Error-controlled Progressive Retrieval of Scientific Data under Derivable Quantities of Interest

Xuan Wu
University of Kentucky
Lexington, USA
xuan.wu@uky.edu

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

Jieyang Chen
University of Alabama at Birmingham
Birmingham, USA
jchen3@uab.edu

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

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

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

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

***Abstract*—The unprecedented amount of scientific data has introduced heavy pressure on the current data storage and transmission systems. Progressive compression has been proposed to mitigate this problem, which offers data access with on-demand precision. However, existing approaches only consider precision control on primary data, leaving uncertainties on the quantities of interest (QoIs) derived from it. In this work, we present a progressive data retrieval framework with guaranteed error control on derivable QoIs. Our contributions are three-fold. (1) We carefully derive the theories to strictly control QoI errors during progressive retrieval. Our theory is generic and can be applied to any QoIs that can be composited by the basis of derivable QoIs proved in the paper. (2) We design and develop a generic progressive retrieval framework based on the proposed theories, and optimize it by exploring feasible progressive representations. (3) We evaluate our framework using five real-world datasets with a diverse set of QoIs. Experiments demonstrate that our framework can faithfully respect any user-specified QoI error bounds in the evaluated applications. This leads to over $2.02\times$ performance gain in data transfer tasks compared to transferring the primary data while guaranteeing a QoI error that is less than 1E-5.**

***Index Terms*—High-performance computing, data compression, progressive retrieval, scientific data, error control**

I. Introduction

The arrival of the first generation of exascale machines and the continuous upgrading of experimental and observational facilities have presented a huge strain on storage, I/O, and networking due to unprecedented data volume and velocities. Because of the limited spacing at high-end parallel file systems (PFS), most of these data must be moved to lower-tier storages, such as tapes, right after generation. Future analyses, which require retrieving data from a central repository and moving across wide area networks, must consider the cost of data retrieval and movement. Recently, lossy compression methods [1]–[5] have been developed to tackle the I/O and storage bottleneck as they demonstrate greater compressibility than lossless compressors on floating-point scientific data. Since most simulation and experimental devices inherently involve uncertainty and variability, data can be reduced, provided the loss of accuracy is under prescribed bounds.

The current leading error-controlled lossy scientific data compressors, including MGARD [5]–[7], SZ [2], [8], [9], and ZFP [4], to name a few, carry mathematically proved theories, which guarantee that errors in the reconstructed data to stay below user-prescribed error bounds. However, most of these compressors only allow to prescribe a single error bound, assuming a "one-size" accuracy will fit all subsequent data explorations. In contrast, the reduced data may be used for various downstream analyses that are either known or unknown upon data generation. To ensure the fidelity of scientific discovery, users have too often conservatively chosen error bounds that cater to the most pessimistic use cases. Such over-preservation during data compression, unfortunately, will lead to great expense in data retrieval when faced with a diversity of analyses and use-cases of varying requirements on data fidelity.

Data refactoring and progressive retrieval pose a potential solution to combat the diverse requests on data fidelity at lower data movement cost [10]–[16]. Notably, MGARD and ZFP have separately developed the progressive reconstruction feature [15], based on multi-level methods (for progressive resolution with MGARD) and bit-plane encoding (for progressive precision with both compressors). They allow data to be archived at nearly full accuracy and retrieved on an as-needed basis, often at

* Corresponding author: Xin Liang, Department of Computer Science, University of Kentucky, Lexington, KY 40506.

SC24, November 17-22, 2024, Atlanta, Georgia, USA
979-8-3503-5291-7/24/$31.00 ©2024 IEEE

reduced resolution and/or precision, for faster data transmission and computations. The progressive reconstruction also allows data to be incrementally recomposed to higher fidelity when more data components become available.

Despite the potential advantages of progressive retrieval, the gap between the errors in primary data and derived quantities of interest (QoIs) should not be overlooked [17]. Obeying the error bounds for QoIs is challenging as the relation between the primary data and the QoI can be highly nonlinear [18]. Blindly refining the approximation of the data during progressive retrieval leads to under- or over-estimations, which may not produce correct outcomes in the downstream analyses. Motivated by the disconnection in error control objectives, several works have recently started to explore the preservation of QoIs for a few specific analytic tasks [19]–[22] or region-of-interest (RoI) during data compression [23]. Nevertheless, directly applying existing QoI-preservation techniques used for compression to progressive retrieval is non-trivial for several reasons:

- QoI-preserving compressors that can handle a broad range of analytic functions are required to prescribe point-wise varied error bounds, whereas the compressors capable of performing progressive retrieval are based on bitplane encoding or multi-level techniques, featuring globally uniform error bounds.
- The original values of QoIs are a prerequisite for most QoI-preserving compressors, which are usually computed prior to data compression. Such ground truth values, however, are unattainable with progressive retrieval unless data is recomposed to full fidelity.
- These aforementioned difficulties can be further exacerbated when retrieval is targeted at preserving the multivariate and composite QoIs that involve multiple data fields and analytic functions.

In this paper, we propose a generic framework to progressively retrieve scientific data with strict error control on *derivable QoIs*. We define *derivable QoIs* as downstream quantities that can be explicitly composited by a set of basis functions, including polynomials, square root, and radical functions, along with their combinations through additive, multiplicative, divisive operations, and other functional compositions (see Definition 2 and 3). The combination of the above basis function and operations will cover a broad range of physical properties, such as kinetic energy, momentum, and magnitude of velocity, that are commonly used in real-world scientific applications. Since the derived errors in QoIs vary across data space, we target the L^∞ bound as it measures the extreme case, and the preservation of L^∞ error will automatically ensure the satisfaction of the point-wise error bound. In addition, we propose theories to estimate the errors of derivable QoIs based on the errors of primary data, as the ground truth of QoI values cannot be obtained during progressive retrieval, and use the proposed estimators to guide the process of data refinement. We further explore and investigate the efficiency of different progressive methods using our framework. The key contributions are summarized as follows.

- We carefully derive the theory to enable QoI error estimation on progressive representations, which can incrementally refine the data approximation until the estimated errors in QoIs are derived from the recomposed data to satisfy user-prescribed bounds. This theory can be generalized to arbitrary error-controlled progressive compressors and offers error control to a broad range of derivable QoIs provided that they can be composed by the basis functions and operations covered in this paper.
- We develop a generic progressive retrieval framework capable of QoI error control during progressive retrieval based on our theory. We further integrate three general progressive methods into our framework and explore their efficiency. To this end, we revise the decomposition method in PMGARD [15] to enable stable and efficient QoI error control.
- We perform a comprehensive evaluation using scientific data from four real-world applications and one case study with a computational fluid dynamics (CFD) code from Generic Electric (GE). Specifically, we evaluate our framework using different progressive representations and a diverse set of QoIs. Experimental results demonstrate that the proposed method provides strict error control in known QoIs, and this yields over $2.02\times$ performance in data retrieval from remote storage systems via Globus.

The rest of the paper is organized as follows. In Section II, we discuss the related works. In Section III, we formulate the research problem and present an overview of the compression framework. In Section IV, we introduce the theories to enable QoI error control in progressive formats, which serves as the foundation for the proposed work. In Section V, we describe the implementation of the proposed framework along with the optimizations. Section VI demonstrates the evaluation results with real-world datasets and a case study with GE. In Section VII, we conclude the research with a vision for future work.

II. Related works

In this section, we review the lossy compression and progressive retrieval work derived from the former in the context of scientific data defined on Cartesian grids. For works on tree structure, adaptive meshes, and unstructured data, we refer the readers to [10], [24]–[26].

Data compression is a direct way to mitigate the I/O and storage pressure, which has been studied in the scientific computing community for years. Traditional lossless compression techniques [27]–[29] achieve only a modest reduction for floating-point scientific data [30], which falls far from the desires of exascale computing. Conventional lossy compressors, such as JPEG/JPEG2000 [11], [12], while ubiquitous in image transmission, have rarely been used by scientific datasets as they cannot bound errors incurred by compression. Therefore, we limit our review to error-controlled scientific compressors.

The most widely reported error-controlled lossy compressors fall into two broad categories: prediction-based and transform-

based. Prediction-based compressors such as ISABELA [1], SZ [2], [8], [31]–[33], and QoZ [34] rely on varied predictors, such as spline interpolation or polynomial fitting, to decorrelate the data, whereas transform-based ones such as ZFP [4] and TTHRESH [35] employ existing or customized transforms to eliminate redundancy. Coefficients after decorrelation/transform may be quantized into integers and then losslessly compressed through entropy or embedded encoding approaches to reduce the size. Notably, scientific lossy compressors carry mathematical theories for quantization and encoding, which ensure the maximal error between the original and reconstructed data is less than a user-specified error bound. Recently, several compressors even advanced the error control onto downstream QoIs that are derived from the reconstructed data [7], [18]–[22], [36].

