A General Framework for Error-controlled Unstructured Scientific Data Compression

Qian Gong*, Zhe Wang*, Viktor Reshniak*, Xin Liang†, Jieyang Chen‡, Qing Liu§, Tushar M. Athawale*, Yi Ju‖, Anand Rangarajan¶, Sanjay Ranka¶, Norbert Podhorszki*, Rick Archibald*, Scott Klasky*
Oak Ridge National Laboratory, Oak Ridge, TN*
University of Kentucky, Lexington, KY†
University of Alabama, Birmingham, AL‡
New Jersey Institute of Technology, Newark, CA§
University of Florida, Gainesville, FL¶
Max Planck Computing and Data Facility, Germany‖
Email: *gongq@ornl.gov

***Abstract*—Data compression plays a key role in reducing storage and I/O costs. Traditional lossy methods primarily target data on rectilinear grids and cannot leverage the spatial coherence in unstructured mesh data, leading to suboptimal compression ratios. We present a multi-component, error-bounded compression framework designed to enhance the compression of floating-point unstructured mesh data, which is common in scientific applications. Our approach involves interpolating mesh data onto a rectilinear grid and then separately compressing the grid interpolation and the interpolation residuals. This method is general, independent of mesh types and typologies, and can be seamlessly integrated with existing lossy compressors for improved performance. We evaluated our framework across twelve variables from two synthetic datasets and two real-world simulation datasets. The results indicate that the multi-component framework consistently outperforms state-of-the-art lossy compressors on unstructured data, achieving, on average, a $2.3 - 3.5\times$ improvement in compression ratios, with error bounds ranging from 1×10^{-6} to 1×10^{-2}. We further investigate the impact of hyperparameters, such as grid spacing and error allocation, to deliver optimal compression ratios in diverse datasets.**

***Index Terms*—unstructured data compression, error-control, multi-components**

I. Introduction

Unstructured meshes are widely utilized in finite element simulations due to their flexibility in representing complex geometries [1], [2]. The resulting datasets consist of mesh topology and multi-variate data defined on mesh vertices. As advancements in HPC hardware enable higher resolution and faster simulation timesteps, the volume of generated data imposes immense demands on storage and transmission. For instance, a gas turbine jet engine simulation using GE's GENESIS code encompasses billions of mesh points [3]. A single variable in double-precision format can reach tens of terabytes, with post-analysis requiring time-series data across multiple time steps. This scenario necessitates the development of effective data reduction techniques that minimize information loss while preserving scientific integrity. Due to the high entropy in floating-point data produced by scientific computations, lossless compression techniques typically achieve limited compression ratios (e.g., less than $2\times$). *Error-controlled lossy compression* has proven successful in scientific data compression by exploiting spatial and temporal redundancies and ensuring loss remains within user-prescribed bounds.

In recent years, numerous lossy compressors for floating-point data have been developed, including MGARD [4]–[6], SZ [7]–[10], ZFP [11], [12], SPERR [13], and GAE [14], [15]. These compressors typically assume a regular grid layout and rely on block or stencil-based algorithms to convert data into small-magnitude coefficients amenable to compression. When applied to unstructured-mesh data, most of these techniques require serialization of node data followed by compression on 1D arrays. This serialization often leads to sub-optimal compression ratios due to overlooking data correlations within the mesh topology and the incoherent data distribution caused by the arbitrary connectivity through mesh cells. While value-based vertex traversal [16] can mitigate some of these issues, it may slow down the compression and add overhead for storing the traversal graph. There are also methods that reduce unstructured data over their original mesh topology, but often transition data into entirely different formats [17], which complicates post-analysis and error preservation, or are limited to specific mesh types or topology (e.g., tetrahedrons which can be gradually refined [18]).

In this paper, we propose a generic approach that can be integrated with *any* error-controlled lossy compressors, without requiring invasive code changes or custom mesh traversal, to significantly improve compression ratios for data defined on *arbitrary* unstructured meshes. Our method approximates datasets defined over unstructured meshes onto rectilinear grids and performs *multi-component* compression

This manuscript has been authored in part by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for U.S. Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

on the interpolated values on the rectilinear grid and the interpolation residuals on the original unstructured mesh vertices. To motivate our approach, we compare a 2D unstructured mesh and a 2D rectilinear grid. Figure 1 shows a variable (pressure) simulated by OpenFoam and its scatter-based interpolation on a 2D rectilinear grid. Using a number of nodes equivalent to 39% of the mesh vertices, the resulting approximation faithfully conveys the properties of the original variable and produces low approximation errors when interpolated back to mesh vertices. These approximation errors, refereed as residuals in this paper, are more amenable to compression than the original data values, and the approximation on the rectilinear grid can also achieve high compression ratios by exploiting data correlations in high-dimensional space.

