Improving Progressive Retrieval for HPC Scientific Data using Deep Neural Network

1st Jinzhen Wang
New Jersey Institute of Technology
Newark, USA
jw447@njit.edu

2nd Xin Liang
University of Kentucky
Lexington, USA
xliang@uky.edu

3rd Ben Whitney
Oak Ridge National Laboratory
Oak Ridge, USA
whitneybe@ornl.gov

4th Jieyang Chen
Oak Ridge National Laboratory
Oak Ridge, USA
chenj3@ornl.gov

5th Qian Gong
Oak Ridge National Laboratory
Oak Ridge, USA
gongq@ornl.gov

6th Xubin He
Temple University
Philadelphia, USA
xubin.he@temple.edu

7th Lipeng Wan
Georgia State University
Atlanta, USA
lwan@gsu.edu

8th Scott Klasky
Oak Ridge National Laboratory
Oak Ridge, USA
klasky@ornl.gov

9th Norbert Podhorszki
Oak Ridge National Laboratory
Oak Ridge, USA
pnorbert@ornl.gov

10th Qing Liu
New Jersey Institute of Technology
Newark, USA
qing.liu@njit.edu

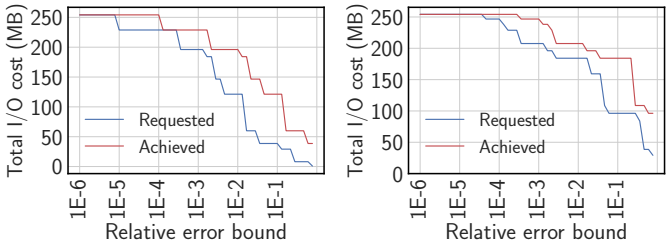
***Abstract*—As the disparity between compute and I/O on high-performance computing systems has continued to widen, it has become increasingly difficult to perform post-hoc data analytics on full-resolution scientific simulation data due to the high I/O cost. Error-bounded data decomposition and progressive data retrieval framework has recently been developed to address such a challenge by performing data decomposition before storage and reading only part of the decomposed data when necessary. However, the performance of the progressive retrieval framework has been suffering from the over-pessimistic error control theory, such that the achieved maximum error of recomposed data is significantly lower than the required error. Therefore, more data than required is fetched for recomposition, incurring additional I/O overhead. In order to tackle this issue, we propose a DNN-based progressive retrieval framework that can better identify the minimum amount of data to be retrieved. Our contributions are as follows: 1) We provide an in-depth investigation of the recently developed progressive retrieval framework; 2) We propose two designs of prediction models (named D-MGARD and E-MGARD) to estimate the amount of retrieved data size based on error bounds. 3) We evaluate our proposed solutions using scientific datasets generated by real-world simulations from two domains. Evaluation results demonstrate the effectiveness of our solution in accurately predicting the amount of retrieval data size, as well as the advantages of our solution over the traditional approach to reducing the I/O overhead. Based on our evaluation, our solution is shown to read significantly less data (5% - 40% with D-MGARD, 20% - 80% with E-MGARD).**

***Index Terms*—High-performance computing, lossy compression, scientific data management, deep learning**

I. Introduction

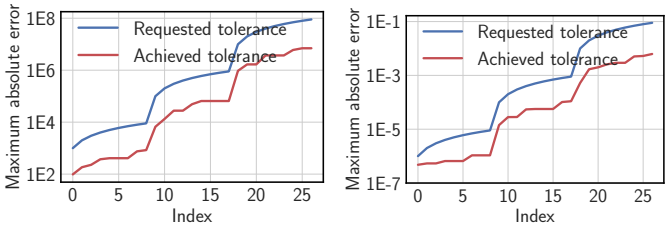
High-performance computing (HPC) is moving rapidly to the era of exascale as empowered by the recent advances in system architecture as well as hardware and software ecosystem. The continuous scaling of application performance has enabled science to be done at an unparalleled microscopic level and new scientific breakthroughs that were not possible in the past. Nevertheless, with the increasing model resolution and fidelity, post-hoc data analytics has become increasingly cumbersome due to the high cost associated with moving data from persistent storage to memory as well as performing computation. Similarly, the aggregation and transportation of large volumes of data across multiple sites incur significant overheads for scientific applications running on grids due to the limited bandwidth of network [1]. To address this challenge, various methods have been developed to allow for more efficient data exploration for scientific simulations, and they generally tackle the issue from the following three dimensions: improving storage and I/O systems [2]–[8], in situ processing [9]–[12], and data reduction [13]–[18] None of these methods are deemed to be the silver bullet to solving the long-standing problem, but rather, they are designed to function at a single level in the system stack and complement each other to achieve the best outcome for applications.

In particular, there has been a multitude of efforts re-designing storage and I/O system for HPC, most notably the insertion of a burst buffer layer [5] into the HPC storage stack, and the use of emerging storage devices, such as NVRAM, die-stacked memory. Meanwhile, in situ processing offers an alternative paradigm that allows data analysis to be done in memory while the simulation is running, without being forced to move data to persistent storage for post-processing. In contrast, data reduction fundamentally addresses the cost of data analysis by lowering the data volume and velocity, and depending on whether there is information loss, the reduction process can be either lossless or lossy. While the former does not incur information loss, which is critical for some applica-



(a) B_x (b) E_x

Fig. 1: The I/O cost incurred by requested tolerance vs. the I/O cost incurred by the over-pessimistic error estimation method. B_x and E_x are scalar fields from the WarpX application.



(a) J_x (b) D_u

Fig. 2: Requested error tolerance vs. the error achieved. It can be observed that the achieved tolerance is constantly lower than requested. J_x and D_u are two scalar fields from WarpX as well as Gray-Scott application respectively.

tion scenarios (e.g., such as checkpoint/restart), its reduction performance is often inadequate to be relevant for handling the massive datasets coming out of scientific simulations. Very recently, lossy data reduction has gained renewed interest from the HPC community. By relaxing the data accuracy in exchange for performance, lossy reduction typically achieves much higher reduction ratios, albeit with highly sophisticated and sometimes application-dependent error control.

Despite the recent success of lossy compression in HPC applications, it suffers from the following weaknesses: 1) Nearly all lossy compressors require data to be compressed at a given error bound. Once the data is compressed, the error tolerance cannot be adjusted after the fact. This causes problems for datasets that are shared within a large community of users where the accuracy needs can be diverse. 2) Lossy compressors were mostly designed to reduce the storage footprint, and after decompression, the same degrees of freedom of data will be fed into data analysis. As such, most lossy compressors do not reduce the cost of computation.

Driven by the aforementioned weaknesses, an error-controlled data decomposition and progressive recomposition were recently designed [19]. The key ideas are that, during storage, data are transformed into different levels of precision coefficients using a combination of a multi-grid-like decomposition and bit-plane encoding. Then upon retrieval, a progressive retrieval framework fetches part of the decomposed data and recomposes an approximation to the original data with reduced accuracy. A major advantage is that the degrees of freedom are reduced so that the cost of both I/O and computation can be controlled in a fine-grained manner. However, as pointed out by another work [20], the progressive retrieval framework often fetches more data than needed, as shown in Fig 1, due to the over-pessimistic error control, as shown in Fig 2. Therefore, additional overhead exists during the data retrieval which hurts the I/O performance.

Intuitively speaking, the retrieved data size depends on both the user-prescribed error bounds as well as the data characteristics. For example, more data should be retrieved if users require a higher data accuracy. On the other hand, if the original data demonstrate stronger smoothness, less data is needed for reconstruction. Therefore, the amount of data need to be retrieved is of the complex interplay between data characteristics and the error bounds, and is hard to capture quantitatively. In this paper, we propose to explore the possibility of leveraging the Deep Neural Network (DNN) to capture such complex interaction, so that minimum amount of data can be retrieved to reduce the I/O overhead. This paper makes the following contributions:

- We provide an in-depth analysis to the performance of progressive data retrieval framework;
- We design a DNN-based progressive retrieval framework with two models (D-MGARD and E-MGARD) to predict the number of bit-planes for retrieval, based on user-prescribed error tolerance. To the best of our knowledge, our work is among the first to do so, which offers significant improvement on the I/O overhead.
- We evaluate the proposed DNN-based progressive retrieval framework on scientific datasets generated by real-world HPC simulations from two domains. We first demonstrate the prediction accuracy of our DNN-based prediction models. We show that our DNN-based predictor can accurately capture the numbers of bit-planes under various user-prescribed error bounds for different fields from scientific applications. We also compare the I/O overhead incurred by our DNN-based progressive retrieval framework against traditional theory-based progressive retrieval framework. We demonstrate that our proposed predictor offers significant improvement (5% - 40% with D-MGARD, 20% - 80% with E-MGARD) in reducing the I/O overhead.