MGARD [5]–[7] derives a *norm* based on the finite element analysis and wavelet theories, applying it to tighter the error bounds such that the most pessimistic QoI cases can be satisfied. Due to the complexity of mathematical derivation, MGARD's current QoI-control theory is only applicable to linear QoIs, which limits its use cases. A variation of SZ has also been proposed in [21], which relies on a pre-evaluation of target QoIs and derivating point-wise compression error bounds such that QoI values computed from the reconstructed data will satisfy user-prescribed error bounds. The post-processing-based QoI-preserved techniques [18], [36] iteratively update the reduced approximation until the derived QoI errors stay below prescribed bounds. Nevertheless, similar to the QoI preservation work with SZ, the post-processing technique requires knowing the original QoI values and is only applicable to univariate QoIs. Several additional compression methods have been developed to reduce the data while preserving topological features such as critical points [19], [20] and contour trees [22], but they do not generalize to other QoIs.

The most prominent downside of lossy compressors is that the "one-size-fits-all" error prescription strategy is prone to underestimating or wasting resources when faced with diverse post-processing tasks. In contrast, progressive compression and retrieval allow for dynamic adjustment of the transmitted data size based on requested fidelity and support incremental recomposition to obtain finer data representations without starting from scratch. The progressive approaches can be generally categorized into progression in resolution and progression in precision. The most well-known approaches in the former category include Fourier and discrete cosine transform [37], wavelets pyramid [38], [39], multi-level decomposition [6], and rank decomposition [35], [40], where data representations in coarser resolution are obtained by retrieving only a subset of coefficients. In comparison, progression in precision is often achieved through encoding the bit planes [4], [15], [39], or iteratively compressing the residues with progressively decreased error bounds [16]. With bit-plane encoding, the precision-based progressive retrieval will be performed among all coefficients, starting from the most to least significant bit. With progressively decreased error bounds, the compression procedure will generate multiple snapshots with different precision for retrieval. Progression in precision can provide more fine-grained retrieval compared to progression in resolution. We also notice that some progressive techniques, such as the PMGARD [15], support progression in both categories. Specifically, PMGARD combines the orthogonal decomposition method in MGARD with bitplane encoding to provide guaranteed error control on primary data.

Despite the potential to fulfill data requests of arbitrary precision, none of the existing progressive compression techniques provide error control on downstream QoIs. In this work, we bridge the gap by developing a generic framework to determine the proper amount of data to retrieve in progressive formats to meet user-specified QoI tolerances, which is expected to significantly reduce the retrieval size and thus improve data movement performance. To the best of our knowledge, this is the first attempt to tackle QoI preservation during progressive retrieval.

TABLE I: Notations

Symbol	Description
n_e	Number of data points.
n_v	Number of variables.
n_q	Number of target QoIs.
n_s	Number of progressive segments.
τ	Error tolerance on QoIs.
x, x'	Single scalar values.
s	Data fragments produced by progressive compression.
ϵ	Error bound on the primary data.
ξ	Real error in the primary data (upper bounded by ϵ).
$\mathbf{x}, \mathbf{x}', \boldsymbol{\epsilon}$	Vectors of x, x', ϵ in multivariate cases.
x_i, x_i', ϵ_i	The i-th element in $\mathbf{x}, \mathbf{x}', \boldsymbol{\epsilon}$.
f	Univariate QoI that applies to data on a single field.
g	Multivariate QoI that applies to data on multiple fields.
$\Delta(f, x, \epsilon)$	Upper bound of QoI error in f at x with error bound ϵ.
$\Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon})$	Upper bound of QoI error in g at $\boldsymbol{x}$ with error bound $\boldsymbol{\epsilon}$.
$\lvert\cdot\rvert$	Operator of getting absolute value.
$\{\cdot_i\}$	An array of the referred element.

III. Overview

We formulate our research problem in this section, followed by an overview of the proposed framework. The notations used in the paper are summarized in Table I.

A. Problem formulation

Our progressive retrieval framework is designed to extract only the "necessary" amount of progressive fragments and guarantee that the user-prescribed error tolerance in QoIs derived from the reconstructed data is met. The capability of estimating the errors in QoIs is crucial for the trustability of the reconstructed data, and it can accelerate the process of reading data from low-tier storage or remote central repository by minimizing the data volume. Below, we define the requested functionalities in progressive compressors and the *derivable* QoIs targeted in this paper.

Definition 1: An error-controlled progressive compression method shall be able to (1) refactor the original data $\{x_1', \cdots, x_{n_e}'\}$ into progressive fragments $\{s_1, \cdots, s_{n_s}\}$ for archiving, where n_e is the number of original data points and

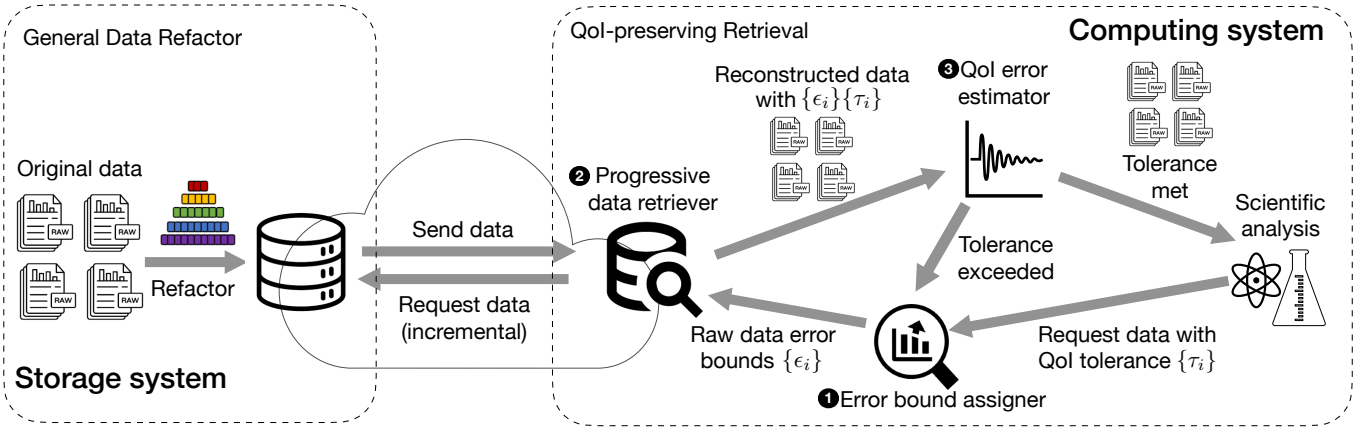

Fig. 1: Workflow of the proposed QoI-preserving framework with three key modules. We assume that data is refactored and stored in storage systems when generated, and our framework is able to progressively retrieve data from storage while guaranteeing user-specified QoI error bounds. This is extremely useful when data movement becomes the performance bottleneck, which is the case when data is located in secondary or remote storage systems.

n_s is the number of progressive segments; (2) reconstruct data $\{x_1, \cdots, x_{n_e}\}$ from $\{s_1, \cdots, s_j\}$ such that $\max_i |x'_i - x_i| < \epsilon_j$, where ϵ_j is the prescribed error bound when recomposing data using the first j segments.

Definition 2: A univariate QoI is a univariate function $f : \mathbb{R} \to \mathbb{R}$ that maps a scalar value to a quantity. The derivable univariate QoIs include a family of QoIs that can be formulated by the set of basis functions defined in Table II.

Definition 3: A multivariate QoI is a multivariate function $g : \mathbb{R}^n \to \mathbb{R}$ that maps a vector to a quantity. The derivable multivariate QoIs include a family of QoIs that can be composited using the derivable univariate QoIs and operations of derivable multivariate QoIs defined in Table II.

TABLE II: Bases of Derivable QoIs

Name	Category	Formula	Example in GE
Polynomials	univariate	$f(x) = \sum a_i x^i$	(1) (5), (6)
Square root	univariate	$f(x) = \sqrt{x}$	(1), (3), (6)
Radical	univariate	$f(x) = 1/(x + c)$	(6)
Addition	multivariate	$g(x_1, \cdots, x_n) = \sum x_i$	(1)
Multiplication	multivariate	$g(x_1, \cdots, x_n) = \Pi x_i$	(5), (6)
Division	multivariate	$g(x_1, x_2) = x_1/x_2$	(2), (4)
Composition	both	$(f_1 \circ f_2)(x) = f_1(f_2(x))$	(1) – (6)

Despite the fact that there are only seven families of QoIs listed above, their compositions cover a broad range of functions commonly used in scientific studies. Below, we showcase how to break down the user-requested QoIs into the basis of derivable QoIs using the non-proprietary data generated by a Computational Fluid Dynamics (CFD) code from GE. The CFD simulation produces velocity V_x, V_y, V_z, pressure P, and density D on a set of unstructured meshes. We linearize the data into a one-dimensional array, choosing six QoIs used in the posthoc analyses with the detailed equations listed below, as they will be used in our experimental evaluation later on:

$$V_{total} = \sqrt{\boldsymbol{V_x^2} + \boldsymbol{V_y^2} + \boldsymbol{V_z^2}}, \tag{1}$$

$$T = \frac{\boldsymbol{P}}{\boldsymbol{D} * R}, \tag{2}$$

$$C = \sqrt{\gamma * R * \boldsymbol{T}}, \tag{3}$$

$$Mach = \frac{\boldsymbol{V_{total}}}{\boldsymbol{C}}, \tag{4}$$

$$PT = \boldsymbol{P}(1 + \frac{\gamma}{2} * \boldsymbol{Mach} * \boldsymbol{Mach})^{mi}, \tag{5}$$

$$\mu = \mu_r (\frac{\boldsymbol{T}}{T_r})^{1.5} \frac{T_r + S}{\boldsymbol{T} + S}. \tag{6}$$

Here, $R = 287.1$, $\gamma = 1.4$, $mi = 3.5$, $\mu_r = 1.716e-5$, $T_r = 273.15$, and $S = 110.4$ are different constant values, and the input and intermediate variables are bolded. Using PT defined in Equation (5) as an example, the formula can be decomposed into the multiplication of P and $(1 + \frac{\gamma}{2} * Mach * Mach)^{mi}$, where the latter is a composition of the square root function and a polynomial of $Mach$.