Our framework can be summarized as follows. Given the vertices and associated data values of an unstructured grid and a user-prescribed error bound, our framework first constructs a rectilinear grid optimized for high compression ratios and low approximation errors. Next, we map the data values from the mesh vertices to the rectilinear grid nodes using a customized or user-supplied interpolation kernel and compute the residuals at the mesh vertices. Finally, we distribute the error bound and use error-controlled lossy compressors to independently reduce both the approximation and the residual components.

The salient aspects of our work are:

- We develop a generic and effective framework to reduce unstructured mesh data based on mesh-rectilinear grid interpolation and multi-component compression. Existing error-controlled lossy compressors can be seamlessly integrated into our framework to enhance their performance in a rather non-disruptive manner.
- Our approach does not require special types of meshes or typologies. Additionally, since the mesh-to-rectilinear-grid conversion is independent of the field values, the corresponding vertex mapping can be precomputed and reused across multiple variables and timesteps, which reduces storage overhead and accelerates computations.
- The compression ratio and computational cost can be adjusted via the grid spacing and the error bounds used for compressing the grid approximation and residuals. The overhead in computing time varies with grid spacing, ranging from 26% to 100% under the parameter settings used in our experiments. Our approach achieves an average improvement in compression ratios of approximately $2.3 - 3.5\times$, and up to $14\times$, among three lossy compressors evaluated within our framework.

The details of our approach follow a review of related work on error-controlled and unstructured data compression. We demonstrate the generality of the proposed framework through the integration with three state-of-the-art lossy compressors and evaluate performance across different variables taken from multiple synthetic and real applications simulated unstructured mesh datasets.

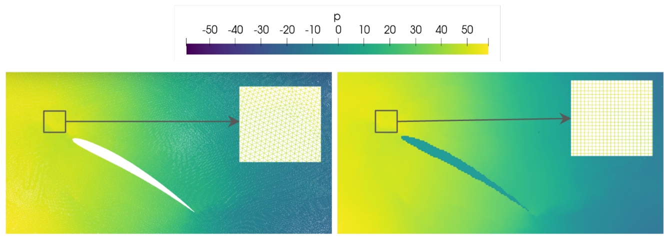


Fig. 1: Example of a variable generated using OpenFoam on a 2D unstructured mesh (left) and its interpolation on a rectilinear grid (right). The airfoil blade is a hollow region in the left figure and interpolated as zeros in the right figure.

II. Related Work

Lossy compression reduces data by exploiting redundancies, achieving greater compression ratios than lossless compression at the cost of some accuracy. It's crucial that the data compressed using lossy methods meet numerical accuracy requirements to ensure the preservation of scientific integrity. State-of-the-art, error-controlled compressors fall into two categories: transformation-based approaches and prediction-based approaches. Transformation-based methods (e.g., MGARD [4], [6], [19], ZFP [12], TTHRESH [20]) utilize customized transformations to mitigate data redundancies. In contrast, prediction-based methods (e.g., ISABELA [21], SZ [22]) leverage diverse predictors to de-correlate data. Both types of methods produce coefficients of smaller magnitude that become zero after quantization, enabling compacted storage of the compressed data.

Most existing lossy compressors are designed for rectilinear grid data. For example, ZFP divides data into non-overlapped blocks and transforms the floating-point data in each block into variable-length bit-streams independently. TTHRESH reduces dimensionality using techniques like Tucker and HOSVD, which rely on tensor decomposition on multidimensional grids. Similarly, all predictors implemented in the SZ family (e.g., Lorenzo, polynomial interpolation, and regression) require partitioning data into small blocks or sub-grids of different sampling rates.

Few lossy compressors can directly handle unstructured mesh data. MGARD reduces data through orthogonal decomposition, requiring a hierarchical refinement to be performed among triangular/tetrahedral cells, which limits its use case [18]. Ren et al. [16] proposed a prediction-traversal approach that sequentially visits/compresses mesh nodes until an unpredictable node is encountered, then initiates a separate sequence from a new seed until all nodes are traversed. Due to the additional costs of storing traversal sequences, this approach delivers improvements only within limited error ranges and has a high computational overhead, with a throughput of less than 1MB/s. In contrast, most state-of-the-art lossy compressors (e.g., MGARD, SZ, ZFP) achieve throughputs of tens to hundreds of MB/s [4], [23].

