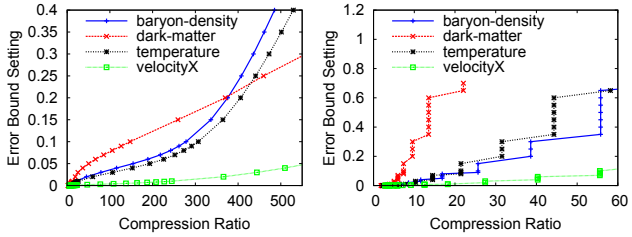
We propose two levels to assess the capability of FXRZ, based on training datasets versus runtime datasets.

- **Capability Level 1**: *Accurate decision across different time steps based on the same application model with the same simulation configuration*. In this situation, we focus on the same application model with unchanged simulation configuration. Specifically, the users train the model using prior set of snapshot data and their corresponding lossy compression results, and then use the trained model to make decisions for latter snapshot data with different time steps. A typical example is training the model using the snapshots 1 - 30 from Hurricane Isabel simulation [36] and test the decision accuracy based on the latter snapshots such as 48.
- **Capability Level 2**: *Accurate decision across different simulation configurations within the application model*. Level 2 is very practical, in that every application model or package (such as QMCPack [27], Nyx [26]) have multiple users with different research purposes. In general, these users are using the same application package/model but running the simulation with distinct configurations. In this situation, FXRZ is expected to make accurate decisions for a user given dataset using the model trained by the datasets generated by other users.

### B. Data Augmentation based on Interpolation

We know that a ML model requires a fair amount of samples in the training. Whereas, generating a large number of compression result samples by running lossy compressors is very expensive, because the training stage requires as many different datasets and different error bound settings as possible, which would inevitably result in numerous compression operations to execute.

To resolve this issue, we propose a data augmentation method that can effectively expand the limited number of compression results to reach the expected amount of training samples for FXRZ. *Our main observation is that the compression ratios within similar error bound range are very close with each other for a lossy compressor*. As such, we augment the compression result samples by leveraging linear interpolation (or least-square method). Specifically, we first run the specified compressor for a couple of representative error bound settings to generate a certain number of compression ratios. These results will be treated as *stationary points*, based on which we can produce the more compression ratio results by the linear interpolation method. Such an augmented compression result curve forms an error bound setting function of compression ratio, so that an expected error bound can be interpolated for any given compression ratio (on the curve) in the training stage. We further illustrate our




(a) Interpolation for SZ (b) Interpolation for ZFP

Fig. 2: Illustrating Linear Interpolation of Compression Results based on Nyx Simulation Datasets

idea by Fig. 2. To save space, we give only two examples from two representative compressors, SZ and ZFP, for Nyx Baryon density field dataset. The datasets and compressors will be discussed in details in Section V-A2 and V-A3. The points on the curves are the stationary points and the curves represent the interpolated curves. Stationary points (measured compression ratio, error bound) are generated by running the respective compressor at uniformly spanned, on average, 25 different error bound settings. As aligned with our observation, Fig. 2 shows us that compression ratios are close between two consecutive stationary points where corresponding error bounds are close. Hence, by leveraging the interpolated curve, we can estimate the error bound setting for any given compression ratio in the range of stationary points. For example, with Baryon Density, we get the error bound of $\sim$0.1 for compression ratio 270 on SZ, and the error bound of $\sim$0.2 for the compression ratio 33 on ZFP. It is worth noting that ZFP's interpolated result follows a stairwise curve, since ZFP's compression ratio increases piecewisely with the error bound because of its coefficient bitplane truncation. Although the relationship between error bound and compression ratio is not linear [20], we find the relationship to be approximately linear between two consecutive stationary points. In fact, the average percentage difference between measured compression ratios from interpolated error bounds (Y-axis) and given compression ratios (X-axis) is only 3.04%, 3.96%, 5.48%, 4.34% for the four respective compressors SZ, ZFP, FPZIP, MGARD+ across all applications.

### C. Features Extraction

Without effective feature characteristics, FXRZ may suffer from very poor accuracy. We examine eight different features for our analysis. We describe these features and their varying impact on data compressibility as follows.

$$lorenzo_{i,j} = d_{i-1,j} + d_{i,j-1} - d_{i-1,j-1} \tag{1}$$

$$\begin{aligned} lorenzo_{i,j,k} = {} & d_{i-1,j,k} + d_{i,j-1,k} + d_{i,j,k-1} - d_{i-1,j-1,k} \\ & - d_{i,j-1,k-1} - d_{i-1,j,k-1} + d_{i-1,j-1,k-1} \end{aligned} \tag{2}$$

$$spline_i = -\frac{1}{16}d_{i-3} + \frac{9}{16}d_{i-1} + \frac{9}{16}d_{i+1} - \frac{1}{16}d_{i+3} \tag{3}$$

- **Value Range:** It refers to the value range of a dataset, indicating how much the data deviates in the dataset (or the amplitude of the dataset).
- **Mean Value:** Mean value refers to the average of all the data points in a dataset.
- **Mean Neighbor Difference (MND):** Mean neighbor difference is obtained by first calculating the absolute difference of current data value and the average of its neighbor values, and then computing the mean of all the absolute differences.
- **Mean Lorenzo Difference (MLD):** MLD is denoted by the average of absolute difference between data value and its Lorenzo prediction. Equation (1) and (2) demonstrate how the Lorenzo prediction is performed for a data point (*(i,j)* or *(i,j,k)*) in a 2D dataset and 3D dataset, respectively.
- **Mean Spline Difference (MSD):** The MSD feature is calculated as the mean of a specific cubic spline-interpolation fitting error for all data points in the dataset. Equation (3) demonstrates how the cubic spline-interpolation fitting is calculated for a data point in a 1D dataset. Similarly, for a multi-dimensional dataset, Equation (3) is used to calculate $spline_i$ along each dimension separately, and obtain the average value $A$ across all dimensions. Finally, we consider the difference between current data value and $A$ as *MSD* value for the current data point.
- **Other features:** We also explore other features including *Mean Gradient*, *Min Gradient*, *Max Gradient*. Gradient denotes the difference between current data value and its previous data value. But gradient-based features often can not capture smoothness of data values in a dataset. We further give the reason and establish this claim when we discuss the Table II.

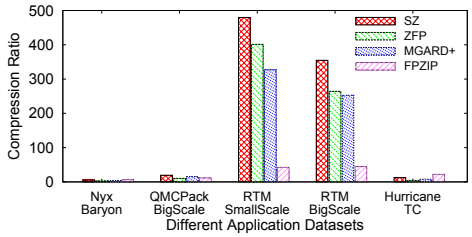



Fig. 3: Compression Ratios across Different Datasets and Compressors under an error bound.

TABLE I: Feature Values across Different Datasets

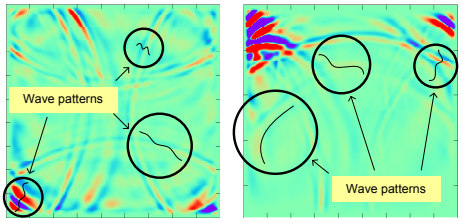
| Feature | Nyx Baryon Density | QMCPack BigScale | RTM SmallScale | RTM BigScale | Hurricane TC |
|---|---|---|---|---|---|
| Value Range | 4.90 | 35.36 | 0.16 | 0.05 | 104.81 |
| Mean Value | 0.97 | 16.75 | 0.09 | 0.02 | 45.63 |
| MND | 0.01 | 0.29 | 1.1E-4 | 5.5E-5 | 0.67 |
| MLD | 0.31 | 0.30 | 9.2E-5 | 4.0E-5 | 31.30 |
| MSD | 8.4E-3 | 0.33 | 1.3E-4 | 6.1E-5 | 0.79 |

*Investigation of Relationships between Data Features and Compressibility:* We now explain how we develop these data features in terms of their characteristics, by analyzing the relationships between these features and the compressibility across different datasets. We refer to Fig. 3 and Table I to explain the relationship under a fixed error bound *e*. We show five datasets from four applications because of space limit. Other datasets show similar characteristics. In Fig. 3, x-axis denotes the dataset, and y-axis shows corresponding compression ratios under *e* with different compressors. Table I shows the feature values across those datasets.