B. Design overview

We illustrate the workflow of the proposed QoI-preserving progressive data retrieval framework in Fig. 1. Our key contribution lies in the retrieval procedure, which iteratively refines the reduced approximation till the estimate errors in QoIs drop below user-prescribed bounds. Specifically, we develop (1) a QoI error estimator that provides an upper bound of the errors in QoI given the reconstructed data and its L^∞ error bounds, and (2) a primary data (PD) error assigner which estimates and prescribes the bounds used for the next round of data retrieval. The progressive compression procedure can be performed using the existing error-controlled progressive compression frameworks [4], [15], [16], where data is refactored and compressed into multi-precision fragments.

The overall data retrieval pipeline can be summarized as follows. Firstly, an analytic task requests a set of QoIs and the desired error tolerance. This request is processed by the PD error-bound assigner (module 1), which gauges the error bounds on each primary data field used by the first round of retrieval. Such error bounds will be sent to a progressive retriever (module 2), which extracts progressive segments from the most to the least significant until the errors in the errors in the reconstructed data reach below the requested bounds. Data will be incrementally recomposed using the newly arrived progressive segments and then fed into the QoI error estimator, along with the error bounds used during retrieval, to estimate the upper bounds of QoI errors under the current data representations (module 3). If the estimated QoI errors are less than the requested tolerances, the data is provisioned for the analyses; otherwise, the current data representations, along with the derived QoI errors, will be forwarded to the PD error bound assigner to estimate the error bounds used in the next round of data retrieval. The pipeline repeats these steps till the targeted QoI error bounds are reached, or a full-fidelity data representation is retrieved. Due to their progressive nature, only incremental portions of the data need to be retrieved in the later requests, which promises high efficiency in managing the data movement from storage systems to applications.

C. Quality assessment

We leverage the widely used rate-distortion curves [15], [41], [42] to evaluate the efficiency of our approach. The X-axis in the curve is bitrate, which represents the average number of bits in the compressed format. It is analogous to the compression ratio in single-snapshot compression, and can be computed by the retrieved data size divided by the number of elements. We use relative QoI errors as our distortion metric for the Y-axis, which is computed by the maximal absolute error of QoI divided by its respective value range. An easy way to compare multiple rate-distortion curves is to fix either the X value or Y value: in the former case, one can compare the errors of different approaches based on the same retrieved data size; in the latter case, one can compare the size of retrieved data under the same quality.

IV. Theoretical foundation

In this section, we introduce the theoretical foundation of the proposed work. The data retrieval is designed to meet the error bounds prescribed on QoIs. Below, we describe how to estimate the errors in QoI using the reconstructed data and its L^∞ error bound during data retrieval. Having such error estimation is critical as we need to iteratively update and examine the QoI errors during data retrieval. Specifically, we derive the upper error bounds for the bases of derivable QoIs (shown in Table II) and discuss their combinations in univariate, multivariate, and composite cases.

The following subsections start with the definition of QoI errors for each case, then theorems and proofs for different types of QoI functions. Please refer to Table I for the notations of the symbols used in our theorems and proofs. Notably, throughout the derivations, we assume the original data x' and the reconstructed data x to satisfy a L^∞ error bound, as will be detailed below.

A. Univariate QoIs

Definition 4: Given a data value x and a L^∞ error bound ϵ used during progressive retrieval, we define $\Delta(f, x, \epsilon)$ as the supremum of QoI error under a univariate QoI $f(x)$: $\Delta(f, x, \epsilon) = \sup_{|x'-x|\leq\epsilon} |f(x') - f(x)|$.

Here, we assume the original value is x' will satisfy the error bound constraint $|x' - x| \leq \epsilon$, then we have $|f(x^*) - f(x)|_\infty \leq \sup_{|x'-x|\leq\epsilon} |f(x') - f(x)|_\infty = \Delta(f, x, \epsilon)$. Note that $\Delta(f, x, \epsilon)$ only relies on the reconstructed data and the error bound used during data retrieval. Below, we present the theorems used for estimating $\Delta(f, x, \epsilon)$ given several univariate QoI functions.

Theorem 1: [Polynomials] An upper bound of $\Delta(f, x, \epsilon)$ for function $f(x) = x^n$ can be written as $\Delta(f, x, \epsilon) \leq \sum_{i=1}^{n} C_n^i |x|^{n-i}\epsilon^i$, where $C_n^i = \frac{n!}{(n-i)!i!}$ is the combination formula.

Proof: $|f(x') - f(x)| = |(x + \xi)^n - x^n| = |\sum_{i=0}^{n} \xi^i x^{(n-i)} - x^n| = |\sum_{i=1}^{n} C_n^i x^{n-i}\xi^i| \leq \sum_{i=1}^{n} C_n^i |x|^{n-i}|\xi^i| \leq \sum_{i=1}^{n} C_n^i |x|^{n-i}\epsilon^i$. ■

Theorem 2: [Square Root] An upper bound for function $f(x) = \sqrt{x}$ can be written as $\Delta(f, x, \epsilon) \leq \epsilon/(\sqrt{\max(x - \epsilon, 0)} + \sqrt{x})$.

Proof: Since the negative $x - \epsilon$ can be replaced by 0, $\sqrt{x'} = \sqrt{x + \xi} \geq \sqrt{\max(0, x - \epsilon)}$. Then $|f(x') - f(x)| = |\sqrt{x + \xi} - \sqrt{x}| = |\xi/(\sqrt{x + \xi} + \sqrt{x})| \leq \epsilon/(\sqrt{\max(x - \epsilon, 0)} + \sqrt{x})$. ■

Theorem 3: [Radical] An upper bound for radical function $f(x) = 1/(x + c)$ can be written as $\Delta(f, x, \epsilon) \leq \epsilon/\{\min(|x + c - \epsilon|, |x + c + \epsilon|)|x + c|\}$, when $x + c \neq 0$ and $\epsilon < |x + c|$.

Proof: $|f(x') - f(x)| = |1/(x + \xi + c) - 1/(x + c)| = |\xi/((x+\xi+c)(x+c))|$. Since $\epsilon < |x+c|$, we have $|x+\xi+c| \geq \min(|x + c - \epsilon|, |x + c + \epsilon|)$. Therefore, $|f(x') - f(x)| \leq \epsilon/\{\min(|x + c - \epsilon|, |x + c + \epsilon|)|x + c|\}$. ■

Note that Theorem 3 does not apply to the case of $\epsilon > |x+c|$, as it may lead to an infinitesimal value of $|x + \xi + c|$, causing the errors in QoI unable to be bounded. Such a case can be avoided by only choosing $\epsilon < |x + c|$ during data retrieval.

B. Multivariate QoIs

Definition 5: Given a vector $\boldsymbol{x} = (x_1, \cdots, x_n)^T$ and a L^∞ error bound vector $\boldsymbol{\epsilon}$ used during data retrieval, we define $\Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon})$ as the supremum of QoI error under a multivariate QoI g: $\Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon}) = \sup_{|\boldsymbol{x'}-\boldsymbol{x}|\leqq\boldsymbol{\epsilon}} |g(\boldsymbol{x'}) - g(\boldsymbol{x})|$, where $\leqq$ indicates that the *less and equal to* relationship is applied for every element in the input vector.

Assume $\xi_i = x'_i - x_i$ and $x'_i \in [x_i - \epsilon_i, x_i + \epsilon_i]$ following the error bound constraints, then we have the following theorems.

Theorem 4: [Addition] An upper bound for weighted summation function $g(\boldsymbol{x}) = \sum_{i=1}^{n} a_i x_i$ is $\Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon}) \leq \sum_{i=1}^{n} |a_i|\epsilon_i$.

Proof: $|g(\boldsymbol{x'}) - g(\boldsymbol{x})| = |\sum_{i=1}^{n} a_i\xi_i| \leq \sum_{i=1}^{n} |a_i||\xi_i| \leq \sum_{i=1}^{n} |a_i|\epsilon_i$. ■

Theorem 5: [Multiplication] An upper bound for multiplication function $g(x_1, x_2) = x_1 x_2$ can be written as $\Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon}) \leq |x_1|\epsilon_2 + |x_2|\epsilon_1 + \epsilon_1\epsilon_2$.