Researchers also explored compressed sensing techniques for unstructured mesh data compression [24], [25]. However,

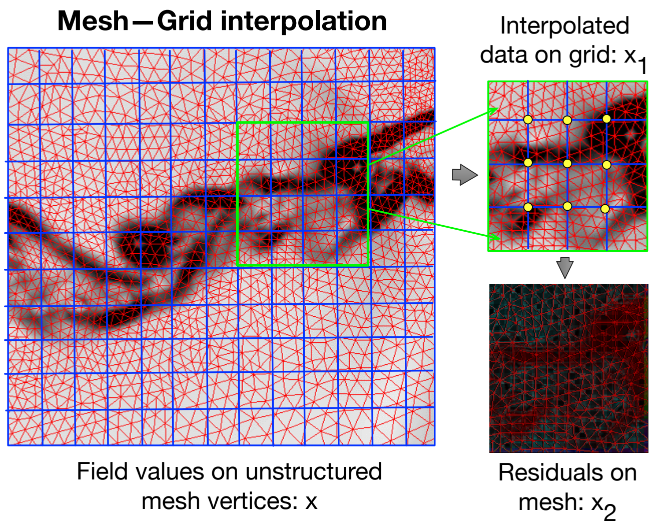

Mesh—Grid interpolation
Interpolated data on grid: x1
Field values on unstructured mesh vertices: x
Residuals on mesh: x2

Fig. 2: A multi-component compression for unstructured mesh data consists of building an interpolation on rectilinear grid and independent compression of grid interpolation and residuals on meshes.

like neural network-based compression models [14], [15], [26], [27], these methods suffer from unbounded errors and require tens to hundreds of iterations to reach a faithful reconstruction. Another set of related research involves the reduction of unstructured meshes. For example, El-Rushaidat et al. [17] aimed to convert unstructured data into rectilinear grids, while Berger and Rigoutsos [28] further explored cluster and adaptive mesh refinement (AMR) algorithms to reduce grids into non-uniform structure consist of fewer rectangular patches. Various studies have also addressed compressing data in AMR forms [29]–[31]. Our work is motivated by some mesh-to-grid data approximation techniques discussed in the aforementioned work but fundamentally differs in two key aspects: first, our primary objective is to reduce the storage cost rather than accelerate analytic tasks such as visualization; second, we aim to maintain the general structure of reduced data and mathematically preserve errors incurred during data compression.

III. Preliminaries

In this section, we provide an overview of the proposed framework and the proof of error control. As illustrated in Figure 2, the key idea is to represent field values on mesh vertices using an approximation over a rectilinear grid and the approximation error (i.e., residuals). We denote the original field values on mesh vertices as x, the interpolated data on grid points as x_1, the residuals on mesh vertices as x_2, and the operator that interpolates x_1 back to mesh vertices as $g(.)$, where $g : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ is a multidimensional, multivariate function that maps the n_1 data points in the rectilinear grid to n_2 data points in the original grid. The original vertex values can then be reconstructed through $x = x_2 + g(x_1)$.

Let $c(.)$ represent an error-controlled compressor that guarantees the error in the reconstructed data satisfies $\max|\epsilon| = \max|u - u'| \leq \tau$, where ϵ is the compression error, τ is the input error bound, and u and u' represent the original and lossy reduced data. Compression of the original data values on mesh vertices leads to an error $\max|\epsilon| = \max|x - x'| \leq \tau$, and compression for the grid approximation and residuals results in two other error components: $\max|\epsilon_1| = \max|x_1 - x_1'| \leq \tau_1$, and $\max|\epsilon_2| = \max|x_2 - x_2'| \leq \tau_2$. The reconstruction of x using the lossy reduced components x_1' and x_2' can be captured by $x' = g(x_1') + x_2'$. For any linear interpolation function g, such that $g_i(x) = a^T x$, where a is the interpolation coefficient, the error in the reconstructed data can be bounded by $(\sum_j |a_j|)\tau_1 + \tau_2$ according to basic linear algebra. Hence, given a prescribed error bound τ on the unstructured mesh data x, we can compress the components of x_1 and x_2 under the error bound τ_1 and τ_2, respectively, and ensure the error in the reconstructed mesh data satisfies $|\epsilon| \leq \tau$ as long as $(\sum_j |a_j|)\tau_1 + \tau_2 \leq \tau$. In a special case where g corresponds to nearest neighbor value or piece-wise linear interpolation of grid points, this requirement reduces to $\tau_1 + \tau_2 \leq \tau$ because $\sum_j |a_j| = 1$.

IV. Multi-component Unstructured Data Compression

A. *Rationale of multi-component compression*

In the previous section, we proved that compression for unstructured mesh data can be performed through independent compression of two components—the approximation on rectilinear grid and residuals—while ensuring the accuracy of recomposed data by adjusting the error bounds used for compressing individual components. We propose constructing the rectilinear grid based on the locations, rather than field values, of mesh vertices. The compression ratio (CR) of the proposed multi-component approach is captured as CR $= (c_1 + c_2)/S$, where c_1 and c_2 are the compressed sizes of interpolation approximation and residual components, respectively, and S is data original size. We exclude the grid and mesh layouts (e.g., node/vertices coordinates and cell connectivity) from the CR measurement as they remain static and can be reused across numerous timesteps and multiple variables sharing the same set of meshes.