*Value Range* and *Mean Value* reveal the amplitude and spreadness of data values in a dataset. By comparing Fig. 3 and Table I, RTM datasets have smaller *Value Range* (0.16 and 0.05) than other datasets have but show much higher compression ratios for different compressors. Smaller *Value Range* indicates that the data values in a dataset tend to be close to each other, which makes it easier for the compressors to compress. But only *Value Range* cannot always reflect the above relationship as we can see by comparing the Nyx and Hurricane datasets' *Value Range* with compression ratios. More specifically, although *Value Range* of Nyx is lower than that of Hurricane, Nyx's compression ratios are still lower than those of Hurricane, which can be reasoned about with the *Mean Value* feature. *Mean Value* of Hurricane dataset follows its *Value Range* more closely than Nyx does: the ratio of *Value Range* to *Mean Value* for Hurricane is lower than that of Nyx. This implies that data values in Hurricane dataset are actually closer than those in Nyx. Hence, combined effect of *Value Range* and *Mean Value* reflects true data spreadness.

Moreover, *MND* and *MLD* reveal the spatial data smoothness in a dataset. The less value of *MND* and *MLD* indicates higher data smoothness, hence higher compression ratio because of ease of compression by compressors. By comparing Fig. 3 and Table I, the less *MND* and *MLD* values are, the more compressible the datasets are. As *MND* is based on only neighbor data values, it can reflect the local data smoothness well. In contrast, *MLD* is based on large regions of data values, hence it can reflect the overall data smoothness in a dataset. Hence, both features can complement each other well. Finally, *MSD* feature is particularly effective in detecting the wave textures/patterns, which are very common in many scientific datasets [37], e.g., RTM, Hurricane, QMCPack datasets. In Fig. 4, we show an example of such textures present in RTM dataset. By comparing Fig. 3 and Table I, RTM datasets show less MSD values than others. Less MSD values reflect more smooth wave textures, hence RTM datasets show relatively higher compression ratios across different compressors.

We now quantify the relationship between the features and data compressibility. We first obtain compression ratios of different snapshots or simulation configurations of a dataset based on a particular compressor with the same error bound setting. We then calculate different features for each of those datasets with different snapshots/configurations. Then, for each compressor, we calculate the average Pearson Product-Moment Correlation Coefficients [38] for different datasets




(a) Cross Section Visualization (b) Longitudinal Section Visualization

Fig. 4: Wave Texture in RTM simulation Data

across different error bounds in order to investigate the correlation between the features and data compressibility. We apply different error bounds on datasets, which can obtain a high diversity of the compressibility, such that the correlation analysis would be accurate and effective. We show such correlations in Table II. As shown in the Table, *Value Range*, *Mean Value*, *MND*, *MLD*, *MSD* are the most correlated features. Hence, we adopt these five features for FXRZ. Also, we can see that gradient-based features exhibit the least correlation. On the one hand, the *Max Gradient* is too sensitive to the data changes in space. On the other hand, *Min Gradient* and *Mean Gradient* are too mild to indicate the data change because the scientific data are quite smooth in most of regions in general (as shown in Fig. 4). As such, the gradient-based features are excluded.

TABLE II: Average Correlation Coefficent between Corresponding Feature and Compression Ratio across Different Datasets for Different Compressors

| Comp. | Value Range | Avg. Value | MND | MLD | MSD | Mean-Gradient | Min-Gradient | Max-Gradient |
|---|---|---|---|---|---|---|---|---|
| SZ | 0.73 | 0.71 | 0.69 | 0.64 | 0.70 | 0.54 | 0.54 | 0.47 |
| ZFP | 0.65 | 0.61 | 0.64 | 0.64 | 0.65 | 0.47 | 0.54 | 0.50 |
| FPZIP | 0.60 | 0.67 | 0.77 | 0.80 | 0.75 | 0.46 | 0.39 | 0.20 |
| MGARD+ | 0.62 | 0.64 | 0.70 | 0.60 | 0.70 | 0.43 | 0.31 | 0.47 |

### D. ML Model Selection

For any dataset, we utilize 5 extracted features and a target compression ratio as input for the ML model. During inference phase, the model predicts expected error bound setting.

The Classifier models are not suitable for our framework, because classifiers generally focus on discrete states or classes of the results while error control configurations (FXRZ outputs) are often continuous values in practice. For instance, error bound values could be any floating-point number.

TABLE III: Average Estimation Error Based on Target Compression Ratios with RFR, AdaBoost and SVR model

| Comp. | Nyx Velocity-X | | | QMCPack BigScale Spin0 | | | RTM BigScale Snapshot-800 | | |
|---|---|---|---|---|---|---|---|---|---|
| | **RFR** | AdaBoost | SVR | **RFR** | AdaBoost | SVR | **RFR** | AdaBoost | SVR |
| SZ | 10.17% | 31.59% | 111.17% | 1.52% | 71.21% | 82.89% | 13.88% | 28.79% | 97.46% |
| ZFP | 4.68% | 20.18% | 74.12% | 2.59% | 40.57% | 40.99% | 3.71% | 41.61% | 125.13% |

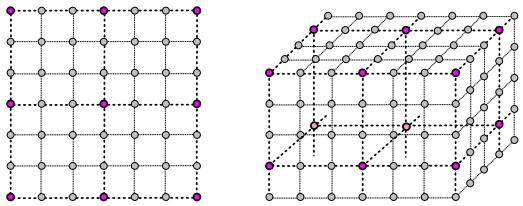
We select three popular ML models to apply on our framework FXRZ. For all of them, we use k-fold cross validation to tune the hyperparameters and improve the performance. We show the average estimation errors (defined in Formula 5) for ML models in Table III. 1) *Support Vector Regressor (SVR)* [39] is a counterpart of *Support Vector Machine (SVM)* and accepts non-linearity in the data for regression analysis. But we find *SVR* is not a good fit for our problem setting, in terms of the average estimation errors as shown in Table III. According to the table, based on three example datasats with two representative compressors (SZ and ZFP), the SVR suffers from very high estimation errors than the other two. The key reason is that the bestfit error configurations are not sometimes sufficiently separable enough to build hyperplanes to differentiate distinct compression ratio results. 2) *AdaBoost Regressor* [40] is a meta-estimation technique. As shown in Table III, AdaBoost suffers from relatively higher average estimation error. We find that *AdaBoost* suffers from high estimation errors when target compression ratios (and their corresponding expected error configurations) are relatively lower. The possible reason could be those lower error configurations in training data are very close to each other and tiny changes in expected error bound settings from the lower range might not be captured well by AdaBoost regression. So, this is also not a good fit for our problem setting. 3) *Random Forest Regressor (RFR)* [41] is a good fit to our problem setting because it has the special ability to correct overfitting problem by building lots of trees. As shown in Table III, *RFR*'s average estimation error is lowest among three. Hence, we adopt *RFR* for the analysis in FXRZ, in terms of data features and augmented lossy compression results.