Proof: $|g(\boldsymbol{x'}) - g(\boldsymbol{x})| = |(x_1 + \xi_1)(x_2 + \xi_2) - x_1 x_2| = |x_1\xi_2 + x_2\xi_1 + \xi_1\xi_2| \leq |x_1||\xi_2| + |x_2||\xi_1| + |\xi_1\xi_2| \leq |x_1|\epsilon_2 + |x_2|\epsilon_1 + \epsilon_1\epsilon_2$. ∎

Theorem 6: [Division] An upper bound for division function $g(x_1, x_2) = x_1/x_2$ can be written as $\Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon}) \leq (|x_1|\epsilon_2 + |x_2|\epsilon_1)/\{|x_2| \min(|x_2 - \epsilon_2|, |x_2 + \epsilon_2|)\}$ when $\epsilon < |x_2|$.

Proof: $|g(\boldsymbol{x'}) - g(\boldsymbol{x})| = |(x_1 + \xi_1)/(x_2 + \xi_2) - x_1/x_2| = |(x_2\xi_1 - x_1\xi_2)/\{(x_2 + \xi_2)x_2\}| \leq (|x_2||\xi_1| + |x_1||\xi_2|)/(|x_2 + \xi_2||x_2|) \leq (|x_1|\epsilon_2 + |x_2|\epsilon_1)/\{|x_2| \min(|x_2 - \epsilon_2|, |x_2 + \epsilon_2|)\}$. ∎

C. Composite QoIs

This subsection applies the theories of addition, scalar multiplication, and composition for the QoI functions presented in the previous subsection to broaden the range of QoIs that can be preserved during retrieval. Specifically, we have the following theorems for composite QoIs.

Theorem 7: [Additive] $\Delta(f, x, \epsilon)$ satisfies the additive property $\Delta(f_1 + f_2, x, \epsilon) \leq \Delta(f_1, x, \epsilon) + \Delta(f_2, x, \epsilon)$.

Proof: $\Delta(f_1 + f_2, x, \epsilon) = |f_1(x + \xi) + f_2(x + \xi) - f_1(x) - f_2(x)| \leq |f_1(x+\xi) - f_1(x)| + |f_2(x+\xi) - f_2(x)| = \Delta(f_1, x, \epsilon) + \Delta(f_2, x, \epsilon)$. ∎

Theorem 8: [Multiplicative] $\Delta(f, x, \epsilon)$ satisfies the multiplicative property $\Delta(af, x, \epsilon) = a\Delta(f, x, \epsilon)$ for any constant $a \neq 0$.

Proof: $\Delta(af, x, \epsilon) = |af(x+\xi) - af(x)| = a|f(x+\xi) - f(x)| = a\Delta(f, x, \epsilon)$. ∎

Theorem 9: [Composition] $\Delta(f, x, \epsilon)$ satisfies the composite property $\Delta(f_1 \circ f_2, x, \epsilon) = \Delta(f_1, f_2(x), \Delta(f_2, x, \epsilon))$.

Proof: Denote $y = f_2(x)$ and $\xi' = f_2(x+\xi) - f_2(x)$, then $\xi' \in [f_2(x) - \Delta(f_2, x, \epsilon), f_2(x) + \Delta(f_2, x, \epsilon)]$. Correspondingly, $\Delta(f_1 \circ f_2, x, \epsilon) = |f_1(f_2(x+\xi)) - f_1(f_2(x))| = |f_1(y + \xi') - f_1(y)| = \Delta(f_1, f_2(x), \Delta(f_2, x, \epsilon))$. ∎

Although the above proofs have been conducted on univariate QoIs, the same theorems apply to the additive, multiplicative, and composite operations in multivariate QoIs. Additionally, we derive the error estimators for the composition of univariate QoIs and multivariate QoIs and summarize them in the two lemmas below. We omit the details of the proofs due to limited space. The proofs are similar to the procedure in Theorem 9.

Lemma 1: Denote $f \circ g$ as the composition of a univariate function f and a multivariate function g such that $f \circ g(\boldsymbol{x}) = f(g(\boldsymbol{x}))$. We have $\Delta(f \circ g, \boldsymbol{x}, \boldsymbol{\epsilon}) = \Delta(f, g(\boldsymbol{x}), \Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon}))$.

Lemma 2: Denote $g \circ \{f_1, \cdots, f_n\}$ as an element-wise composition of a multivariate function g and n univariate functions $\{f_1, \cdots, f_n\}$ such that $g \circ \{f_1, \cdots, f_n\}(x_1, \cdots, x_n) = g(f_1(x_1), \cdots, f_n(x_n))$. We have $\Delta(g \circ \{f_1, \cdots, f_n\}, \boldsymbol{x}, \boldsymbol{\epsilon}) = \Delta(g, \boldsymbol{x'}, \Delta(g, \boldsymbol{x}, \boldsymbol{\epsilon'}))$ where $\boldsymbol{x'} = (f_1(x_1), \cdots, f_n(x_n))^T$ and $\boldsymbol{\epsilon'} = (\Delta(f_1, x_1, \epsilon_1), \cdots, \Delta(f_n, x_n, \epsilon_n))^T$.

These composite theorems and lemmas greatly extend our flexibility, allowing for progressively retrieving and bounding errors in a variety of QoIs. For instance, multiplications of multiple variables in the form of Πx_i can be preserved by iteratively leveraging the multiplication theory (Theorem 5) and composite property (Theorem 9). Errors in a general polynomial in the form of $\sum a_i x^i$ can be upper bounded by applying the additive (Theorem 7), multiplicative (Theorem 8) properties, and the error estimation of power functions (Theorem 1).

D. Example derivation on GE case study

Here, we showcase how to estimate the V_{total} in the GE case study using the proposed methods. Let x_1, x_2, x_3 denote V_x, V_y, V_z. The V_{total}, as denoted in Equation (1), can be formulated as the composition of a univariate function $f_1(x) = \sqrt{x}$ with the composition of a multivariate function $g_1(x_1, x_2, x_3) = x_1 + x_2 + x_3$ and an univariate function $f_2(x) = x^2$, which yields $V_{total} = f_1(g_1(f_2(x_1), f_2(x_2), f_2(x_3)))$. We estimate the upper bound of errors in f_2, g_1, and f_1 sequentially. First, the upper bound of errors in $f_2(x_i)$ $(i = 1, 2, 3)$ can be estimated using Theorem 1. This forms the error bound vector $\boldsymbol{\epsilon_{f2}} = (\Delta(f_2, x_1, \epsilon_1), \Delta(f_2, x_2, \epsilon_2), \Delta(f_2, x_3, \epsilon_3))^T$. Meanwhile, the value of $f_2(x_i)$ can be computed to obtain the new value vector $\boldsymbol{x_{f2}} = (f_2(x_1), f_2(x_2), f_2(x_3))$. After that, $\boldsymbol{\epsilon_{f2}}$ and $\boldsymbol{x_{f2}}$ will be used to compute $\epsilon_{g1} = \Delta(g_1 \circ f_2, \boldsymbol{x_{f2}}, \boldsymbol{\epsilon_{f2}})$ using Theorem 4. At last, we will compute $x_{g1} = g_1(f_2(x_1), f_2(x_2), f_2(x_3))$ to derive the final QoI error bound $\Delta(f_1 \circ g_1 \circ f_2, x_{g1}, \epsilon_{g1})$ using Theorem 2.

While using QoIs in GE as a demonstrative example, our theories are extendable to other scientific applications due to the following reasons. First, some common QoIs in the paper can be directly used by other applications (e.g., total velocity in climatology and cosmology). Second, the set of basis QoIs can composite diverse and complex functions such as multivariate polynomials and rational functions, which cover a broad range of QoIs, including molar concentration multiplications in combustion. Third, our theory can extend to new operators with derivable error control (e.g., isosurface [21]). This demonstrates the genericity of the proposed QoI-preserving theory.

V. Implementation and Optimization

In this section, we present the implementation of the QoI-preserving progressive retrieval framework, followed by optimizations and explorations on efficiency.

A. Algorithm and implementation

Our pipeline consists of two stages: a *data refactoring* stage, which transforms the original data into multi-precision segments for storage, and a *data retrieval* stage, which fetches and recomposes data till it reaches user-specified QoI tolerances. We omit the discussion on progressive refactoring as it's a direct application of the existing techniques. Generally speaking, for every data field, the refactoring stage produces a set of multi-precision segments and the corresponding metadata.