The remainder of this paper is organized as follows. In Section II we discuss the background and motivation for our work. Section III formulates the research question and describes the overall design of DNN-based progressive retrieval framework. Section IV presents the evaluation results. Section V discusses the related work along with conclusions in Section VI.

II. Background and Motivation

A. Progressive Data Retrieval

Progressive data retrieval aims to address the problem of high computation cost at full data resolution, as well as data

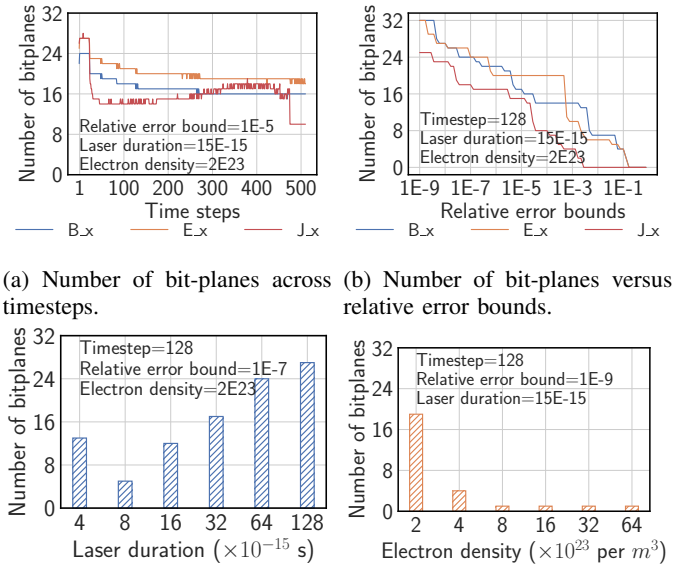
transmission when I/O bandwidth is constrained. The idea is to transmit a small part of the data each time, based on the user-prescribed accuracy requirements, so that the transmission can be finished efficiently. The general approach is first to decompose data into a multi-resolution hierarchy level of coefficients, such that, the highest coefficient level with the lowest resolution contains the most information. The design of such hierarchy (e.g., the number of levels and the resolution of each level) follows the storage hierarchy of HPC systems. For example, the highest level data, which is supposed to be frequently accessed, can be placed on the fastest storage tier (e.g., NVMe); while the lowest level data should be placed on slowest storage tier (HDD or tapes), given it will be least likely to be accessed.

During data transmission, one or more coefficient levels are transmitted based on the system I/O bandwidth and user requirement of data accuracy. Then the transmitted data levels can be recomposed to generate an approximation to the original data with reduced accuracy. Due to the complex I/O bandwidth status and various user requirements, it is vital for the retrieval framework to support fine-grained progressiveness. As such, bit-plane-based encoding schemes are commonly used in representing the coefficient level data, in order to support truncation on the bitstream. For example, Hoang et al. proposed to use binary encoding to represent the tree-based hierarchical data structure [21]. Another example is MGARD [19], [20], a recently developed transform-based lossy compressor that adopts nega-binary encoding for representation, which will be detailed as follows.

B. MGARD

MGARD combines the features of lossy data compression with multi-level decomposition and progressive retrieval. During compression, a data decomposer first transforms original multi-dimensional data to multi-level coefficients using orthogonal L2 projection and interpolation. Then an interleaver linearizes the coefficient levels to 1D for precision encoding. The linearized coefficients are then encoded in the bit-plane fashion and compressed losslessly with ZSTD. In order to support error-bounded progressive retrieval, an error matrix is further collected to represent the error incurred on coefficient levels by partially quantizing bit-planes. The encoded coefficient levels, error matrix as well as other metadata are written to files and placed across the storage hierarchy.

During decompression, MGARD first estimates the number of bit-planes to be retrieved from each coefficient level with a maximum error estimator. Such an error estimator is able to predict the maximum data reconstruction error based on the number of bit-planes as well as the error matrix collected during compression. Thus MGARD recursively finds the minimum number of bit-planes to retrieve that satisfies the user-requested error control. Then a size interpreter calculates the retrieval size as well as the precision segments to fetch from the storage hierarchy. Then the bit-planes retrieved from various precision segments are recomposed to form the decompressed data.



(a) Number of bit-planes across timesteps.

(b) Number of bit-planes versus relative error bounds.

(c) Number of bit-planes versus laser duration.

(d) Number of bit-planes versus electron density.

Fig. 3: MGARD number of bit-planes versus timesteps, relative error bounds and simulation-dependent parameters. B_x, E_x and J_x represent magnetic field, electrical field and current density respectively along x-axis in WarpX laser-driven electron acceleration simulation.

C. Over-aggressive error control

The key capability of a progressive data retrieval framework is to identify the amount of data needed given users' prescribed error bounds. For example, MGARD [20] adopts a theory-based error control scheme, that first computes the significance of each bit-plane that contributes towards the reconstructed data accuracy. Then the number of bit-planes that should be retrieved under a user-prescribed error bound can be deduced by progressively comparing the achieved accuracy and requested error bound for each bit-plane.

Nevertheless, the error control theory developed by the early works [19] estimates the maximum absolute error bound using absolute row sum which neglects the cancellation between positive and negative errors. Therefore, there is a significant gap between the user-requested error tolerance and the actual error achieved, which results in the number of bit-planes retrieved larger than what is required and a higher I/O performance overhead. As shown in Fig. 2, the achieved error tolerance is constantly lower than requested by orders of magnitude. Correspondingly, the achieved I/O cost is significantly higher than requested, as shown in Fig. 1.

Motivation 1: The theory-based error control is overly pessimistic and incurs high overheads for data retrieval. As such, a more precise error control method is needed to improve the I/O performance of data retrieval.

D. High dimensionality of bit-plane retrieval

Given the various scenarios, the number of bit-planes to be retrieved is a multi-variant function that depends on simulation

TABLE I: Notations

Symbols	Description
D	The retrieved data size (bytes).
L	The number of coefficient levels.
e	User-requested maximum absolute error.
err	Maximum absolute error of reconstructed data.
Err	Maximum absolute error of coefficient levels.
B	The total number of bit-planes on each coefficient level.
b_l	The number of bit-planes to retrieve on coefficient level l.
$\mathcal{L}(x, y)$	The loss function used to control the prediction error.
$\mathbb{S}$	The sizes of bit-planes across coefficient levels (bytes).
$\mathcal{F}$	The mapping function from input error bound to b.
F	A set of data features used in DNN training.

timesteps, error tolerances, as well as data characteristics, which are directly affected by simulation input parameters. In Fig. 3, we demonstrate MGARD's number of bit-planes versus such variables. As demonstrated in Fig. 3a, the number of bit-planes across timesteps demonstrates non-linear behaviors. In Fig. 3b, we show that the number of bit-planes reduces while the tolerance loosens. In Fig. 3c and Fig. 3d, we show the number of bit-planes also manifests complicated behaviors with the change of *electron density* and *laser peak amplitude*, which are two initial conditions to the WarpX simulation experiment. Therefore, estimating MGARD's number of bit-planes is a non-linear high-dimensional problem that is difficult to solve with traditional modeling techniques, and we seek to tackle with Deep Neural Network (DNN) as a data-driven approach. **Motivation 2: The number of bit-planes to be retrieved to satisfy a given error bound is highly sophisticated and can be affected by many factors. Therefore it is beneficial to capture using the DNN-based approaches.**

III. DNN-based Progressive Retrieval

In this section we first formulate the research problem of improving progressive retrieval for HPC scientific data towards minimum I/O overhead and then discuss the details of design and implementation. For the convenience of discussion, we adopt MGARD as an example of bit-plane-based progressive data retrieval framework to discuss the details of our design. We first list the commonly used notations in Table I.