### E. Optimization of Performance and Accuracy

In this section, we describe our optimization strategies which aim to further boost the execution performance and also improve the model accuracy.

*1) Uniform Sampling for Feature Extraction:* In order to avoid scanning the full dataset to calculate the feature, we calculate the features based on uniformly stride-$K$ sampled data points (shown in Fig. 5), which are selected every $K$ data points along each direction. In our experiments, the sampled data take only 1.50% of total data points (stride=4 in the sampling), which can still obtain a high accuracy for our solution (shown in Sec V-F). On average, across all datasets and compressors, FXRZ with 1.5% sampling, and considering all data points (100% sampling) yield average estimation errors of 8.24%, and 6.23% respectively. As we see, 1.5% sampling yield similar estimation error compared to 100% sampling but make FXRZ much faster because of less amount of sampling. In fact, such a sampling method (1.5%) makes the analysis time take only $\frac{1}{50}\times$ of the analysis time when using all data points, which significantly speeds up the overall analysis of our framework FXRZ.

*2) Adjusting Target Compression Ratio for Better Accuracy:* We observe that overall compression ratio of a dataset is often very sensitive to the area of the smooth region in space. These smooth regions are generally very easy to compress with extremely high compression ratios, hence they contribute to overestimation of the true compressibility of a dataset by not revealing accurate data density in the dataset. To achieve



(a) 2D uniform sampling (b) 3D uniform sampling

Fig. 5: 3-point Stripe Uniform Sampling

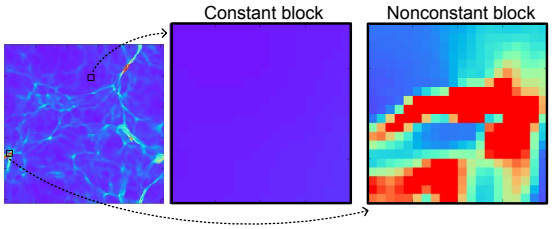



Fig. 6: Illustration of Constant/Non-constant Blocks

that, we adjust the target compression ratio by excluding the smooth regions without loss of generality. To this end, we introduce a *novel optimization strategy* that can improve the FXRZ accuracy in estimating error configuration, by adjusting the target compression ratio based on the density of the data, which we call *Compressibility Adjustment (CA)*. Specifically, For that, we split the whole dataset into many small blocks (e.g., 4×4×4 for 3D datasets in our experiment). If a block has very small deviation (its value range is lower than a threshold), we call it a *constant block*; otherwise, it is a *non-constant block*. How to determine the value range threshold will be discussed later. Fig. 6 illustrates the constant blocks and non-constant blocks using an example dataset (Nyx Temperature of size 512×512×512).

We explain the calculation of the adjusted compression ratio as follows. To avoid the over-adjustment, we need to make sure the data values within each constant block are fairly close with each other (i.e., the value range threshold is relatively small). Consequently, their output data size is assumed to be 0 after compression without loss of generality. In other words, the compressibility of a dataset is only determined by the non-constant blocks. As such, the adjusted compression ratio is calculated as Formula (4).

$$ACR = TCR \times R \tag{4}$$

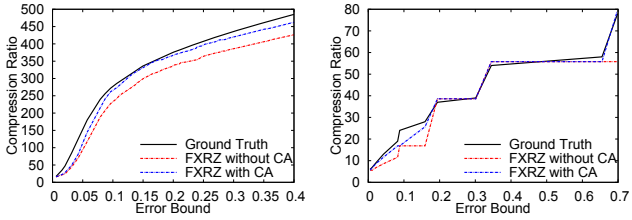
where *TCR* refers to the user-specified target compression ratio, *ACR* denotes the adjusted compression ratio, and *R* is the percentage of the non-constant blocks in given dataset. Accordingly, we convert the input *TCR* to *ACR* before feeding into the ML model. Using *ACR* helps obtain much higher accuracy, to be validated in Section V-E.

To measure *R*, we need to find a proper value range threshold to distinguish constant and non-constant blocks. As *mean value* indicates spreadness of data values, it can be used to determine the threshold. Thus, we use a coefficient, $\lambda$, of *mean value* to obtain the threshold. As shown in Table IV, 15% of *mean value* ($\lambda$=0.15) as the threshold is the optimal setting to distinguish blocks. With the information of total number of these two types of blocks, we finally measure $R$.

TABLE IV: Average Estimation Error by $\lambda$ of 0.05, 0.10, 0.15

| Compressor | Nyx Baryon Density | | | QMCPack BigScale Spin0 | | | RTM BigScale Snapshot-800 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0.05 | 0.10 | **0.15** | 0.05 | 0.10 | **0.15** | 0.05 | 0.10 | **0.15** |
| SZ | 20.31% | 16.76% | 7.63% | 2.02% | 4.58% | 1.52% | 27.37% | 15.46% | 13.79% |
| ZFP | 21.35% | 22.14% | 6.95% | 6.56% | 6.11% | 2.59% | 9.06% | 15.63% | 3.17% |

Now, we verify the effectiveness of our *CA* design using Fig. 7 based on SZ and ZFP, by comparing the accuracy of FXRZ between with and without *CA* for Nyx Baryon Density. We describe Nyx (Baryon Density) in Section V-A2. The black curve of each sub-figure refers to the (*TCR*). The red curve denotes *MCR without CA* design, whereas the blue curve denotes *MCR with CA* design. Recall that the closer the *MCR* to the *TCR* is, the more accurate our FXRZ is (see Section III-A). Fig. 7 shows that blue curve is very close to or even overlaps with the black curve sometimes, whereas the red curve is often distant from the black curve. This figure clearly shows that our novel CA strategy is very useful to the improvement of FXRZ accuracy.




(a) NYX Baryon Density Result (SZ) (b) NYX Baryon Density Result (ZFP)

Fig. 7: *CA* Optimization for Better Accuracy

## V. Performance Evaluation

In this section, we describe our experimental setup and present the performance evaluation results.

### A. Experimental Settings

We describe the environment setting, datasets, testing compressors and related works to be evaluated as follows.

*1) Environment:* We perform our experiments on the Intel Broadwell nodes (Intel Xeon E5-2695v4) of ANL Bebop [42], which is a supercomputer managed by Laboratory Computing Research Centers (LCRC) at Argonne. Bebop is featured with 1000+ nodes connected with Omni-Path Fabric Interconnect network. Each Intel Broadwell node has up to 128GB DDR4 and 36 cores. Its storage system is using General Parallel File Systems (GPFS), which is equipped with two I/O nodes, offering ∼2GB/s I/O bandwidth. This Bebop machine has been widely used in scientific data analysis or performance evaluation for lossy compression research such as [21], [43].

*2) Datasets:* We evaluate our method using multiple application datasets from real scientific applications across different domains. Most of the datasets can be downloaded from the SDRBench database [44]. These datasets are frequently

used in recent studies [37], [45]–[48]. The datasets used in our experiments satisfy following criteria: (1) training and testing datasets are largely different from the perspective of compression ratio and visualization; (2) they are all drawn from real-world scientific datasets that cover various domains such as cosmology, weather etc; (3) they conform to the two capability levels mentioned in Section IV-A. We describe the datasets in Table V.