The proposed QoI-preserved data retrieval pipeline is presented in Algorithm 2, assuming that the retrieval starts from scratch. Lines 1-6 show the initialization of the progressive data representation and error bounds used in the first round of retrieval. The `while` loop starting from line 7 presents the iterative procedure used for QoI-preserving data retrieval

Algorithm 1 GENERAL DATA REFACTOR

Input: number of variables n_v, all variables $\{v_i\}$
Output: refactored multi-precision segments $\{\{s_p\}_i\}$ and metadata $\{m_i\}$

1: **for** $j = 1 \rightarrow n_v$ **do**
2: $\{s_p\}_i, m_i = \texttt{refactor}(v_i)$
3: **end for**

Algorithm 2 QOI-PRESERVED DATA RETRIEVAL

Input: refactored multi-precision segments $\{\{s_p\}_i\}$ and metadata $\{m_i\}$, value range of original variables $\{range_i\}$, requested QoI tolerances $\{\tau_i\}$
Output: retrieved data

1: **for** $i = 1 \rightarrow n_v$ **do**
2: $v_i \leftarrow \{0, \cdots, 0\}$ /*initial all the data fields as zero vector*/
3: **end for**
4: **for** $j = 1 \rightarrow n_v$ **do**
5: $\epsilon_j \leftarrow \texttt{assign\_eb}(range_j, \{\tau_i\})$ /*assign the initial error bounds*/
6: **end for**
7: tolerance_met $\leftarrow$ **false**
8: **while** !tolerance_met **do**
9: **for** $i = 1 \rightarrow n_v$ **do**
10: $v_i = \texttt{progressive\_construct}(v_i, \{s_p\}_i, m_i, \epsilon_i)$ /*construct the i-th variable to the target precision ϵ_i*/
11: **end for**
12: tolerance_met $\leftarrow$ **true** /*initialize flag as true*/
13: $\{\tau'_k\} \leftarrow 0$ /*initialize max estimated QoI errors*/
14: **for** $j = 1 \rightarrow n_e$ **do**
15: **for** $k = 1 \rightarrow n_q$ **do**
16: $\tau' \leftarrow \texttt{estimate\_error}(\{v_i\}_k, \{\epsilon_i\}_k)$ /*estimate QoI errors under current representation based on Section IV*/
17: **if** $\tau' > \tau_k$ **then**
18: tolerance_met $\leftarrow$ **false** /*if the errors of at least one QoIs are not met, set flag to false for another iteration*/
19: **if** $\tau' > \tau'_k$ **then**
20: $\tau'_k \leftarrow \tau', ind_k \leftarrow j$ /*record position of max error*/
21: **end if**
22: **end if**
23: **end for**
24: **end for**
25: **for** $k = 1 \rightarrow n_q$ **do**
26: $\{\epsilon_i\} \leftarrow \texttt{reassign\_eb}(\tau'_k, ind_k, \tau_k, \{v_i\}, \{\epsilon_i\})$ /*compute the new error bounds for variables based on data with max QoI errors*/
27: **end for**
28: **end while**
29: **return** $\{v_i\}$

that the data representations are gradually refined and then checked for QoI errors till all QoI tolerances are satisfied or a full-fidelity data representation has been retrieved. The `progressive_construct` function takes the newly retrieved multi-precision segments and recomposes the current data representation to a more accurate approximation. Lines 13-23 estimate the errors of QoIs derived from the reconstructed data using the theorems and lemmas discussed in Section IV, and compare them against user-requested error tolerance. When the requested tolerances are not met, we record the maximal estimated errors as well as their corresponding locations in the data space. Such information will be used for optimizing the error bound assigned for the next round of data retrieval (lines 17 - 22).

Regarding the PD error bound used for data retrieval, the initialization and iterative refinement stages adopt separate error assignment algorithms. At the initialization (line 5) stage, we

Algorithm 3 ASSIGN_EB

Input: value range $range$, requested QoI tolerances $\{\tau_i\}$
Output: error bound of current variable

1: $\epsilon \leftarrow 1$ /*initilize eb to maximal possible relative bound*/
2: **for** $j = 1 \rightarrow n_q$ **do**
3: **if** the j-th QoI involves this variable **then**
4: ϵ = min(eb, τ_j)
5: **end if**
6: **end for**
7: **return** $\epsilon * range$

Algorithm 4 REASSIGN_EB

Input: index k with largest QoI error, requested QoI tolerance τ_k, current data $\{v_i\}$, current error bounds $\{\epsilon_i\}$, reduction factor $c = 1.5$
Output: new error bound of the current variables

1: $\tau' \leftarrow \texttt{estimate\_error}(\{v_i\}_k, \{\epsilon_i\}_k)$ /*re-estimate QoI errors under potentially updated error bounds; see Section IV for details*/
2: **while** $\tau' > \tau$ **do**
3: **for** v_i involved in this QoI **do**
4: $\epsilon_i = \epsilon / c$
5: **end for**
6: $\tau' \leftarrow \texttt{estimate\_error}(\{v_i\}_k, \{\epsilon_i\}_k)$ /*re-estimate again*/
7: **end while**
8: **return** $\{\epsilon_i\}$

adopt the Algorithm 3: when a data field is utilized by multiple QoIs, its error bound will be determined by the minimal relative tolerance among all the requested QoIs involving this field. At the iterative refinement (line 26) stage, we adopt a uniform error-tightening strategy detailed in Algorithm 4, and prioritize the evaluation of the data point that generates the largest QoI errors in the previous evaluation. If the estimated QoI errors exceed the tolerance, we reduce the error bounds of all the variables involved in the computation for this QoI by a constant factor c ($c = 1.5$ in our implementation), iteratively retrieve the data, then estimate the QoI errors under the new error bounds and the reconstruction. Note that we execute the QoI error estimation only on the data point with the largest QoI error at the iterative refinement stage, which decreases the number of required iterations in Algorithm 2.

We have also implemented a mask-based outlier management method that filters out irregular points that potentially lead to unbounded error estimation. Using the CFD simulation data generated by GE as the example again, for nodes with values of $V_x = V_y = V_z = 0$, their decompressed values will be tiny when the error bounds used for retrieval are small. These close-to-zero values, however, could yield loose upper bounds for V_{total} (seeing Theorem 2) despite the small real errors. Accoordingly, in this example, we will use a bit-map to record the position of any data point with 0 total velocity, and only refactor data points whose values are non-zero.

B. Exploration on the best-fit progressive representation

Despite the fact that the proposed QoI error-control theories and pipeline can be generalized to any progressive compressors that meet the Definition 1, the amount of data retrieved may vary due to the different progressive refactoring and error bounding theories adopted by each compressor. In this section,

we review and evaluate the pros and cons of three error-controlled progressive compression algorithms.

Error-controlled compression with multiple snapshots: This type of methods leverages existing error-controlled compressors to compress data using a list of error bounds $\{\epsilon_i\}$, ranging from small to large. When an error bound ϵ^* is requested during progressive retrieval, one can choose a snapshot with a minimal i such that $\epsilon_i < \epsilon^*$. Due to the overlapping information among these snapshots, redundancies may be high when multiple precisions are requested during the progressive retrieval.

Error-controlled delta compression [16]: This type of methods also reduces data into multiple snapshots of multi-precision segments but eliminates the redundancy across the snapshots by compressing the residues (errors between the original and decompressed data) instead of the original data. As a result, it can be more efficient than directly compressing data into multiple snapshots with different error bounds but requires retrieving all first i snapshots (i is the minimal integer such that $\epsilon_i < \epsilon^*$) when the target error bound is ϵ^*.

Progressive compression with bitplane: This type of methods does not require to pre-set error bounds. Instead, it encodes the data using bitplanes such that the data can be retrieved and recomposed on demand. Similar to the error-controlled delta compression, it may not be the most efficient when only a single error bound is requested – directly compressing data using the requested error bound usually generates the smallest data footprints in such cases. PMGARD [15] is the leading technique in this kind, but its performance suffers from loose error control and thus may return more precision fragments than needed.

Below, we evaluate the performance of these three kinds of methods when a set of successively lower relative error bounds (i.e., a series of requests $\{\epsilon_i\}$ such that $\epsilon_{i+1} < \epsilon_i$) are requested. For the first two categories of progressive compressors, we use $SZ3$ as the underlying error-controlled compressors as it provides the tightest L^∞ error bound and thus could yield larger compression ratios than compressors with looser error bound [33], [43]. From now on, we refer to the integration of progression with multiple-snapshot, progression with delta compression, and SZ3 as `PSZ3` and `PSZ3_delta`, respectively. For progression with bitplane, we use PMGARD as the underlying refactor. The evaluations have been conducted using GE's CFD simulation data and their six QoIs (see Table III for details on datasets).

We evaluated 4 data fields in Fig. 2. The trend of VelocityY is similar to those of VelocityX and VelocityZ, and thus omitted. We set $\epsilon_i = 10^{-i}$ for $i = 1, 2, \cdots 10$ for both `PSZ3` and `PSZ3_delta` to create multiple snapshots (because this setting has a reasonable trade-off between additional storage cost and retrieval efficiency), and requested errors bounds on primary data as $\{\epsilon'_i\} = 0.1 * 2^{-i}$ for $i = 1, 2, \cdots 20$. The rate-distortion curves can be interpreted as follows: for any data point (x, y), its value represents the bitrate x (analogous to the percentage of data retrieved) under the requested tolerance y. As such, the closer a curve is to the left bottom (left indicates lower retrieve percentage and bottom indicates low error), the better. Notably, `PSZ3` has large bitrates under progressive requests, which is expected due to the redundancy among the multiple snapshots. Since `PSZ3` and `PSZ3_delta` rely on pre-set error bounds to produce multi-precision fragments, their bitrate curves exhibit a stair-case pattern: the amount of retrieved data remain constant across several adjacent error bounds and then present a sudden drop as the retrieval moves to the next snapshot. This leads to suboptimal results when the desired error bound is slightly lower than one of the pre-set error bounds. On the contrary, the trends of bitrate in `PMGARD` are linear with respect to the requested error bounds. Nevertheless, `PMGARD` constantly generates larger bitrates than those of `PSZ3-delta`, as the error bounds implemented with the former are looser than the latter.