A. Problem formulation

As we mentioned above, the I/O overhead, which is also the data retrieval size, depends on both the number of coefficient levels as well as the number of bit-planes of retrieval on each level. Denote $l(0 \leq l < L)$ as the current coefficient level where L is the total number of coefficient levels, $b_l(0 \leq b_l \leq B)$ as the number of bit-planes to retrieve for level l where B is the total number of bit-planes on each coefficient level (32 for single-precision floating-point data), $k(0 \leq k \leq b)$ as the current bit-plane index, and $\mathbb{S}$ as the sizes of bit-planes on each level. We have the data retrieval size D as the accumulation of bit-plane sizes across coefficient levels, as shown in Equation 1.

$$D = \sum_{l=0}^{L-1} \sum_{k=0}^{b_l} \mathbb{S}_{lk} \tag{1}$$

In particular, the total number of coefficient levels L is jointly determined by both data resolution and user input during data decomposition. On the other hand, the sizes of bit-planes across coefficient levels $\mathbb{S}$ also depend on the resolutions of coefficient levels, which are also determined during data decomposition. Therefore, both L and $\mathbb{S}$ can be considered as constant during data retrieval, and that the overall size of progressive retrieval D can be directly calculated by the number of bit-planes of retrieval b_l. Ideally, the purpose of progressive retrieval framework is to compute the number of bit-planes of retrieval b_l based on the user requested maximum absolute error e, as shown in Equation 2, to the extent that the achieved maximum error err after recomposition is less than but very closed to e, so that users can understand the information loss incurred by progressive retrieval. However, as we mentioned before, the current framework achieves a significantly lower maximum absolute error than the one user requested ($err \ll e$), such that the err became almost agnostic to users, while resulting in higher b_l and larger D.

$$b_l = \arg\min_e \mathcal{F}(e), b_l \in [b_0, b_1, ...b_{L-1}] \tag{2}$$

In this work, our objective is to improve the performance of the progressive retrieval framework by optimizing the error control mechanism of the original MGARD. In particular, we aim to find a better mapping function that users can utilize to guide the choice of error bounds and data retrieval. While the relation between e and b_l is unclear, err and b_l are corresponding to each other by nature. Therefore, we aim to directly learn the mapping between the achieved maximum error and the number of bit-planes, leading to our first approach.

Approach A: For each field of data from scientific applications, we first perform compression experiments using the original MGARD under an extensive set of input error bounds e, record the number of bit-planes of retrieval b_l as well as the achieved error err. Then, we can train a DNN model to predict b_l with err, which can be used to take the place of the error interpreter in the original MGARD. This predictive model, named D-MGARD, takes the achieved maximum error err, along with a set of data features, as input and directly predicts the number of bit-planes of retrieval b_l for each coefficient level. It bypasses the original error control mechanism in MGARD completely and predicts the number of bit-planes based on the inherent correlation from the data retriever.

The reasons why we choose DNN over traditional ML methods for D-MGARD are mainly two-fold: 1) Given the large problem space and complexity of our work (large numbers of simulation configurations, timesteps, and compression error bounds), the number of bit-planes exhibits extremely complicated and non-linear behaviors where DNN is naturally anticipated to perform well; 2) With the possible extension of problem space, where we seek to predict more data attributes with the model, the feature selection and engineering is challenging for ML methods whereas DNN can benefit from more advanced model architecture to extract features automatically.

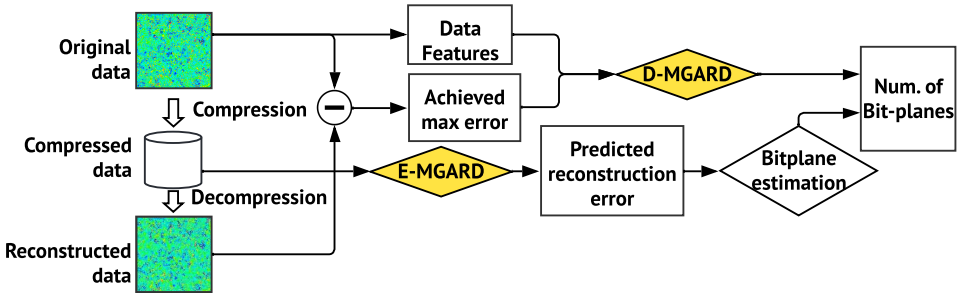


Fig. 4: The overview of DNN-based progressive data retrieval framework. Highlighted components indicate our design.

As D-GMARD achieves the prediction of the number of bit-planes in a purely data-driven way, such model provides limited interpretability as it treats the mapping as a black box. In the following, we propose another approach that is more closely coupled with the philosophy of MGARD.

Approach B: Upon recomposition, MGARD estimates the reconstruction error based on the partially retrieved coefficient levels. Given the resolution and mesh structure of the original dataset, the coefficient level error Err is converted to data reconstruction error err. As we discussed previously, the pessimistic error estimation from Err to err is the key reason for the aggressive error control which leads to the extra I/O overhead. Therefore we propose to improve the error estimation with DNN. We propose to design a DNN model, named E-MGARD to predict the data reconstruction error err given the coefficient level error Err. Such a model replaces the original MGARD to achieve more precise error control.

B. Design overview

We present the overall design of the framework in Fig 4. We showed D-MGARD and E-MGARD in the the framework as we previously introduced. In the following, we discuss the design details of D-MGARD and E-MGARD towards the numbers of bit-planes prediction, as well as the selection between two approaches.

C. D-MGARD: number of bit-planes estimation

The idea is to learn a prediction model that directly maps between the number of bit-planes and the achieved maximum error of reconstructed data. Such that when users require the reconstructed data to a specific maximum error, our model can predict the number of bit-planes that need to be fetched. This approach does not take into consideration how MGARD handles the input data, but rather treats all processing and transformation operations of MGARD as a black box. The D-MGARD contains the following steps: 1) run the compression experiments under a set of absolute errors; 2) collect the achieved maximum errors as well as the numbers of bit-planes fetched from coefficient levels under achieved absolute errors; 3) train a multi-target prediction model with the achieved maximum error as input and numbers of bit-planes as the target. In order to account for the impact of data characteristics on the performance of MGARD, the prediction model also take a set of statistical data features as input.

Compared to the traditional theory-based predictor, the proposed D-MGARD directly learns the mapping between the number of bit-planes to the achieved maximum absolute error. In order to achieve this, the error bounds that we feed into the model as input are *achieved* error (red curves in Fig.2) we collected before training, rather than the one users requested.

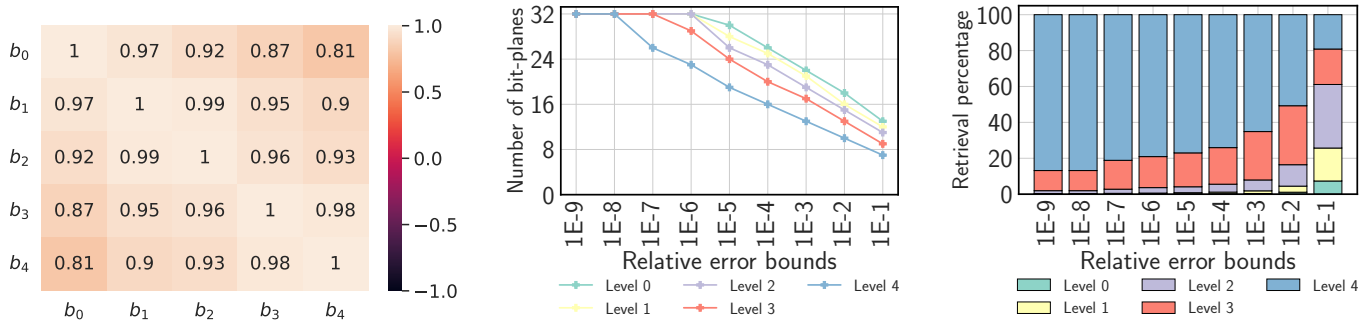
As we previously discussed, D-MGARD essentially performs *regression* tasks by taking data features and user-requested error bounds as input and producing the number of bit-planes for each coefficient level as output. Generally speaking, we consider the scenarios where the data is decomposed to more than one level. Therefore, such a regression task is also called *multi-output regression*.