TABLE V: Description of Application Datasets

| App. | # Fields | TSteps | Dim | Size | Domain |
|---|---|---|---|---|---|
| Nyx-1 | 4 | 6 | 512×512×512 | 12.00GB | Cosmology |
| Nyx-2 | 4 | 1 | 512×512×512 | 2.00GB | Cosmology |
| QMCPack-1 | 1 | 1 | 288×115×69×69 | 0.59GB | Quantum Structure |
| QMCPack-2 | 2 | 1 | 480×115×69×69 | 1.96GB | Quantum Structure |
| QMCPack-3 | 2 | 1 | 816×115×69×69 | 3.33GB | Quantum Structure |
| RTM-Small | 1 | 7 | 449×449×235 | 1.24GB | Seismic Wave |
| RTM-Big | 1 | 2 | 849×849×235 | 1.26GB | Seismic Wave |
| Hurricane | 2 | 7 | 100×500×500 | 1.30GB | Weather |

As shown in the table, we glean a total of 56 different simulation configuration and snapshot datasets from 4 scientific applications across different domains. Other fields show similar results. We perform our experiments to assess both capability levels proposed in Section IV-A.

**Capability level 1 Assessment**: The Hurricane Isabel simulation dataset (shown as Hurricane in the table) is used to assess capability level 1. We use two Hurricane Isabel fields, QCLOUD and TC, for our experiment. For each field, we train our framework FXRZ using the 6 time steps chosen uniformly (5, 10, 15, 20, 25, 30) and then test the accuracy by the last time step 48 of the corresponding fields.

**Capability level 2 Assessment**: We use datasets from Nyx, RTM and QMCPack applications to assess the capability level 2. Hurricane is not included for this assessment because we do not have multiple datasets generated by the simulation runs with different configurations for the Hurricane Isabel. For Nyx, we train the model by using the Nyx-1 datasets downloaded from SDRBench database [44] and test the accuracy using the Nyx-2 datasets downloaded from Nyx database [26], these are generated based on different configuration settings in Nyx simulation, which includes four critical fields: Baryon density, Dark matter density, Temperature, and Velocity-X. For Reverse Time Migration (RTM) application, we train the model using 7 snapshots (time step 50, 100, 200, 300, 400, 450, 500) generated by a small-scale simulation (449×449×235) and test the model accuracy with another big-scale simulation dataset (849×849×235). For Qmcpack, we train the model using two small-scale datasets (QMCPACK-1 and QMCPACK-2) of various sizes and test the model with a big-scale dataset (QMCPACK-3). Qmcpack has two fields, Spin0 and Spin1.

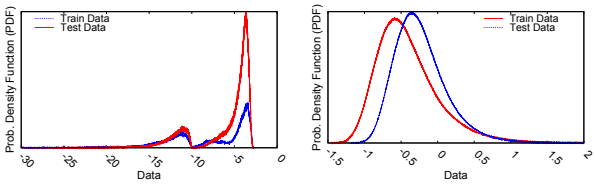
*3) Testing Compressors:* We not only evaluate our framework across different applications but also across four state-of-the-art error-bounded compressors, including SZ [18], [21], ZFP [12], FPZIP [19] and MGARD+ [13], as described below.

- SZ: SZ is an error-bounded lossy compressor, which has been widely tested and used in the community.
- ZFP: ZFP is another error-bounded lossy compressor, which has also been very efficient in lossy compression for scientific datasets. We are using the latest released version 0.5.5 in our experiments.
- FPZIP: FPZIP is an outstanding lossy compressor supporting lossy compression, which allows users to control the data distortion by setting a precision parameter (an integer from 1 to 32) corresponding to different numbers of significant mantissa bits.
- MGARD+: MGARD+ is an accelerated version of the error-controlled lossy compressor MGARD [49].

*4) Baseline:* To the best of our knowledge, there is only one existing compressor-agnostic fixed-ratio lossy compression framework, namely *FRaZ* [20]. FRaZ incurs very high runtime cost because FRaZ needs to iteratively search for the appropriate error bound setting (from the comprehensive range of error bounds) and run the compressor with each explored error bound setting to measure the compression ratio if this would yield to the target compression ratio. For fairness of comparison, the FRaZ is configured as follows: (1) for each testing dataset, we provide FRaZ the same global search range of error bounds (lower and upper error bounds) as we consider for FXRZ, (2) FRaZ allows to divide the whole search range into $k$ bins to allow the search potentially covering as much search space as possible (here error bound settings). We set $k$ to 3 to have a good balance between search coverage and max-iterations, (3) FRaZ allows to set a *max-iterations* for each bin for its search. Thus, *max-iterations* and *number-bins* together provide us total max iterations. We evaluate FRaZ (Using Bebop's single node) under two different max iterations, 6 and 15, to balance its experiment time. Note that the execution time cost by FRaZ under 15 iterations is already considerably longer than the time cost by FXRZ. The reason is that FRaZ needs to run the underlined compressor iteratively multiple times to estimate the appropriate error bound setting, while our framework FXRZ is totally compression-free. We comprehensively compare FXRZ with FRaZ in different facets (such as accuracy, performance), based on different compressors, error bounds, and end-to-end I/O performance on Bebop.

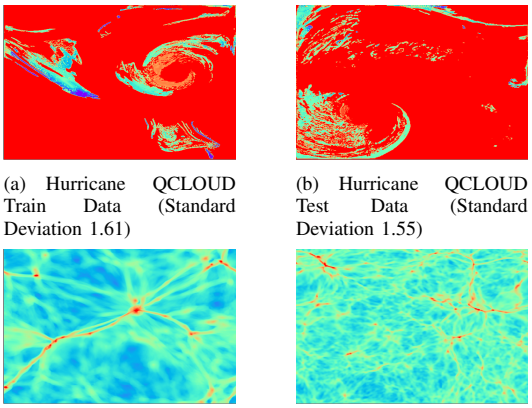
### *B. Demonstration of Variability in Datasets*

As mentioned previously, the datasets used in our experiments exhibit different characteristics between training datasets and testing datasets. To verify this point, we investigate the datasets from the perspective of data distribution, visualization and standard deviation. For the sake of space, we draw two examples from each capability level as shown in Fig. 8 and 9. Other application datasets also exhibit distinct data properties between training and testing datasets. Fig. 8 shows that both Hurricane QCLOUD and Nyx Baryon Density demonstrate different data distributions between training and test data. Fig. 9 demonstrates distinct standard deviation and data visualization between different training and test datasets, based on which we can also clearly observe that Hurricane and Nyx are two largely different representative applications with distinct natures.




(a) Hurricane QCLOUD(Cap.-1) (b) Nyx Baryon Density(Cap.-2)

Fig. 8: Distribution Examples of Train and Test Data



(a) Hurricane QCLOUD Train Data (Standard Deviation 1.61)

(b) Hurricane QCLOUD Test Data (Standard Deviation 1.55)

(c) Nyx Baryon Density Train Data (Standard Deviation 0.42)

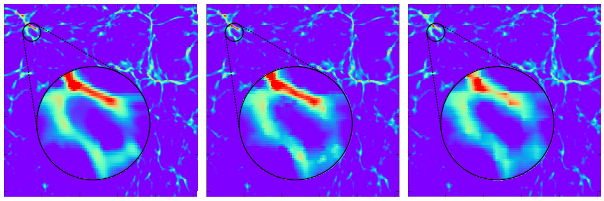
(d) Nyx Baryon Density Test Data (Standard Deviation 0.36)

Fig. 9: Visualization of Training Data vs. Testing Data To Exemplify Their Discrepancy

### C. Analysis of Data Distortion

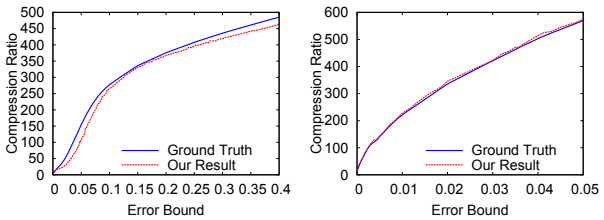
In our evaluation, for each application dataset, we identify the valid range of the compression ratios for users, which corresponds to a very wide range of data distortions definitely covering the acceptable settings for users. In fact, the reconstructed data generated after lossy decompression may have largely different data distortion because of possible considerably different user-specified error bounds.