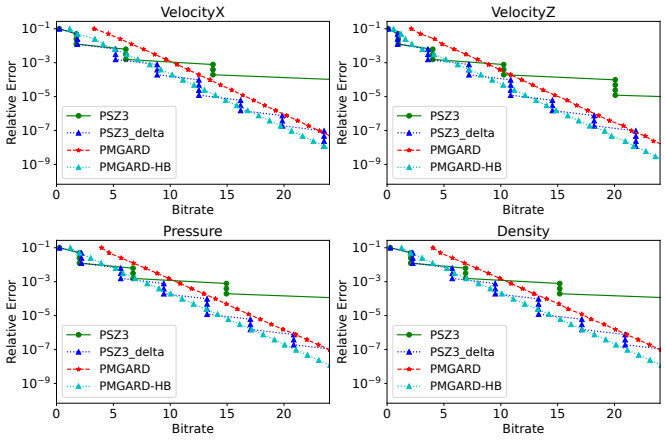

Fig. 2: Requested error and the resulting bitrate for different error-controlled progressive compressors.

For `PMGARD`, the gap between the requested bounds and real errors is mainly caused by the decomposition algorithm, in particular a L^2 projection, which maps low-level coefficient nodes to high-level nodal nodes employed for data decorrelation. This decomposition algorithm is derived from MGARD, which is specifically designed to provide optimal error control in L^2 norm, whereas can cause over-pessimistic estimation on L^∞. This is further validated by the experimental results shown in Fig 3, which examines the difference among the requested tolerance, estimated upper bound, and actual error measured after progressive retrieval. It shows that while the estimated errors are close to the requested tolerance, the actual errors are far smaller, causing an over-retrieval problem.

We propose to reduce the gap by omitting the L^2 projection in `PMGARD`'s decomposition algorithm. This could yield two benefits. First, without the cross-level intervention, the L^∞ norm can be accurately estimated through a summation of the maximal error bounds across all levels. Second, because L^2 projection is time-consuming, removing it can accelerate the refactoring and reconstruction process. We call the new progressive algorithm without L^2 projection `PMGARD-HB` as it replaces the orthogonal basis in MGARD with a hierarchical

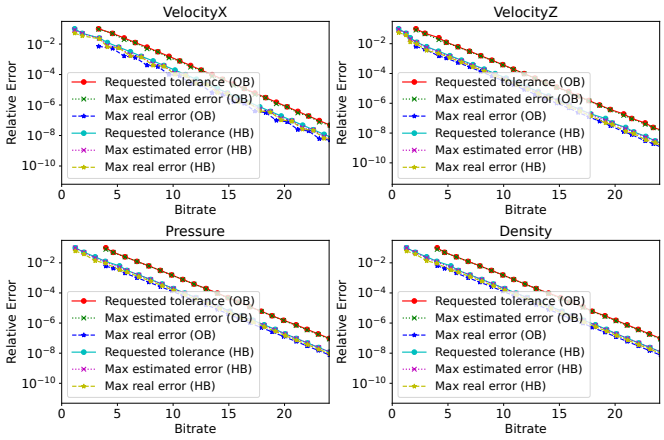

Fig. 3: Impact of decomposition basis on GE-small data. OB and HB represent PMGARD and PMGARD−HB, respectively.

basis from a mathematical point of perspective. As shown in Fig. 3, PMGARD−HB yields much more accurate error estimation than PMGARD, and this leads to constantly lower bitrates in all requested error bounds. We further compare PMGARD−HB with PSZ3 and PSZ3_delta in Fig. 2, where one can observe the advantages in most bitrates.

VI. Evaluation

We evaluate our methods in terms of error control, progressive retrieval efficiency, and performance using five datasets from four real-world applications with respective QoIs. Specifically, we first validate the error control of the QoI-preserving theory in Section IV, and then demonstrate the efficiency of three progressive methods –PSZ3, PSZ3−delta, and PMGARD−HB– which are the representatives of three mainstream error-controlled progressive approaches according to the literature.

A. Experiment setup

We conduct all our experiments using the Morgan Compute Cluster (MCC) [44] located at the University of Kentucky with 100 Gbps InfiniBand HDR interconnect. Each compute node in the system is equipped with 2 AMD EPYC ROME 7702P processors, each with 64 cores and 256 GB memory. Our benchmark datasets are from multiple computational science domains including CFD, cosmology [45], climate [46], and combustion [47], and their specifications are listed in Table III. Note that we use $\{\_\_\}$ in the dimensionality of GE data because its second dimension may have variable sizes. We use the first four datasets for sequential evaluation and the last dataset for measuring data transfer performance in a distributed memory environment. For QoIs, we use Equation (1) – (6) for the two GE datasets, and we test total velocity (i.e., V_{total} in Equation (1), also referred to as VTOT in the rest of the evaluation) for NYX and Hurricane data to demonstrate the generalibility. For S3D data, the data represent the molar concentration of 8 species associated with 21 reactions, and their multiplications generate the intermediate variables to derive the rate of progress. In particular, we present 4 multiplications involved in 2 reactions. For instance, x_0, x_1, x_3, x_4, x_5 used in our evaluation represent species H_2, O_2, H, O, OH, respectively, and x_1x_3 computes the molar concentration of O_2 and H in the reaction $H+O_2 \rightleftharpoons O+OH$. While we convert single-precision NYX and Hurricane data to double-precision for evaluation with small error bounds, our method directly applies to single-precision floating-point data.

TABLE III: Datasets and QoIs

Dataset	Dimensions	n_v	Type	Size	QoIs
GE-small	$200 \times \{\_\_\}$	5	double	137.96 MB	$Eq.(1)$ – (6)
Hurricane	$100 \times 500 \times 500$	3	double	572.20 MB	Total velocity
NYX	$512 \times 512 \times 512$	3	double	3.00 GB	Total velocity
S3D	$1200 \times 334 \times 200$	8	double	4.78 GB	Molar concentration multiplication
GE-large	$96 \times \{\_\_\}$	5	double	7.79 GB	$Eq.(1)$ – (6)

B. QoI error control

We first show that our theory provides guaranteed error control on the derivable QoIs in the evaluated applications. We use PMGARD−HB for demonstration purposes, and the same functionality can be provided by PSZ3 and PSZ3−delta as well. In particular, we present the max estimated QoI errors and actual QoI errors of the proposed method under a progressive set of requested QoI errors for GE data in Figs. 4. It is observed that the actual QoI errors in our method are always smaller than the estimated QoI errors, which are usually close but strictly smaller than the requested QoI errors. This validates our theory, which always provides an upper bound for the QoI error estimations. Different trends are observed for different variables as well. For instance, one can see a gap between the max estimated errors and actual errors in total velocity when the bitrate is low. This is because some decompressed data become close to 0 when the error bound is high, in which case the estimation of $\sqrt{x}$ generally leads to a loose bound. This situation becomes better when the error bound decreases to a certain threshold, which implies the diminishment of near-zero decompressed values. Furthermore, one can notice a larger gap between the max estimated errors and actual errors in PT than that in the other QoIs, which is reasonable because the estimation in PT is the most complex and involves more relaxation. In addition, the trends in T and C are similar, which is also as expected because their formulas are very similar.

We also present the results of total velocity on NYX and Hurricane data in Fig. 5, as well as four examples for molar concentration multiplications on S3D data in Fig. 6. Similar trends of QoI errors in total velocity are observed in the other two datasets, which demonstrate the generality of our algorithm. Also, our QoI error estimator demonstrates high accuracy on S3D QoIs. This is because these QoIs only involve the multiplications of two variables, which have predictable errors in almost all cases.

C. Retrieval efficiency

We then compare the efficiency of the three progressive approaches in terms of their retrieved data sizes. In particular,

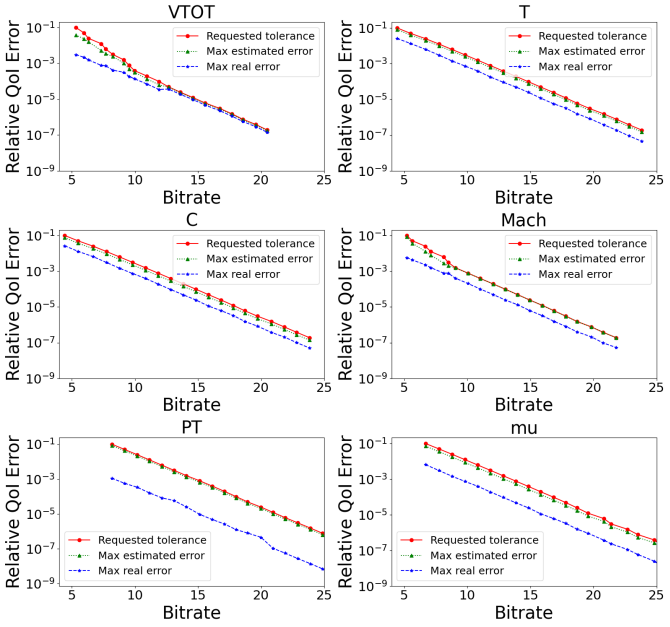

Fig. 4: Max estimated and max actual QoI errors under given requested QoI errors of PMGARD-HB on GE-small.