The first and most intuitive approach is to train a multi-level perceptron (MLP) model with input and output layer dimensions matching the dimension of training variables and targets respectively. However, as pointed out by literature [22], such a MLP model usually suffers from low training accuracy, as the correlation among target variables are not accounted for.

In Fig.5a, we demonstrate the correlation matrix of the numbers of bit-planes across five levels. As shown in the figure, the numbers of bit-planes are strongly correlated with each other. Such correlation is caused by: 1) the values of bit-plane numbers, which all are integers in the range of [0, 32]; 2) the greedy-based bit-plane retriever fetches bit-planes by accuracy efficiency ranking. Therefore, the number of bit-planes to be retrieved from a certain level depends not only on the user-requested error bound, but also the number of bit-planes of all other levels.

The accuracy efficiency for a bit-plane used by MGARD's greedy-based retriever is the ratio between the error reduction by retrieving the bit-plane and its resolution. Based on our investigation, we observe that for bit-planes with the same bit location, those on a higher level generally demonstrate higher accuracy efficiency than those on a lower level, as the difference in the resolution across coefficient levels. Therefore, bit-planes on higher levels are usually retrieved first. We show such behavior in Fig.5b, where we display the number of bit-planes retrieved from each of the five coefficient levels across relative error bounds during data retrieval. As shown in the figure, level 0 always contributes the most bit-planes while level 4 always contributes the least.

We further demonstrate the breakdown of bit-plane sizes across five coefficient levels in Fig.5c. It is interesting to



(a) Correlation matrix of the numbers of bit-planes on five levels.

(b) The number of bit-planes retrieved from coefficient levels across error bounds.

(c) The breakdown of retrieval size (%) from coefficient levels across error bounds.

Fig. 5: MGARD retrieval performance across relative error bounds. The values are collected by running MGARD on WarpX laser-driven electron acceleration simulation dataset. In this figure, level_0 is the highest level with the lowest resolution.

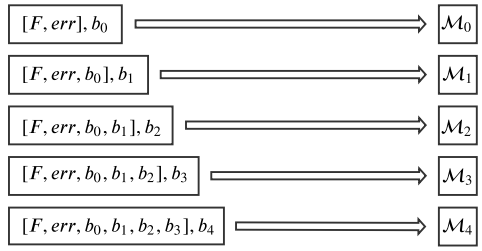
show that, despite the lowest number of bit-planes level 4 contributing to the retrieved data, it holds the most significant proportion of the retrieved data size under most error bounds. The only exception is the relative error bound of 1E-1 where the data accuracy requirement is very low so that there is almost no need to read from level 3 and level 4.

Such breakdown suggests that the numbers of bit-planes across the coefficient levels are not of the same importance, rather those on a lower level play more important roles in determining the retrieved data size. Therefore, in order to minimize the I/O overhead, it is more important to capture the numbers of bit-planes for lower levels than for higher levels.

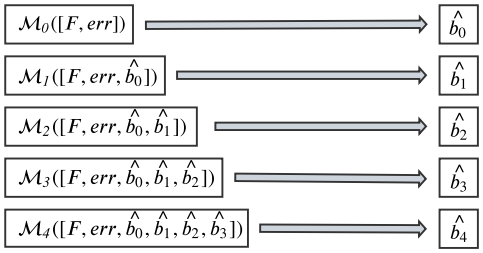
In order to leverage the correlation among the numbers of bit-planes and consider the weighted importance of numbers of bit-planes across coefficient levels, we design D-MGARD as a chained multi-output regression model (CMOR) as shown in Fig.6. Let $L = 5$ in this case as an example. Essentially, the idea is to train a separate MLP for each level, denoted as $\mathcal{M}_l (0 \leq l < 5)$, to capture the number of bit-planes. However, in order to leverage the correlation among the numbers of bit-planes, the training variables for each level include not only the general variables like data features $\mathbb{F}$ and achieved maximum absolute error err, but also the numbers of bit-planes from previous levels $[b_0, ...b_{l-1}]$. For example, as shown in Fig. 6a, $\mathcal{M}_1$ is trained with b_0 as an additional feature, $\mathcal{M}_2$ with b_0 and b_1, ... and so on. During the online prediction, as shown in Fig.6b, the number of bit-planes for each level is predicted with each pre-trained model following a sequential order, from level 0 to level 4.

Such model design is closely coupled with the greedy-based bit-plane retriever as well as the characteristics of numbers of bit-planes we discussed above. Specifically, the most important number of bit-planes (on level 4) is trained with most features. On the other hand, all MLP models can be trained in parallel, without causing additional training overhead.

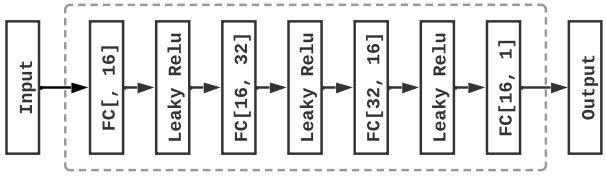
The network architecture of the MLP model on each level is shown in Fig. 6c. We configure the network with six fully-connected hidden layers and activation function of leaky ReLu [23] in between. We note that the dimension of the input



(a) Offline training stage. Those variables included in brackets are training features and b_l on each level is the training target.



(b) Online prediction stage. Variables with hat symbols are predicted by MLP model.

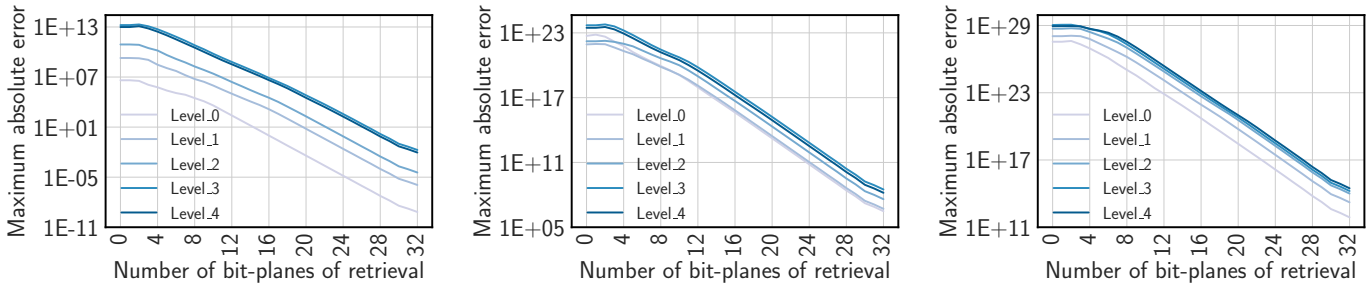


(c) Multi-level perceptron (MLP) network used to predict the number of bit-planes on each level. Note that the output dimension of MLP model on each level is always one, while the input dimension is different across the levels.

Fig. 6: Chained multi-output regression (CMOR) model for a five-level hierarchy.

layer is different across levels, accounting for the additional training features on each level, while the dimension of the output layer is always one.

During the model training, we seek to minimize the loss function, which is the average prediction error of bit-planes numbers for each level, as shown in Equation 3.