We use an example to show the distinct reconstructed data qualities when using different error bounds, and the range of error bound configurations we choose for the later accuracy analysis is very comprehensive. Fig. 10 demonstrates the visual quality of reconstructed data by applying SZ on Nyx(Baryon density) with a small error bound 0.05 and a high error bound 0.4, respectively. As shown in the figure (see zoomed region), the error bound of 0.4 suffers from a prominently degraded visual quality compared with the error bound of 0.05. To provide an example of practical significance, we also analyze the distribution of halos' locations in Nyx Baryon Density affected by the lossy compression with various error bounds by the Nyx analysis package [50]. Halos refer to a cluster of particles which simulates the galaxy formation, and this is fundamental information to more sophisticated study in cosmology research. According to our analysis, when the error bounds are set to 0.001, 0.05 and 0.45 respectively, the percentages of halos mislocated from their original position are 0.46%, 10.81% and 79.17% respectively. Such a result



(a) Original (b) Error Bound =0.05 (CR=154) (c) Error Bound = 0.4 (CR=485)

Fig. 10: Visualization of Baryon Density (Using SZ)




(a) Nyx (Baryon Density) (b) QMCPack-3 Spin0

Fig. 11: Valid Compression Ratio Range for FXRZ (using SZ)

shows that the error bound configuration (specifically, it is from 1E-5 to 0.4 in our experiments) used in our evaluation covers a very wide range according to Nyx user's quality-of-interest. That is, without loss of generality, our assessment is conducted with comprehensive error bound settings. Fig 11 demonstrates the valid range of compression ratios (with SZ) for two example datasets – Nyx(Baryon Density) and QMCPack-3(spin0), respectively. As shown in Fig. 11 (a), for example, we choose the compression ratios from 0 to ∼500 for Nyx(Baryon Density), because larger ratios would cause significant data distortions, as shown in Fig. 10(c). The valid range of compression ratios of other datasets are chosen similarly based on reasonable data distortion.

### D. Analysis of FXRZ Training Time

We provide the training time (using Bebop's single node) of FXRZ across different application dataset and compressors in Table VI. For each application and compressor, total training time includes obtaining stationary points (discussed in Section IV-B), augmenting data by interpolation and *RFR* training time. As shown in Table VI, FXRZ incurs very low training time overhead, on average, 13.59 mins. Thanks to the linear interpolation (discussed in Section IV-B) for making total training time very low, as we do not need to run the compressor too many times to obtain ample training data. Note that, training times with MGARD+ is relatively higher because time to obtain the stationary points with MGARD+ is higher because MGARD+ usually has higher compression time than other compressors. Note that, this training time is a one-time cost as we do not need to run compressors at all during the inference phase to apply FXRZ to estimate error bound setting for a target compression ratio.

### E. Validation of Optimization: Adjusting TCR

We now analyze the impact of our optimization (*opt*), *adjusting target compression ratio*, on the estimation accuracy by

TABLE VI: Total Training Time (in Mins) by FXRZ

| Comp. | Nyx-1 Baryon | Nyx-1 Dark Matter | Nyx-1 Temperature | Nyx-1 Velocity-X | QMCPack-1 & 2 | RTM-Small | Hurricane QCLOUD | Hurricane TC |
|---|---|---|---|---|---|---|---|---|
| SZ | 14.92 | 16.45 | 14.17 | 15.10 | 13.83 | 5.43 | 2.79 | 3.14 |
| ZFP | 22.58 | 23.87 | 24.83 | 21.26 | 16.88 | 3.04 | 2.27 | 4.90 |
| MGARD+ | 31.48 | 26.93 | 32.38 | 28.31 | 23.39 | 10.44 | 4.75 | 5.53 |
| FPZIP | 11.30 | 14.86 | 10.08 | 10.73 | 10.57 | 3.98 | 2.43 | 2.34 |

comparing the average estimation errors between *opt* and without *opt* for different datasets across different compressors in Table VII. We pick one dataset from each of four applications because of space limit. Other datasets show similar results. We use Formula 5 and the same set of *TCR*s used in Section V-F to measure average estimation errors. In Table VII across different datasets and compressors, the average estimation errors with *opt* can be as much as 15.23% and on average 4.13% lower than those without the *opt*. The key reason is that after the TCR adjustment, ACR (Section IV-E2) reveals the true data density in a dataset to project the compressibility better. Consequently, these results across different datasets and compressors show the effectiveness of our *opt* towards more accuracy in FXRZ.

TABLE VII: Comparison of Average Estimation Error with and without Adjusting Target Compression Ratio Optimization

| Comp. | Nyx Baryon Density | | QMCPack BigScale Spin0 | | RTM BigScale Snap.-800 | | Hurricane TC Snap.-48 | |
|---|---|---|---|---|---|---|---|---|
| | **With Opt** | Without Opt | **With Opt** | Without Opt | **With Opt** | Without Opt | **With Opt** | Without Opt |
| SZ | 7.63% | 17.14% | 1.52% | 2.13% | 13.88% | 20.88% | 4.50% | 10.90% |
| ZFP | 6.95% | 22.18% | 2.59% | 5.53% | 3.17% | 8.37% | 14.92% | 8.63% |
| MGARD+ | 22.76% | 30.15% | 1.16% | 2.94% | 13.81% | 19.23% | 6.51% | 9.66% |
| FPZIP | 5.36% | 14.56% | 5.83% | 6.56% | 6.89% | 6.83% | 8.10% | 3.15% |

### *F. Accuracy Evaluation*

*1) Evaluation Based on Capability Level 1 and 2:* We compare the accuracy of FXRZ with the baseline FRaZ in Fig. 12 and 13 based on different application datasets. We determine accuracy by how close the *measured compression ratio (MCR)* (obtained from FXRZ estimated error bound setting) to the *target compression ratio (TCR)* is.

Without loss of generality, on average, we set *TCR*s to 25 different values uniformly (to balance the experiment time) which are all reasonable/applicable according to their visualization and data distortion (Section V-C). The reasonable settings have to be tuned slightly across compressors, because some compressor such as ZFP cannot reach too high compression ratio as SZ does. After that, we run FXRZ and FRaZ to get the estimated error configurations, based on which we run corresponding compressors to verify the accuracy of compression ratios.