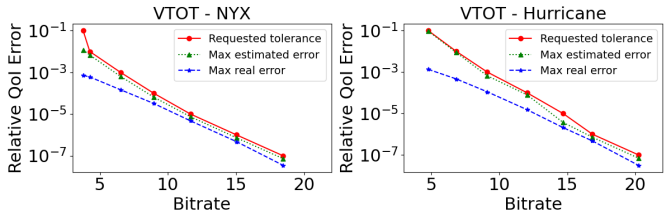

Fig. 5: Max estimated and max actual QoI errors under given requested QoI errors of PMGARD-HB on NYX and Hurricane.

we compare their bitrates under one requested QoI error to demonstrate generic cases; requesting progressive error bounds leads to similar results for PSZ3-delta and PMGARD-HB but may negatively impact PSZ3. Similar to the setting in Section V, we choose $\epsilon_i = 10^{-i}$ for $i = 1, 2, \cdots 18$) (used 18 because some datasets such as S3D requires high precision) as the pre-set error bounds for PSZ3 and PSZ3-delta.

a) Retrieved data size: We present the comparison of the three progressive approaches on the GE-small and S3D datasets in Fig. 7 and 8, respectively. The requested QoI errors are set to $\tau = 0.1 * 2^{-i}$ for $i = 0, 1, \cdots 19$, and we omit the total velocity on NYX and Hurricane as they have similar trends to the total velocity in the GE case. According to these figures, MGARD-HB generally leads to the best bitrate among all the three methods, and it has the most steady curve; PSZ3-delta is comparable to MGARD-HB in most cases, but it suffers from the sudden increase of bitrate in certain ranges, which is probably caused by the use of an additional multi-precision segment; PSZ3 is the least efficient in general due to the redundancy in the representation. In addition, it has very wild behavior when there is only a minor change on the request QoI error bound, which

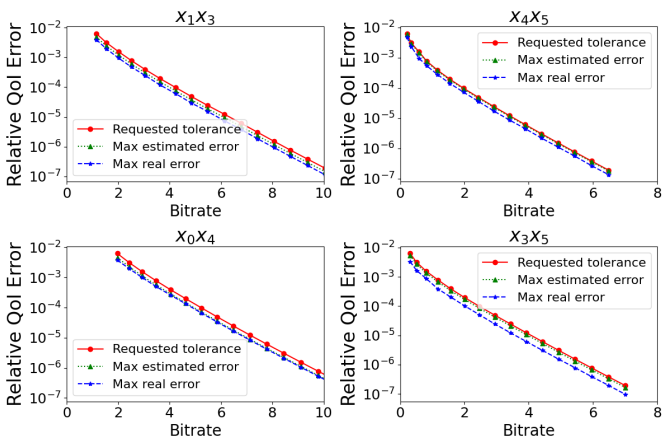

Fig. 6: Max estimated QoI errors and max actual QoI errors under given requested QoI errors of PMGARD-HB on S3D data.

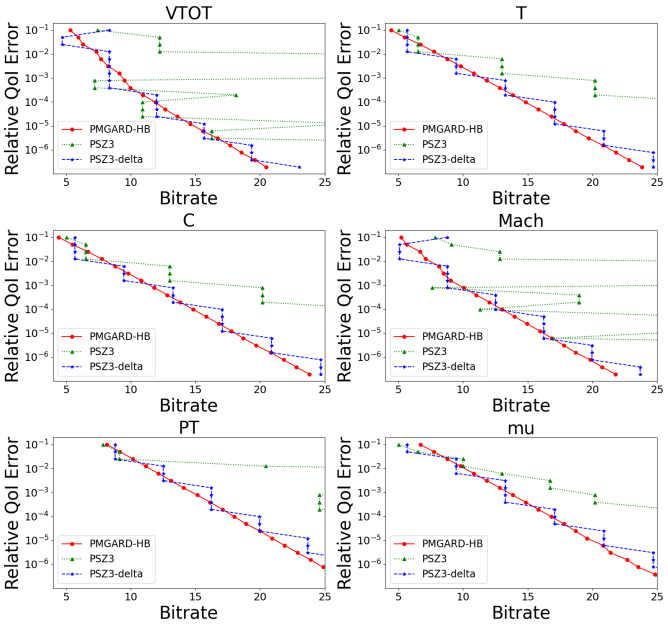

Fig. 7: Retrieval efficiency of different progressive approaches on GE-small data.

is probably caused by some extreme values (e.g., near-zero value in total velocity). Nevertheless, it performs reasonably well on S3D because of the high compressibility of the dataset and the relatively easy-to-preserve QoIs.

b) Refactoring and retrieval time: We present the refactoring and retrieval time of the three methods in Table IV. According to the table, PMGARD-HB has the least data refactoring time because it only needs to perform a single decomposition with bitplane encoding; in contrast, PSZ3 and PSZ3-delta require the execution of the compression procedure on either original data or the residues for 18 times (equal to the number of pre-set error bounds). The retrieval time of the three methods is in the same order, and their differences are mainly caused by the complexity of the decompression/reconstruction algorithms

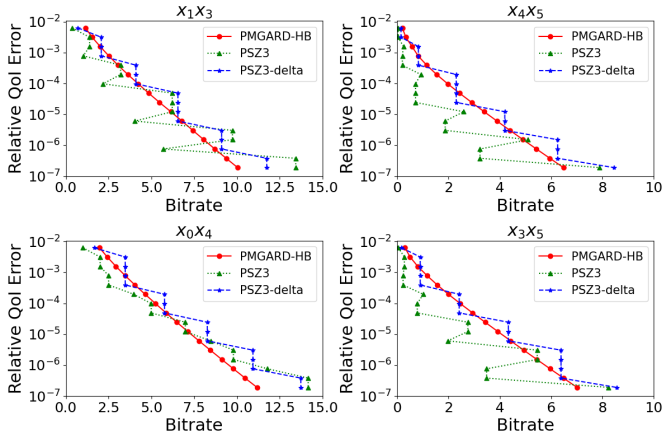

Fig. 8: Retrieval efficiency of different progressive approaches on S3D data.

TABLE IV: Refactor and retrieval time (seconds) of different progressive approaches on GE-small data

Compressor	Refactoring	Requested QoI error bound (VTOT)				
		1E-1	1E-2	1E-3	1E-4	1E-5
PMGARD-HB	3.30	0.84	0.95	1.12	1.34	1.52
PSZ3	14.63	0.53	0.69	0.63	1.27	1.15
PSZ3-delta	11.99	0.72	0.72	0.72	0.94	1.08

and the number of iterations used to determine the proper error bound on primary data.

D. Remote data transfer performance

We showcase how the proposed method can potentially improve the performance of data retrieval from remote sites using the GE-large data with VTOT as the target QoI. The refactored data is stored at MCC, and the data retrieval request is initiated from the Anvil supercomputer [48] located at Purdue University. The experiment is performed using 96 cores, each of which will deal with one data block in the GE-large data independently, and the data transfer is performed using the renowned data service software Globus [49]. The data refactoring time for PMGARD-HB, PSZ3, and PSZ3-delta is 2.17 seconds, 5.18 seconds, and 4.67 seconds, respectively, and the total data transfer time is depicted for remote retrieval in Fig. 9. As a baseline, the transfer time of the original data (3 variables, 4.67 GB in total) is roughly 11.7 seconds, as indicated by the dashed line. For progressive approaches, the data transfer time includes the retrieval time, which determines the proper amount of data, and the transfer time, which is the actual time for transmitting them. It is observed that all the progressive approaches lead to less total data transfer time when certain QoI errors can be tolerated. PMGARD-HB and PSZ3-delta exhibit similar performance as they have similar sizes of reduced data in this case, but PMGARD-HB features a shorter data refactoring time as noted above. When compared with the vanilla data transfer with the original data, PMGARD-HB yields 2.02× data transfer performance if the requested QoI error tolerance is 1E-5, because the size of the transferred data is less than 27% of the original one.