(a) B_x (b) E_x (c) J_x

Fig. 7: The absolute error of progressive retrieval from coefficient levels. The values are based on the three fields from the WarpX application at timestep t=32.

$$\mathcal{L}(b_i, \hat{b_i}) = \frac{1}{L}\sum_{i=0}^{L-1} \ell(b_i, \hat{b_i}) \tag{3}$$

Common choices of loss function $\ell(x, y)$ generally include the mean squared error (MSE) and the mean absolute error (MAE). However, based on our investigation, neither is a good choice in our case. On the one hand, controlling prediction error with MAE often leads to long tails in the distribution of prediction error, indicating the existence of large outliers. The reason is that MAE does not penalize the large prediction errors enough. On the other hand, controlling with MSE can capture those outliers fine but often leads to large average prediction errors. The reason is that MSE, by taking square on prediction errors, is not sensitive to small prediction errors. Based on empirical experiments on various loss functions, we observe superior training accuracy can be obtained in our case by controlling the prediction error with the Huber loss function [24], which is defined in Equation 4.

$$\ell(x, y) = \begin{cases} \frac{1}{2}(x-y)^2, & \text{if} |x-y| < \delta \\ \delta(|x-y| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{4}$$

The Huber loss is a combination of MSE and MAE that is less sensitive to outliers than the MSE. It is quadratic for small prediction errors less than a certain threshold δ and linear for large prediction errors beyond δ. In this work, we observe that setting $\delta = 1$ achieves the best training accuracy. Such that the loss function used in our work can be written as follows.

$$\ell(b_i, \hat{b_i}) = \begin{cases} \frac{1}{2}(b_i - \hat{b_i})^2, & \text{if} |b_i - \hat{b_i}| < 1 \\ (|b_i - \hat{b_i}| - \frac{1}{2}), & \text{otherwise} \end{cases} \tag{5}$$

D. E-MGARD: error control optimization

According to the error control of MGARD recomposition, the maximum error between original data and recomposed data is bounded by the following Eq 6, where $Err[l][b_l]$ denotes the absolute error of l-th $(0 \leq l < L)$ coefficient level when retrieving the first $b_l (0 \leq b_l \leq B)$ bit-planes, and C is a constant that maps the absolute error of coefficient levels to the absolute error of the reconstructed data. It has been noted [19] that C depends on the data characteristics, specifically the mesh structure of the input dataset. Therefore, it has to be manually derived based on the mesh structure of each scientific application. Nevertheless, the derived mapping constant still suffers from sub-optimal performance. As shown in Fig 2, the mapping constant yields over-pessimistic error control and thus results in extra I/O overhead.

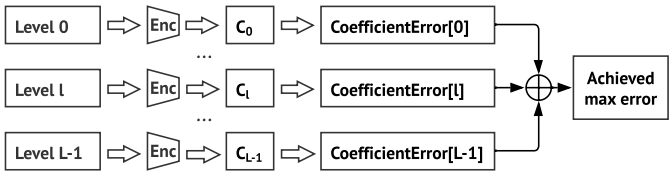


Fig. 8: The design of E-MGARD error prediction model.

$$err \leq C \sum_{l=0}^{L-1} Err[l][b_l] \tag{6}$$

One issue with the current error control approach is that the same mapping constant is applied to all coefficient levels, implying that the absolute error incurred from progressive retrieval on each coefficient level has the same impact on the absolute error of reconstructed dataset. Below we demonstrate that this is not the case.

In Fig 7, we demonstrate the absolute error of each coefficient level incurred by progressively retrieving an increasing number of bit-planes. It can be shown in the figure that, the magnitude of absolute error across coefficient levels shows significant differences. Therefore, when the same mapping constant C is applied to different coefficient levels, the absolute error of the reconstructed dataset is biased towards the absolute error of lower coefficient levels, reducing the granularity of error control and over-pessimistic error estimation.

In order to tackle this issue, we propose to learn a mapping constant for each coefficient level. We present the design of our approach, named E-MGARD, in Fig 8. Specifically, the Enc block in the figure is an encoder network with hidden layer dimensions of 2048, 512, 128 and 8. The activation function is ReLu. As shown in the figure, for each coefficient level from a decomposed dataset, an encoder network transforms it into a set of latent features which is further used to predict the mapping constant for the current level. Then the achieved

maximum error of reconstructed data can be calculated by Equation 7, where $C_l \in \{C_0, C_1, ...C_{L-1}\}$ denotes the learned mapping constant for each coefficient level.

$$err \leq \sum_{l=0}^{L-1} C_l Err[l][b_l] \tag{7}$$

E. Selection between D-MGARD and E-MGARD

Between the two approaches we proposed in this section, the advantage of D-MGARD is the easiness of learning the prediction model, namely users do not need to have a deep understanding of MGARD, or other progressive retrieval methods. However, D-MGARD suffers from mainly two disadvantages: 1) the difficulty to summarize a set of data features to account for the impact of data characteristics on the MGARD performance, which will limit the modeling capability and result in estimation error; 2) the lack of understanding how MGARD processes and transforms the input data. Especially, the lack of understanding of MGARD prevents a more precise prediction model from being developed.

On the other hand, E-MGARD focuses more on improving the error control of the original MGARD. It is more precise on the modeling scope compared to D-MGARD. Especially for the bit-plane estimation using coefficient level error (as shown in Fig 4), where D-MGARD takes it as a part of the black box and estimates with a chained multi-output regression model, E-MGARD leverages the greedy-based estimation method from original MGARD for calculation, which is more accurate by nature. However, the design of E-MGARD is tightly coupled with the recomposition scheme of MGARD, making it hard for E-MGARD to extend to other progressive retrieval methods.

IV. Performance Evaluation

In this section, we present the evaluation results for our DNN-based progressive retrieval framework. We first discuss the experiment setup as follows.

A. Experimental setup

1) Hardware platform: We perform the progressive decomposition and recomposition experiments on Summit supercomputer at Oak Ridge National Laboratory [25]. All experiments are performed on a single CPU-node with two 16-core 3.0 GHz AMD EPYC processors and 256GB of main memory. We train and evaluate the DNN-based progressive retrieval framework on Google Colab environment with a single GPU. The Colab environment is driven by three Intel Xeon processors with 26GB memory, as well as an Nvidia Tesla P100 GPU with 16GB HBM.

2) Scientific datasets: We use Gray-Scott simulation [26] and WarpX simulation [27] as evaluation datasets in our work. The datasets are obtained by running simulations on Summit. The detailed information of the datasets is shown in Table II. All datasets are double-precision floating-point values.

3) MGARD compression experiment: In order to cover various application scenarios where users can have arbitrary error bound requirements, we compress the simulation datasets using MGARD under a large range of relative error bounds, including 81 error bound values. Specifically, the relative error bounds used in this work are [1E-9, 2E-9, ... 8E-1, 9E-1]. We then collect the number of bit-planes on each coefficient level as well as the achieved maximum error associated with each input error bound to assemble the training records for D-MAGRD. We also store the decomposed coefficient level data as the training set for E-MGARD. In order to compress the multiple timesteps of application data, we assume the data range of each field at each timestep are calculated during the simulation and available during compression.

TABLE II: Application datasets

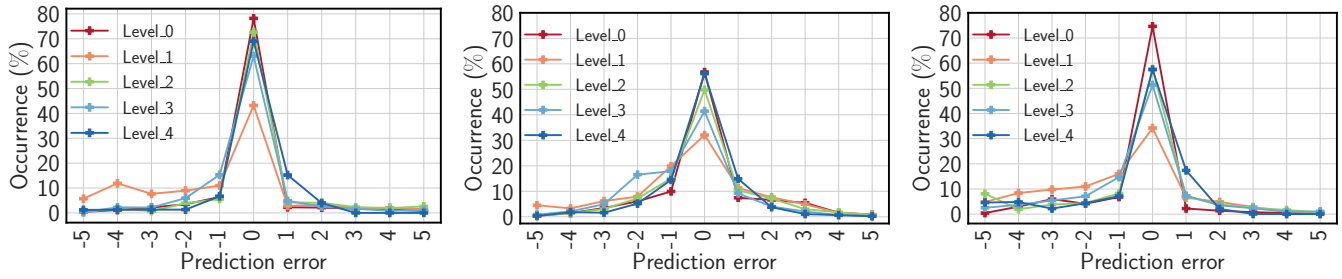
Application	Fields of use	Dimensions	# Timesteps
Gray-Scott	D_u, D_v	512^3	512
WarpX	B_x, E_x, J_x	512^3	512

4) Training configurations: For each field of Gray-Scott and WarpX application, we train our models on the first 256 timesteps of the dataset and test on the remaining ones. The D-MGARD is trained with a learning rate of 0.00005, batch size of 256, and E-MGARD is trained with a learning rate of 0.00001, batch size of 64. Both models have been trained for 300 epochs, and the cost of training for each model is 0.7 hours and 1.2 hours respectively. We would like to point out that, in our work, our split of training data via timesteps is to mimic the way scientific applications dump data during simulation. Throughout the simulation process, applications will periodically dump variable data which are evolving with time. The purpose of training on the first half timesteps and testing on the later half is to evaluate whether the trained model can be reused when the data shows changes over time. Under the scenario of "train once, infer many times", we expect our model can be reused on new datasets generated from the same applications and the training time can be amortized.