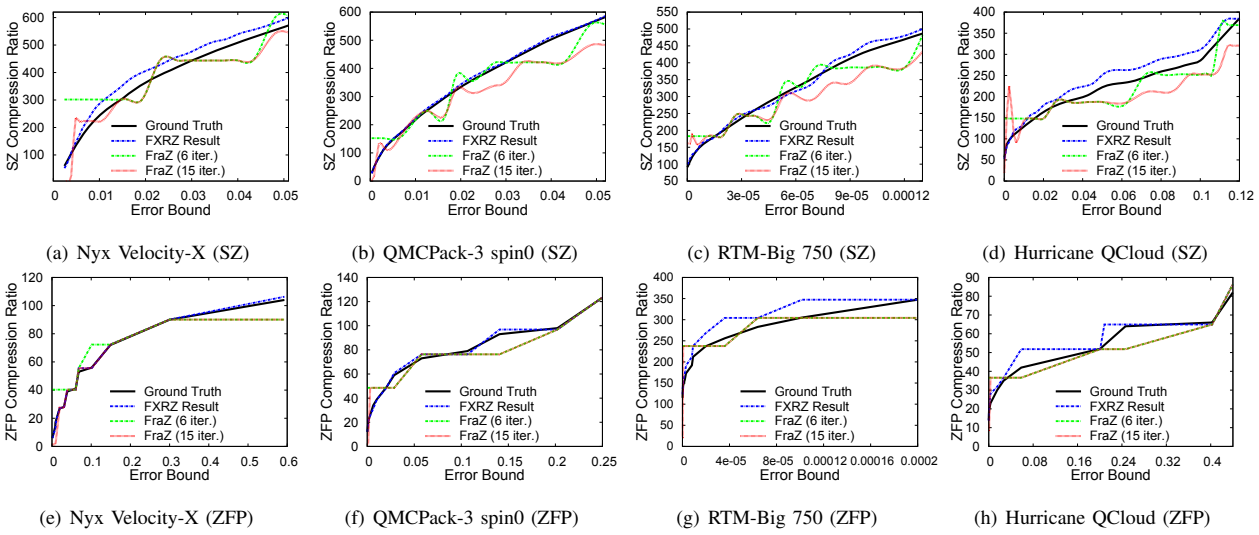
In Fig. 12, for the sake of space, we demonstrate the accuracy result of one testing field or snapshot dataset per application based on both SZ and ZFP. In the figure, we use *Ground Truth* to denote the *TCR*. We show the results of FRaZ based on 6 and 15 iterations along with results of our framework FXRZ in the figure. As shown in the figure, FRaZ struggles to maintain a good accuracy for both 6 and 15 iterations compared with FXRZ in most of the situations. We observe that with higher iterations (15 iterations), FRaZ is able to lower the estimation error (higher accuracy accordingly) compared with the setting of 6 iterations. The key reason is that FRaZ adopts a trial-and-error mechanism to search for the expected error configuration, which relies on a large number of iterations to converge. As such, when FRaZ is given more iterations, its performance overhead would turn dramatically high (to be shown in Table VIII) because of its inevitably multiple expensive runs of the underlying compressors.

On the other hand, we observe that our FXRZ exhibits a high accuracy in most of the cases, despite projecting slight errors with the ZFP compressor to a certain extent. This is not the issue of FXRZ but because of ZFP's characteristic. Specifically, ZFP's compression ratio increases **piecewisely** with the error bound (Section IV-B), so that in some cases there is no exact *MCR* (compressor derived) in correspondence to the *TCR*. That is, some *TCR*s cannot be realized by ZFP in practice, so that no estimation method can match that targets in principle. This is why the FXRZ with ZFP may have slightly lower or higher *MCR*s compared with target compression ratio, as shown in the figure.

Furthermore, in Fig. 13, we show estimation error (in percentage) between *TCR* and *MCR* for each testing snapshot or simulation configuration dataset generated from 4 real-world scientific applications. *MCR* is obtained by running the corresponding compressor based on the model estimated error configuration for each testing dataset. For each testing snapshot or simulation configuration dataset, we show the result by averaging all the estimation errors in our experiments with a corresponding compressor. The estimation error is calculated based on Formula (5). Fig. 13 shows that FXRZ have very low average estimation errors in most of the cases. On average, FXRZ exhibits only 8.24% estimation error across all the four compressors. On the other hand, FRaZ with 15 iterations exhibits low estimation error than that of FRaZ with 6 iterations, which is consistent with our previous analysis. More specifically, their estimation errors across four compressors are 34.48% and 19.37% on average, respectively.

$$EstimationError = \frac{|TCR - MCR|}{TCR} \tag{5}$$

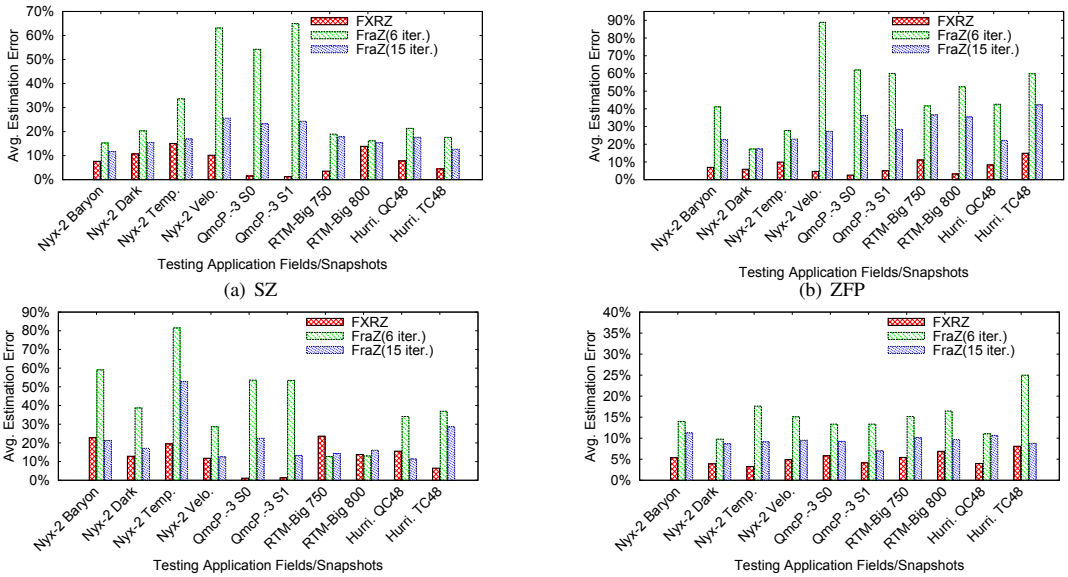
*2) Evaluation Based on Different Application Scopes:* Here we show the robustness of our FXRZ by evaluating it across different application scopes. We perform FXRZ training with datasets from Nyx, QMCPack, Hurricane and RTM-SmallScale. Then, we test the model with RTM-BigScale dataset. Note that obtaining a high accuracy in such a case is very challenging because the training datasets with various application scopes may distract the model's attention and the datasets outputted by RTM-BigScale and RTM-SmallScale have distinct precision. As shown in Fig 14, FXRZ still maintains low average estimation errors which are 11.49%, 6.76%, 13.66%, 19.81% while FRaZ shows 17.85%, 35.51%, 14.31%, 10.11% for SZ, ZFP, MGARD+ and FPZIP respectively. This demonstrates that the features we exploit is fairly effective in characterising data properties during inference phase even when different application domains are present in training data.




(a) Nyx Velocity-X (SZ) (b) QMCPack-3 spin0 (SZ) (c) RTM-Big 750 (SZ) (d) Hurricane QCloud (SZ)

(e) Nyx Velocity-X (ZFP) (f) QMCPack-3 spin0 (ZFP) (g) RTM-Big 750 (ZFP) (h) Hurricane QCloud (ZFP)

Fig. 12: Estimation Error Comparison between FXRZ and FRaZ (with 6 and 15 iterations)



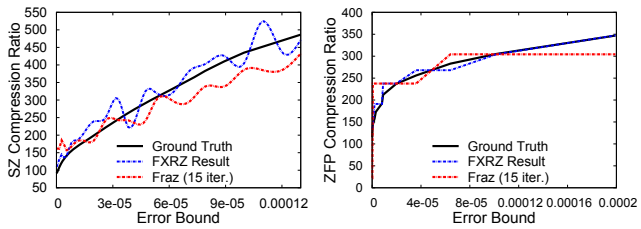

(a) SZ (b) ZFP

(c) MGARD+ (d) FPZIP

Fig. 13: Average Estimation Error across Different Target Compression Ratios for FXRZ and FRaZ (with 6 and 15 iterations). Velo. stands for Velocity-X, S0 stands for Spin0, S1 stands for Spin1, QC stands for QCloud.