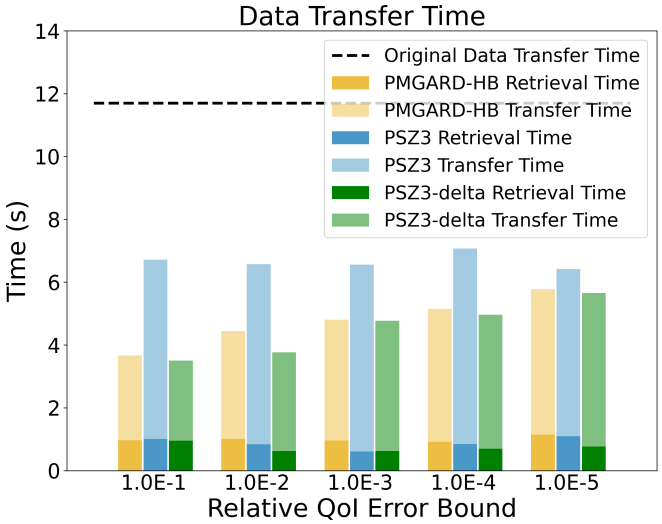

Fig. 9: Data transfer time (from MCC to Anvil via Globus) under different requested QoI error bounds with 96 cores using GE-large data. The dashed line indicates the time for transferring the original data.

VII. Conclusion

In this paper, we present a progressive data retrieval framework that is able to provide QoI error control on demand. We derive the theory to preserve a set of derivable QoIs, and leverage them to preserve six QoIs in a computational fluid dynamics simulation from a real-world application. Our theory is generic, and can be easily extended to preserve a wide range of QoIs that can be composited by the provided derivable QoIs. We further integrate three representative progressive methods into our framework and explore their efficiency in QoI preservation on five datasets from scientific applications. Experimental results demonstrate that the proposed framework can provide guaranteed error control on the target QoIs, which will lead to 2.02× data transfer performance while ensuring a QoI error of 1E-5. In the future, we will investigate how to extend this framework to incorporate more QoIs and progressive methods. We will also research how to enable tighter error controls with better efficiency.

Acknowledgments

This research was supported by the Exascale Computing Project CODAR, SIRIUS-2 ASCR research project, the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), and the Scientific Discovery through Advanced Computing (SciDAC) program, specifically the RAPIDS-2 SciDAC institute. It was also supported by the National Science Foundation under Grant OAC-2330367, OAC-2311756, OAC-2311757, OAC-2313122, and OIA-2327266. We would like to thank the University of Kentucky Center for Computational Sciences and Information Technology Services Research Computing for its support and use of the Lipscomb Compute Cluster, Morgan Compute Cluster, and associated research computing resources.

REFERENCES

[1] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Isabela for effective in situ compression of scientific data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 524–540, 2013.

[2] 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.

[3] 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.

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

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

[6] Ainsworth, Mark and Tugluk, Ozan and Whitney, Ben and Klasky, Scott, "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.

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

[8] 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*. IEEE, 2018, pp. 438–447.

[9] 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.

[10] H. Bhatia, D. Hoang, N. Morrical, V. Pascucci, P.-T. Bremer, and P. Lindstrom, "Amm: Adaptive multilinear meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 6, pp. 2350–2363, 2022.

[11] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[12] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The jpeg2000 still image coding system: an overview," *IEEE transactions on consumer electronics*, vol. 46, no. 4, pp. 1103–1127, 2000.

[13] J. P. Clyne, E. Bethel, H. Childs, and C. Hansen, "Progressive data access for regular grids." 2012.

[14] D. Hoang, P. Klacansky, H. Bhatia, P.-T. Bremer, P. Lindstrom, and V. Pascucci, "A study of the trade-off between reducing precision and reducing resolution for data analysis and visualization," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 1193–1203, 2018.

[15] 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*, 2021, pp. 1–13.

[16] V. A. Magri and P. Lindstrom, "A general framework for progressive data compression and retrieval," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[17] Q. Gong, X. Liang, B. Whitney, J. Y. Choi, J. Chen, L. Wan, S. Ethier, S.-H. Ku, R. M. Churchill, C.-S. Chang, M. Ainsworth, O. Tugluk, T. Munson, D. Pugmire, R. Archibald, and S. Klasky, "Maintaining trust in reduction: preserving the accuracy of quantities of interest for lossy compression," in *Smoky Mountains Computational Sciences and Engineering Conference*. Springer, 2021.

[18] J. Lee, Q. Gong, J. Choi, T. Banerjee, S. Klasky, S. Ranka, and A. Rangarajan, "Error-bounded learned scientific data compression with preservation of derived quantities," *Applied Sciences*, vol. 12, no. 13, p. 6718, 2022.

[19] X. Liang, H. Guo, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, and T. Peterka, "Toward feature-preserving 2d and 3d vector field compression." in *PacificVis*, 2020, pp. 81–90.

[20] X. Liang, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, T. Peterka, and H. Guo, "Toward feature-preserving vector field compression," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–16, 2022.

[21] P. Jiao, S. Di, H. Guo, K. Zhao, J. Tian, D. Tao, X. Liang, and F. Cappello, "Toward quantity-of-interest preserving lossy compression for scientific data," *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 697–710, 2022.

[22] L. Yan, X. Liang, H. Guo, and B. Wang, "Toposz: Preserving topology in error-bounded lossy compression," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[23] Q. Gong, C. Zhang, X. Liang, V. Reshniak, J. Chen, A. Rangarajan, S. Ranka, N. Vidal, L. Wan, P. Ullrich *et al.*, "Spatiotemporally adaptive compression for scientific dataset with feature preservation–a case study on simulation data with extreme climate events analysis," in *2023 IEEE 19th International Conference on e-Science (e-Science)*. IEEE, 2023, pp. 1–10.

[24] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs, "Data reduction techniques for simulation, visualization and data analysis," in *Computer graphics forum*, vol. 37, no. 6. Wiley Online Library, 2018, pp. 422–447.

[25] D. Hoang, H. Bhatia, P. Lindstrom, and V. Pascucci, "Progressive tree-based compression of large-scale particle data," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[26] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on circuits and systems for video technology*, vol. 6, no. 3, pp. 243–250, 1996.

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

[28] P. Deutsch, "Gzip file format specification version 4.3," 1996.

[29] Y. Collet, "Zstandard - real-time data compression algorithm," http://facebook.github.io/zstd/, online.

[30] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "Data compression for the exascale computing era-survey," *Supercomputing frontiers and innovations*, vol. 1, no. 2, pp. 76–88, 2014.

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

[32] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, and F. Cappello, "Significantly improving lossy compression for hpc datasets with second-order prediction and parameter optimization," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 89–100.

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

[34] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Dynamic quality metric oriented error bounded lossy compression for scientific datasets," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2022, pp. 1–15.

[35] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "Tthresh: Tensor compression for multidimensional visual data," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 9, pp. 2891–2903, 2019.

[36] T. Banerjee, J. Choi, J. Lee, Q. Gong, R. Wang, S. Klasky, A. Rangarajan, and S. Ranka, "An algorithmic and software pipeline for very large scale scientific data compression with error guarantees," in *2022 IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 2022, pp. 226–235.

[37] J. Zhang, X. Zhuo, A. Moon, H. Liu, and S. W. Son, "Efficient encoding and reconstruction of hpc datasets for checkpoint/restart," in *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2019, pp. 79–91.

[38] S. Li, S. Jaroszynski, S. Pearse, L. Orf, and J. Clyne, "Vapor: A visualization package tailored to analyze simulation data in earth system science," *Atmosphere*, vol. 10, no. 9, p. 488, 2019.

[39] S. Li, P. Lindstrom, and J. Clyne, "Lossy scientific data compression with sperr," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2023, pp. 1007–1017.

[40] G. Ballard, A. Klinvex, and T. G. Kolda, "Tuckermpi: A parallel c++/mpi software package for large-scale data compression via the tucker tensor decomposition," *ACM Transactions on Mathematical Software (TOMS)*, vol. 46, no. 2, pp. 1–31, 2020.

[41] Y. Liu, S. Di, K. Zhao, S. Jin, C. Wang, K. Chard, D. Tao, I. Foster, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data with diverse constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4440–4457, 2022.

[42] J. Liu, S. Di, K. Zhao, X. Liang, S. Jin, Z. Jian, J. Huang, S. Wu, Z. Chen, and F. Cappello, "High-performance effective scientific error-bounded lossy compression with auto-tuned multi-component interpolation," *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–27, 2024.

[43] 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*, vol. 30, no. 8, pp. 1857–1871, 2019.

[44] "Morgan Compute Cluster," https://docs.ccs.uky.edu, 2023, online.

[45] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel, "Nyx: A massively parallel amr code for computational cosmology," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, 2013.

[46] H. I. dataset, http://sciviscontest-staging.ieeevis.org/2004/data.html, online.

[47] J. Chen, "S3d-legion: An exascale software for direct numerical simulation of turbulent combustion with complex multicomponent chemistry," in *Exascale Scientific Applications*. Chapman and Hall/CRC, 2017, pp. 257–278.

[48] X. C. Song, P. Smith, R. Kalyanam, X. Zhu, E. Adams, K. Colby, P. Finnegan, E. Gough, E. Hillery, R. Irvine *et al.*, "Anvil-system architecture and experiences from deployment and early user operations," in *Practice and experience in advanced research computing*, 2022, pp. 1–9.

[49] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, 1997.