For the rest of this section, we evaluate the performance of our proposed framework. Specifically, we first evaluate the prediction accuracy of D-MGARD across simulation timesteps as well as across data resolutions. Then for E-MGARD, we evaluate the achieved maximum error of reconstructed data against input error and achieved maximum error of original MGARD. We then evaluate both approaches on the total retrieval size against the original MGARD.

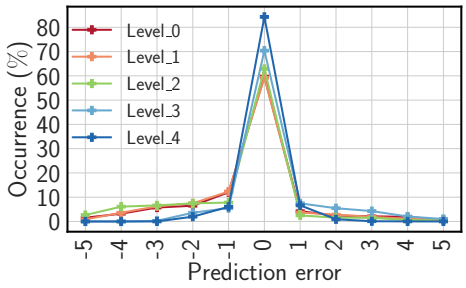
B. Prediction accuracy across simulation timesteps

Considering the nature of scientific applications that evolves with the simulation timesteps, the most important question we need to address in order for our model to be adopted for production is whether our model can be trained on early timesteps and applied to future ones. Therefore, for a certain application dataset, we train on the first half of timesteps and test the model performance on the second half. In Fig 9, we present the distribution of D-MGARD prediction error on the WarpX laser-driven electron acceleration application. We trained the model on the first 256 timesteps of J_x field and evaluated the prediction accuracy on J_x, B_x as well as E_x. As shown in

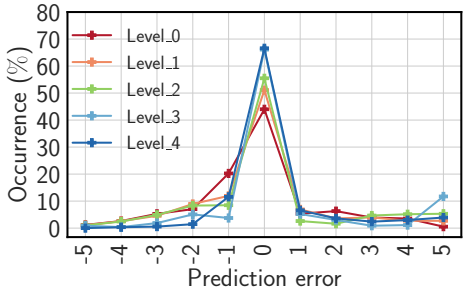


(a) J_x (b) B_x (c) E_x

Fig. 9: Prediction error distribution of D-MGARD on WarpX laser-driven application. The D-MGARD is trained on the first 256 timesteps of J_x field, thus the prediction error values of J_x are collected on later 256 timesteps, whereas the prediction error values of B_x and E_x are collected on total 512 timesteps.



(a) D_u



(b) D_v

Fig. 10: Prediction error distribution of D-MGARD on Gray-Scott application. The D-MGARD is trained on the first 256 timesteps of D_u field, thus the prediction error values of D_u are collected on later 256 timesteps, whereas the prediction error values of D_v are collected on 512 timesteps.

Fig 9a, the absolute prediction error generally ranges between -5 and 5 where positive error indicates over-estimation and negative error indicates under-estimation. For J_x, more than 60% predictions are made without error for levels 1 - 4, with an additional 20% prediction resulting in at most prediction error by one bit-plane. Similar performance holds for B_x and E_x as well, as shown in Figures 9b and 9c that the majority predictions are made correctly. Furthermore, the prediction error decreases from level 0 to level 4 (more predictions are without error), indicating that the numbers of bit-planes of retrieval on lower coefficient levels are better captured.

In Fig 10, we show the prediction error distribution of D-MGARD on Gray-Scott application. Similarly, D-MGARD performs well on lower coefficient levels to capture the number of bit-planes of retrieval that more than 60% of predictions are made without error.

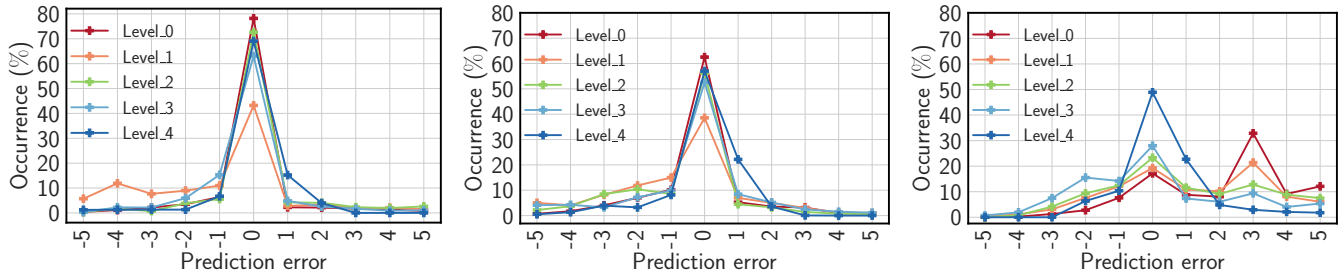
C. Prediction accuracy across simulation resolutions

In this subsection, we aim to answer the following question: can D-MGARD model be applied across simulation resolutions? As running compression experiments and collecting training data on full-resolution data can be resource demanding, it can be beneficial if users can train D-MGARD on low-resolution data and apply it on high-resolution data of the same application. Hereby, we train the model on the J_x field of WarpX application with the resolution of 64^3 and test on the resolutions of 128^3 and 256^3. We demonstrate the distribution of prediction error in Fig. 11. As shown in the figure, D-MGARD model performs well when being trained on the resolution of 64^3 and tested on the resolutions of 64^3 and 128^3. However when being tested on the resolution of 256^3, the prediction accuracy drops significantly. The main reason for such behavior is that more local features manifest in the higher-resolution data, which causes the change of data characteristics and the deviation of MGARD performance, which is hard for D-MGARD to capture.

Despite that D-MGARD model does not work well when the resolution of testing data deviates too much from the resolution of training data, we note that the 80% of predictions on level 4 still achieve error by at most one bit-plane.

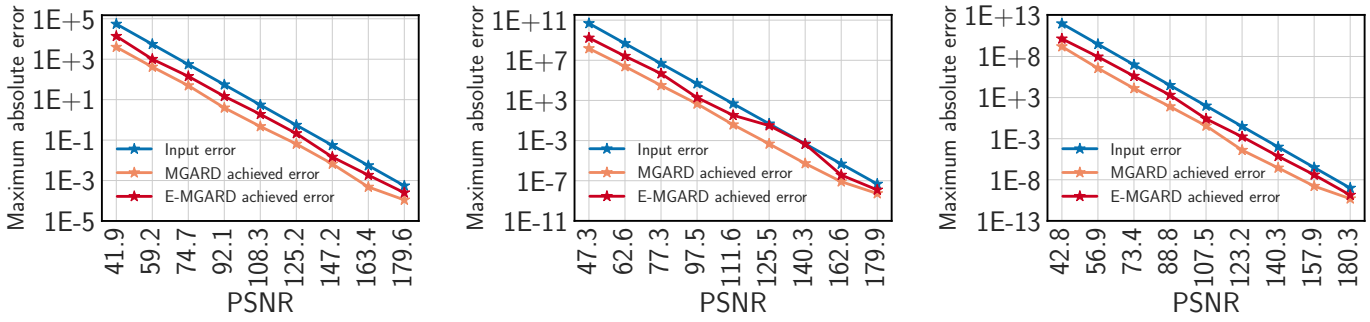
D. Achieved maximum error against original MGARD

In this subsection, we demonstrate the achieved maximum error by E-MGARD. As we discussed previously, E-MGARD improves the error control by predicting the maximum absolute error of reconstruction data with improved mapping from coefficient level error to reconstruction data error. In Fig. 12, we show the achieved maximum error by E-MGARD across PSNR, compared to the maximum error achieved by original MGARD as well as user-requested absolute error bound. It can be shown that the maximum absolute error values achieved by E-MGARD lie closer to the user-requested error, thus providing a better error control.