### G. Performance Evaluation

In Table VIII, we demonstrate the efficiency of our model FXRZ by comparing the average analysis time required by FXRZ and FRaZ (with 15 iterations) with respect to the *compression time*. We obtain the *compression time* by running the corresponding compressor once for each of the testing dataset. We show the results of FRaZ based on 15 iterations because FRaZ has a much better accuracy (low estimation error) with 15 iterations than that with 6 iterations. We refer to *analysis time* as the time required for estimating the expected error configuration based on *TCR*. In our experiment, for any snapshot or simulation configuration dataset, we calculate *average analysis time* by averaging all the analysis times required for estimating error configurations based on each of the uniformly selected 25∼30 different *TCR*s. Finally, we acquire the *average analysis time cost* as the ratio of the *average analysis time* to the *compression time*. For FXRZ, *analysis time* mainly involves feature extraction, calculation of percentage of non-constant blocks, and time taken by the *RFR* model to predict the expected error configuration. For FRaZ, *analysis time* is the search time for estimating the expected error configuration iteratively based on a given *TCR*. As shown in Table VIII, the analysis time cost by FRaZ is significantly




(a) RTM BigScale with SZ (b) RTM BigScale with ZFP

Fig. 14: Estimation Error across Different Application Scopes

larger than FXRZ, as FRaZ requires expensive iterative search. Quantitatively, on average, FRaZ can be 108× slower than FXRZ to find the desired configuration.

TABLE VIII: Average Analysis Time Cost Relative to *Compression Time*: FXRZ vs. FRaZ with 15 Iterations

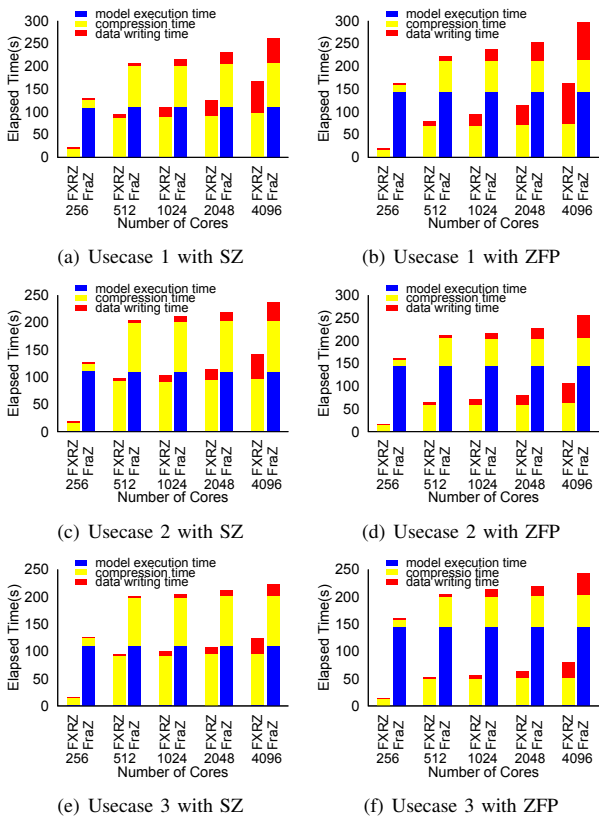
| App. | Test Fields/ Snapshots | SZ | | ZFP | | MGARD+ | | FPZIP | |
|---|---|---|---|---|---|---|---|---|---|
| | | **FXRZ** | FRaZ | **FXRZ** | FRaZ | **FXRZ** | FRaZ | **FXRZ** | FRaZ |
| Nyx | Baryon Density | 0.07x | 5.04x | 0.06x | 6.27x | 0.07x | 13.86x | 0.08x | 17.31x |
| | Dark Matter Density | 0.07x | 6.02x | 0.08x | 7.51x | 0.09x | 16.43x | 0.06x | 10.75x |
| | Temperature | 0.07x | 6.03x | 0.05x | 6.46x | 0.07x | 17.29x | 0.11x | 25.36x |
| | Velocity-X | 0.08x | 5.53x | 0.06x | 5.98x | 0.09x | 23.78x | 0.11x | 20.84x |
| QMCPack | BigScale Spin0 | 0.08x | 7.72x | 0.07x | 7.69x | 0.068x | 11.42x | 0.08x | 22.01x |
| | BigScale Spin1 | 0.08x | 7.78x | 0.07x | 5.42x | 0.07x | 11.51x | 0.08x | 17.12x |
| RTM | BigScale Snapshot-750 | 0.09x | 8.10x | 0.68x | 32.72x | 0.12x | 21.29x | 0.13x | 16.85x |
| | BigScale Snapshot-800 | 0.09x | 7.23x | 0.71x | 29.41x | 0.13x | 19.44x | 0.13x | 15.79x |
| Hurricane | QCLOUD Snapshot-48 | 0.20x | 9.14x | 0.47x | 15.57x | 0.22x | 14.09x | 0.22x | 14.78x |
| | TC Snapshot-48 | 0.17x | 5.93x | 0.19x | 14.84x | 0.18x | 21.20x | 0.18x | 25.72x |
| Average | Across All Domains | 0.10x | 6.85x | 0.24x | 13.19x | 0.11x | 17.03x | 0.12x | 18.65x |

### *H. Parallel Data Dumping Evaluation*

We use Fig. 15 to demonstrate the significant performance gains of our FXRZ compared with FRaZ, when writing the compressed data to the parallel file system (PFS) on a supercomputer – Bebop with up to 4,096 cores. In the experiment, we let each core process a fixed amount of Nyx simulation data (i.e., 2GB), so the total data volume increases with the number of cores. Our experiment follows weak scaling to show the execution scalability: data volume increases as the number of cores grows and the amount of data per core stays the same. The model execution time stays constant because it depends on the amount of data processed per core. The data writing time increases because of the increasing total volume of data with the number of cores. According to the figure, larger execution scale causes longer wall-clock time (including model execution time, compression time and data writing time), because of more data to process at runtime. FXRZ significantly outperforms FRaZ (with a performance gain of 1.18∼8.71×) because of its relatively very high performance in the runtime estimation/analysis.

## VI. Conclusion and Future Work

In this paper, we propose a novel efficient feature-driven compressor-agnostic lossy compression framework (FXRZ) to efficiently estimate appropriate error bound setting based on a target compression ratio. We evaluate FXRZ using 4 lossy compressors with 10 real-world datasets from 4 different applications across different domains. The key findings are: (1)




(a) Usecase 1 with SZ (b) Usecase 1 with ZFP

(c) Usecase 2 with SZ (d) Usecase 2 with ZFP

(e) Usecase 3 with SZ (f) Usecase 3 with ZFP

Fig. 15: Parallel Performance Evaluation (Nyx Simulation)

FXRZ is fairly accurate in estimating the error bound setting that would lead to a measured compression ratio which is close to the target compression ratio. The estimation error is only about 8.24%, on average. Even if training data is from different application scopes, FXRZ can still keep a good accuracy, (2) FXRZ always incurs considerably lower (one or more orders of magnitude lower) execution overhead than the baseline FRaZ does. For instance, on average, FXRZ's online analysis time takes only about 14% of the *compression time*. Moreover, FXRZ is 108× faster than FRaZ, (3) FXRZ significantly outperforms FRaZ with a performance gain of 1.18∼8.71× when performing a parallel data dumping on a supercomputer (Bebop), because of its high performance in the runtime analysis. In the future, we plan to further improve the accuracy by exploring other optimization strategies.

## Acknowledgments

This research was supported by the U.S. Department of Energy, Office of Science and Office of Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357. This research was also supported by the U.S. National Science Foundation under Grants OAC-2003709 and OAC-2104023, and grant No. 2211538 and 2211539. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne National Laboratory).

## REFERENCES


[1] "Hdf5. [online]," https://www.hdfgroup.org/solutions/hdf5/.

[2] W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck, A. Huebl, M. Kim, J. Kress, T. Kurc, Q. Liu, J. Logan, K. Mehta, G. Ostrouchov, M. Parashar, F. Poeschel, D. Pugmire, E. Suchyta, K. Takahashi, N. Thompson, S. Tsutsumi, L. Wan, M. Wolf, K. Wu, and S. Klasky, "Adios 2: The adaptable input output system. a framework for high-performance data management," *SoftwareX*, vol. 12, p. 100561, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711019302560

[3] R. Rew and G. Davis, "Netcdf: an interface for scientific data access," *IEEE computer graphics and applications*, vol. 10, no. 4, pp. 76–82, 1990.

[4] "Restful hdf5. [online]," https://support.hdfgroup.org/pubs/papers/RESTful_HDF5.pdf.

[5] "Hdf5 odbc connector – user's guide. [online]," https://www.hdfgroup.org/wp-content/uploads/2017/07/HDF5_ODBC_Users_Guide.pdf.

[6] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. V. Andel, "Nyx: A MASSIVELY PARALLEL AMR CODE FOR COMPUTATIONAL COSMOLOGY," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, feb 2013. [Online]. Available: https://doi.org/10.1088/0004-637x/765/1/39

[7] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *Acm Sigmod Record*, vol. 34, no. 4, pp. 34–41, 2005.

[8] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou, "The researcher's guide to the data deluge: Querying a scientific database in just a few seconds," *Proceedings of the VLDB Endowment*, vol. 4, no. 12, pp. 1474–1477, 2011.

[9] "H5z dynamically loaded filters [online]," https://portal.hdfgroup.org/display/HDF5/HDF5+Dynamically+Loaded+Filters.

[10] "Pnetcdf-sz [online]," https://github.com/Parallel-NetCDF/PnetCDF-SZ.

[11] SZ2.1, https://github.com/szcompressor/SZ, 2022.

[12] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[13] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. Pugmire, M. Wolf, N. Podhorszki, and S. Klasky, "Mgard+: Optimizing multilevel methods for error-bounded scientific data reduction," 2020.

[14] Globus, https://www.globus.org/, online.

[15] "Data storage and transfers at oak ridge leadership computing facility (olcf)," https://docs.olcf.ornl.gov/data/index.html#data-storage-and-transfers.

[16] "Campaign storage purge policy has been updated (arc ncar)," https://arc.ucar.edu/articles/131, 2021.

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

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

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

[20] R. Underwood, S. Di, J. C. Calhoun, and F. Cappello, "Fraz: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 567–577.

[21] 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," 2018.

[22] Zstandard, http://facebook.github.io/zstd/, 2018, online.

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

[24] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression rate-distortion from automatic online selection between sz and zfp," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[25] X. Liang, S. Di, S. Li, D. Tao, B. Nicolae, Z. Chen, and F. Cappello, "Significantly improving lossy compression quality based on an optimized hybrid prediction model," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356193

[26] Nyx, https://portal.nersc.gov/project/nyx/highz/512/, 2013, online.

[27] J. Kim, A. D. Baczewski, T. D. Beaudet, A. Benali, M. C. Bennett, M. A. Berrill, N. S. Blunt, E. J. L. Borda, M. Casula, D. M. Ceperley, S. Chiesa, B. K. Clark, R. C. Clay, K. T. Delaney, M. Dewing, K. P. Esler, H. Hao, O. Heinonen, P. R. C. Kent, J. T. Krogel, I. Kylänpää, Y. W. Li, M. G. Lopez, Y. Luo, F. D. Malone, R. M. Martin, A. Mathuriya, J. McMinis, C. A. Melton, L. Mitas, M. A. Morales, E. Neuscamman, W. D. Parker, S. D. P. Flores, N. A. Romero, B. M. Rubenstein, J. A. R. Shea, H. Shin, L. Shulenburger, A. F. Tillack, J. P. Townsend, N. M. Tubman, B. V. D. Goetz, J. E. Vincent, D. C. Yang, Y. Yang, S. Zhang, and L. Zhao, "QMCPACK: an open sourceab initioquantum monte carlo package for the electronic structure of atoms, molecules and solids," *Journal of Physics: Condensed Matter*, vol. 30, no. 19, p. 195901, apr 2018. [Online]. Available: https://doi.org/10.1088/1361-648x/aab9c3

[28] H.-W. Zhou, H. Hu, Z. Zou, Y. Wo, and O. Youn, "Reverse time migration: A prospect of seismic imaging methodology," *Earth-Science Reviews*, vol. 179, pp. 207–227, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012825217306256

[29] L. C. L. S. (LCLS-II), https://lcls.slac.stanford.edu/, 2017, online.

[30] T. E. Fornek, "Advanced photon source upgrade project preliminary design report," 9 2017.

[31] F. Cappello, S. Di, S. Li, X. Liang, G. M. Ali, D. Tao, C. Yoon Hong, X.-c. Wu, Y. Alexeev, and T. F. Chong, "Use cases of lossy compression forfloating-point data in scientific datasets," *International Journal of High Performance Computing Applications (IJHPCA)*, 2019.

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

[33] "Anl theta," https://www.alcf.anl.gov/support-center/theta-and-thetagpu.

[34] ORNL, "Summit supercomputer," https://www.olcf.ornl.gov/summit/, 2022.

[35] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356155

[36] "Hurricane isabel dataset," http://vis.computer.org/vis2004contest/data.html, 2004.

[37] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1643–1654.

[38] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[39] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, 1996.

[40] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[41] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[42] Bebop supercomputer, Available at https://www.lcrc.anl.gov/systems/resources/bebop, 2021, online.

[43] X. Liang, S. Di, D. Tao, S. Li, B. Nicolae, Z. Chen, and F. Cappello, "Improving performance of data dumping with lossy compression for scientific simulation," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, 2019, pp. 1–11.

[44] 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)*, 2020, pp. 2716–2724.

[45] J. Tian, S. Di, X. Yu, C. Rivera, K. Zhao, S. Jin, Y. Feng, X. Liang, D. Tao, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data on gpus," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2021, pp. 283–293.
[46] S. Jin, S. Di, S. Byna, D. Tao, and F. Cappello, "Improving prediction-based lossy compression dramatically via ratio-quality modeling," *arXiv preprint arXiv:2111.09815*, 2021.
[47] Y. Liu, S. Di, K. Zhao, S. Jin, C. Wang, K. Chard, D. Tao, I. Foster, and F. Cappello, "Optimizing multi-range based error-bounded lossy compression for scientific datasets," in *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 2021, pp. 394–399.
[48] X. Liang, S. Di, S. Li, D. Tao, B. Nicolae, Z. Chen, and F. Cappello, "Significantly improving lossy compression quality based on an optimized hybrid prediction model," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–26.
[49] 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, pp. 65–76, Dec 2018.
[50] "Nyx analysis package. [online]," https://amrex-astro.github.io/Nyx/.