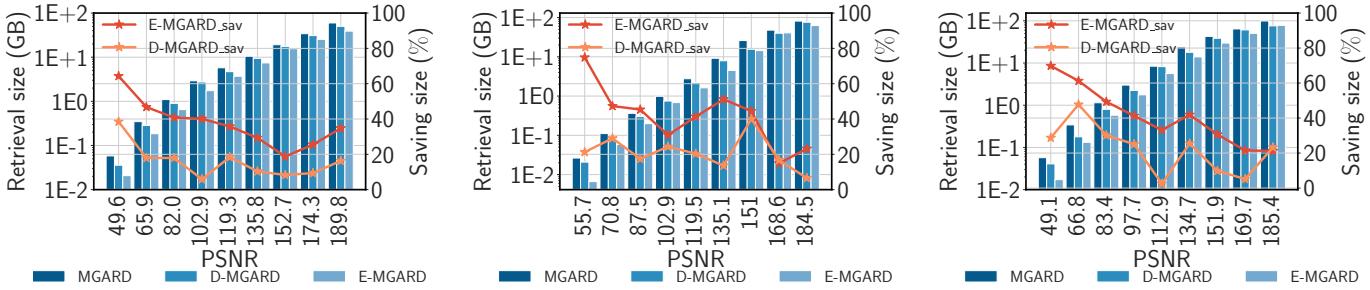
(a) Resolution of 64^3. (b) Resolution of 128^3. (c) Resolution of 256^3.

Fig. 11: Prediction error distribution of D-MGARD across data resolutions. The D-MGARD is trained on the resolution of 64^3 and tested on 128^3 and 256^3. The prediction error values are based on J_x field of WarpX application.



(a) B_x (b) E_x (c) J_x

Fig. 12: E-MGARD achieved maximum absolute error as compared to original MAGRD as well as input error bound. The values are based on WarpX application at timestep t=32. The PSNR values are collected under original MGARD error control.



(a) B_x (b) E_x (c) J_x

Fig. 13: The total retrieval size of D-MGARD and E-MGARD compared to original MGARD across 512 timesteps. The values of retrieval sizes are based on the WarpX application, whereas the PSNR values are collected under original MGARD. It can be shown that D-MGARD typically reduces the retrieval data size between 5% and 40%, whereas E-MGARD reduces the retrieval data size between 20% to 80%.

E. Retrieval size against original MGARD

In Fig 13, we demonstrate the total retrieval size incurred by D-MGARD and E-MGARD as compared to original MGARD. The retrieval sizes are accumulated across total 512 timesteps. It is clear that E-MGARD achieves the least retrieval size and thus the lowest I/O overhead.

We define the percentage of saved retrieval size in the following equation:

$$Sav = |D_{mgard} - D_{new}|/D_{mgard} \quad (8)$$

where D_{new} represents the retrieval size incurred by either D-MGARD or E-MGARD. As shown in the figure (red and orange curves), D-MGARD reduces the retrieval size between 5% and 40%, whereas E-MGARD achieves the saving between 20% and 80%. It can be observed that E-MGARD typically achieves highest saving percentage at small PSNR values, indicating that it holds stronger advantages over D-MGARD and original MGARD when I/O is very limited and users are asking for a low resolution of data being transmitted.

We would also like to point out that, for most evaluation

cases, E-MGARD would achieve a decompression error lower than the input error, although there is no theoretical bound. The reasons are mainly two-fold: 1) The coefficient level reconstruction error is incurred by quantifying the coefficient on each level. This error is further used to estimate the compression error, which guides the selection of bit planes. In reality, the coefficient level reconstruction error with different signs can be canceled with each other. However, there is a lack of theoretical analysis of the signs, and MGARD treats it as there is no error cancellation. Thus, the coefficient level reconstruction error is over-estimated. 2) The predicted compression error will be lower than the input error bound, which is the outcome of MGARD's greedy search.

As E-MGARD only tunes the values of mapping constant C, the second issue can be improved with more granularity. Therefore, the difference between the predicted compression error and coefficient level reconstruction error will be minimized. Nevertheless, the first issue is beyond the scope of our work as our model depends on the collected quantization error to predict compression error as well. Therefore, even with the optimized mapping constants, the numbers of bit-planes to be fetched are still generally more than required, and the compression error of E-MGARD would still be lower than the input.

It is still possible that the E-MGARD achieves a larger compression error than the input, meaning that the error bound is not respected. Cases like these will happen under three conditions: 1) The error bound is extremely small (1E-9, 2E-9), so the mapping constants are highly impactful. 2) The mapping constant is significantly underestimated, leading to the underestimation of predicted compression error and the number of bit-plane. 3) The coefficient level reconstruction error is not over-estimated, meaning that there is no error cancellation across the coefficient levels, which is extremely unlikely for a real dataset.

In this section, we have demonstrated the prediction accuracy of D-MGARD on the number of bit-planes of retrieval across simulation timesteps, data resolution as well as data fields. The prediction error is very low for most cases as the majority of prediction error is by one bit-plane. However, we still notice large prediction errors occurring despite low probability. We think the prediction error is caused by feature selection and engineering. Given the lack of understanding between the number of bit-planes and the achieved maximum error, it remains a challenge to choose a set of intuitive features for prediction. Such a challenge motivates us to further explore more advanced DNN models in the future. On the other hand, we have demonstrated the effectiveness of E-MGARD in optimizing the error control, which results in a more precise maximum error of reconstructed data. While E-MGARD is proven to be beneficial, Fig.13 shows that the most percentage of saving occurs at low PSNR scenarios, indicating imbalanced performance across input error bounds.

Furthermore, Our proposed work is designed and implemented in conjunction with MGARD, by replacing some of the error control functionality within the original MGARD. For example, D-MGARD replaces the error estimator as well as the bit-plane retriever by directly predicting the number of bit-planes of retrieval for each coefficient level based on user-requested maximum error. Then, the size interpreter in MGARD can calculate the retrieval size as well as the precise segment to fetch the data. E-MGARD only bypasses the error estimator by producing a more precise estimation of the achieved maximum data reconstruction error. Then the MGARD can calculate the number of bit-planes of retrieval using the original greedy-based iterative method. Both models will be deployed to work inside the progressive retrieval framework, provided they are previously trained on each application dataset. At this point, both models should be trained offline beforehand. The trained model can be used for inference with the specification of the model parameters through the high-level MGARD decompress API. Currently, the design and deployment of D-MGARD and E-MGARD are separated, and it depends on users to choose between two approaches as we discussed in Section III-E. We would also like to further investigate the combination of two models in the future.

V. Related Work

A. Error-bounded lossy compression

Error-bounded lossy compression techniques have been proven to be effective in reducing data volume while satisfying the accuracy requirements of users. Compared to their lossless counterparts, lossy compression techniques hold the advantage of being able to reduce the data volume considerably. Most recent error-bounded lossy compression techniques can be divided into prediction-based and transform-based approaches, depending on how they remove data redundancies to achieve reduction. Prediction-based techniques generally try to represent data values with a prediction method. For example, motivated by the reduction potential of spline functions [28], [29], ISABELA [15] is designed to compress spatial-temporal scientific data through pre-sorting and B-spline prediction, which makes the random input data more compressible. However, the sorting operation loses locality information, and therefore an extra index needs to be stored which hurts the performance of reduction. Another example is SZ [17], [30], [31], a curve-fitting-based lossy compression algorithm designed by Di *et al.*. It adopts multiple curve-fitting schemes for prediction. Then the curve-fit data points go through linear quantization and Huffman encoding, while the curve-missed data points are stored with binary representation. On the other hand, transform-based techniques usually perform data transforms to convert the original data into a set of de-correlated coefficients. For example, ZFP [16] adopts a block-wise non-orthogonal transform and a customized embedded encoding scheme for compression. The non-orthogonal transform removes the data redundancy within each data block while variable-length embedded encoding compress transform coefficients one bit-plane at a time. Due to the block-wise compression design, it offers random access and truncation capability to the compressed data stream. Austin *et al.* [32] use a Tucker decomposition based method to compress extreme-scale data in parallel.

The results show that the parallel decomposition can achieve high compression ratios and be easily scaled to high core counts. Among numerous lossy compression techniques, SZ and ZFP are demonstrated to be the two best error-bounded lossy compressors [33].

B. Progressive retrieval for HPC dataset

Due to the growing resolution and fidelity of HPC applications, progressive retrieval frameworks emerged to address the imbalance between compute and I/O. Typically, the aim of progressive retrieval frameworks is to store a full-resolution dataset across storage hierarchy so that users can trade-off between data accuracy and I/O as well as analysis speed. Canopus [34] is a progressive data management scheme designed to store and analyze extreme-scale scientific data. It co-designs data decimation, compression and storage, mapping data to different storage tiers with different latency levels so that users can decide the trade-off between analysis speed and data accuracy, and choose to work on lower-accuracy data and augment the accuracy when needed. Hoang [21] proposed a novel way of compressing large-scale particle data that improves the trade-off between data quality and memory footprint. It developed new tree-based data partition and traversal schemes to support progressiveness, random-access as well as error-bounded decoding. MGARD [19], [20] is a recently developed transform-based lossy compressor that combines the feature of multi-level decomposition and progressive retrieval. During compression, it decomposes the original data into orthogonal coefficient levels and encodes coefficient levels using bit-plane encoding. The coefficient levels can be then mapped to storage tiers with different capacities and latency levels. During decompression, bit-planes across coefficient levels are selected based on error bounds via greedy search to achieve minimal data retrieval. It provides significant performance advantages in the scenarios of limited I/O bandwidth. ZFP also begins to support progressive decoding [35] by enabling truncation on the compressed data stream. However, the functionality is not yet available.

VI. Conclusion

In this paper, we propose a DNN-based progressive retrieval framework to reduce the I/O by minimizing the amount of data to be fetched. To this end, we design two prediction models to estimate the number of bit-planes of retrieval (D-MGARD) and improve the error control (E-MGARD) under user-requested error bounds. We evaluate our proposed DNN-based progressive retrieval framework on two scientific application datasets. We demonstrate that our framework holds significant advantages over the traditional approach. By evaluating against the original theory-based progressive data retrieval framework, our solution is shown to access significantly less data (between 5% and 40% with D-MGARD, between 20% and 80% with E-MGARD). Our approach brings opportunities for improving scientific applications running on computing clusters including HPC systems as well as grid computing environments.

Acknowledgment

The authors would like to acknowledge the support from the U.S. Department of Energy ECP and SIRIUS, as well as U.S. National Science Foundation under Grant No. CCF-2144403, CCF-1812861. The work performed at the Temple University is partially supported by U.S. National Science Foundation under Grant No. CCF-1813081 and CCF-2134203. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1621–1631, 2012.

[2] T. Wang, S. Oral, Y. Wang, B. Settlemyer, S. Atchley, and W. Yu, "Burstmem: A high-performance burst buffer system for scientific applications," in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 71–79.

[3] T. Wang, S. Oral, M. Pritchard, B. Wang, and W. Yu, "Trio: Burst buffer based i/o orchestration," in *2015 IEEE International Conference on Cluster Computing*. IEEE, 2015, pp. 194–203.

[4] S. Herbein, D. H. Ahn, D. Lipari, T. R. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer, "Scalable i/o-aware job scheduling for burst buffer enabled hpc clusters," in *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2016, pp. 69–80.

[5] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *012 ieee 28th symposium on mass storage systems and technologies (MSST 12)*. IEEE, 2012, pp. 1–11.

[6] A. Gainaru, G. Aupy, A. Benoit, F. Cappello, Y. Robert, and M. Snir, "Scheduling the i/o of hpc applications under congestion," in *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS 15)*. IEEE, 2015, pp. 1013–1022.

[7] S. Thapaliya, P. Bangalore, J. Lofstead, K. Mohror, and A. Moody, "Managing i/o interference in a shared burst buffer system," in *2016 45th International Conference on Parallel Processing (ICPP)*. IEEE, 2016, pp. 416–425.

[8] Q. Zheng, K. Ren, G. Gibson, B. W. Settlemyer, and G. Grider, "DeltaFS: Exascale file systems scale better without dedicated servers," in *Proceedings of the 10th Parallel Data Storage Workshop (PDSW)*, 2015, pp. 1–6.

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

[10] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen, "An image-based approach to extreme scale in situ visualization and analysis," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 424–434.

[11] U. Ayachit, A. Bauer, E. P. Duque, G. Eisenhauer, N. Ferrier, J. Gu, K. E. Jansen, B. Loring, Z. Lukić, S. Menon *et al.*, "Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2016, p. 79.

[12] D. Huang, Q. Liu, S. Klasky, J. Wang, J. Y. Choi, J. Logan, and N. Podhorszki, "Harnessing data movement in virtual clusters for in-situ execution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 615–629, 2018.

[13] M. Burtscher and P. Ratanaworabhan, "Fpc: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2008.

[14] 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.

[15] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data," in *European Conference on Parallel Processing (EURO-PAR)*. Springer, 2011, pp. 366–379.

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

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

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

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

[20] X. Liang, Q. Gong, J. Chen, B. Whitney, L. Wan, Q. Liu, D. Pugmire, R. Archibald, N. Podhorszki, and S. Klasky, "Error-controlled, progressive, and adaptable retrieval of scientific data with multilevel decomposition," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476179

[21] D. Hoang, H. Bhatia, P. Lindstrom, and V. Pascucci, "High-Quality and Low-Memory-Footprint Progressive Decoding of Large-Scale Particle Data," in *2021 IEEE 11th Symposium on Large Data Analysis and Visualization (LDAV)*. New Orleans, LA, USA: IEEE, Oct. 2021, pp. 32–42. [Online]. Available: https://ieeexplore.ieee.org/document/9623245/

[22] K. Demirel, A. Şahin, and E. Albey, "Ensemble Learning based on Regressor Chains: A Case on Quality Prediction:," in *Proceedings of the 8th International Conference on Data Science, Technology and Applications*. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2019, pp. 267–274.

[23] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[24] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[25] (2021) Summit - oak ridge leadership computing facility. [Online]. Available: https://www.olcf.ornl.gov/summit/

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

[27] A. Myers, A. Almgren, L. D. Amorim, J. Bell, L. Fedeli, L. Ge, K. Gott, D. P. Grote, M. Hogan, A. Huebl *et al.*, "Porting warpx to gpu-accelerated platforms," *Parallel Computing*, vol. 108, p. 102833, 2021.

[28] S. Wold, "Spline functions in data analysis," *Technometrics*, vol. 16, no. 1, pp. 1–11, 1974.

[29] X. He and P. Shi, "Monotone b-spline smoothing," *Journal of the American statistical Association*, vol. 93, no. 442, pp. 643–650, 1998.

[30] 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.

[31] 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.

[32] W. Austin, G. Ballard, and T. G. Kolda, "Parallel tensor compression for large-scale scientific data," in *2016 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 2016, pp. 912–922.

[33] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, and Z. Qiao, "Understanding and modeling lossy compression schemes on hpc scientific data," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS 18)*, 2018, pp. 1–10.

[34] T. Lu, E. Suchyta, D. Pugmire, J. Choi, S. Klasky, Q. Liu, N. Podhorszki, M. Ainsworth, and M. Wolf, "Canopus: A paradigm shift towards elastic extreme-scale data analytics on hpc storage," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017, pp. 58–69.

[35] P. Lindstrom. (2022) zfp 1.0.0 documentation. [Online]. Available: https://zfp.readthedocs.io/en/release1.0.0/directions.html