Unlike vertices in unstructured meshes, whose indices often ignore the spatial coherence of field values, axis-aligned grids have implicit geometric coherence. Thus, lossy compression of approximation values on grids can better exploit the smoothness of scientific data in space and time. Additionally, assuming the interpolated grid values provide a faithful representation of the mesh data, the residual values will be small hence are more compressible than their original values. These points form the rationale for the proposed multi-component compression, suggesting that the combined size from compressing two components could be smaller than directly compressing unstructured mesh data. The main penalty of using multi-component compression lies in the overhead of compressing the grid interpolation and performing interpolation operations.

Nevertheless, provided sufficiently larger compression ratios can be achieved, as demonstrated in Section V, the reduction on I/O time and storage cost will likely compensate for the overhead in compression time. We also study the impact of grid sampling rate on approximate errors and the aggregated compression ratios in Section V-B, providing guidance for users to make trade-offs.

B. Design overview

Figure 3 illustrates the workflow of our multi-component compression and decompression for values on an unstructured mesh. The grid construction produces grid layouts and a mesh-grid mapping table, which can be reused across multivariate and timestep datasets to accelerate the interpolation operation $g(x_1)$ required for residual calculation and mesh data reconstruction. The mesh-to-grid interpolation can be performed through any customized algorithm, and we introduce a computationally light interpolation function in Section IV-D. The residual values on mesh vertices are then calculated through $x - g(x_1)$, where x is the original value at a mesh vertex.

The multi-component framework splits the input error bound τ between τ_1 and τ_2, which are tolerance used for compressing the interpolated grid nodes and mesh residual values. The actual compression is conducted through error-controlled lossy compressors. The interpolated grid values are compressed in the N-dimensional Cartesian coordinate space, whereas the residuals are serialized into a 1D array for compression. The reconstruction follows an inverse procedure to compression, where the interpolation values (i.e., x'_1) and residuals (i.e., x'_2) are decompressed independently and recomposed to form the reconstructed values on mesh vertices. The entire pipeline is automated but allows users to optimize for individual cases by fine-tuning parameters such as the grid spacing and error distribution between grid interpolation and residuals. The hyperparameter tuning and its impact on compression performance will be explored in Section V-B.

C. Grid Construction

The size of mesh cells is often nonuniform across computational space, with finer cells near boundaries and regions requiring a higher resolution. Our grid construction module consists of two steps: initializing grids with uniform spacing along each spatial dimension, followed by adaptive coarsening based on the spatial distribution of mesh vertices and the operator $g(x_1)$ used for grid-to-mesh interpolation. The module also outputs a vertex-wise index table $\{m_i\}$ recording the vertex-node mapping.

The algorithm for initializing the uniform grid is presented in Algorithm 1. Generally, the grid construction module traverses meshes based on cell connectivity, measuring the distance between vertices and choosing the value at the bottom $k\%$ percentile (user-supplied parameter) along each spatial dimension as the grid spacing of the corresponding axis. Grid corners are assigned based on the minimum and maximum value of the mesh vertices along each dimension. As shown in the Algorithm 1, an upper bound on grid discretization $G_{\max}$ may also be prescribed to avoid over discretization.

Algorithm 1 GRID_INIT

Input: mesh vertices coordinates $\{p_i\}$, connectivity $\{c_j\}$, dimension D, $G_{\max}$ max number of nodes along each axis, k%ile of vertex distance
Output: grid corners $\{gs_d\}$ and spacing $\{$min_p_d, max_$p_d\}$

```
1: $\{\{s_i\}_d\}$, $\{$min_$p_d$, max_$p_d\}$ ← traverse_mesh($\{p_i\}$, $\{c_j\}$) /*obtain min/max, and vertex distance along each dimension*/
2: for $d = 1 \to D$ do
3:     $\text{gs}_d \leftarrow$ percentile($k$, $\{s_i\}_d$)
4:     $\text{gn}_d = (\text{max\_}p_d - \text{min\_}p_d)/\text{gs}_d$
5:     while $\text{gn}_d \geq G_{\max}$ do
6:         $k \leftarrow k + \delta$
7:         $\text{gs}_d \leftarrow$ percentile($k$, $\{s_i\}_d$)
8:     end while
9: end for
10: return $\{\text{gs}_d\}$, $\{$min_$p_d$, max_$p_d\}$
```

We implemented a lightweight function $g(.)$ to minimize the computational overhead. For each mesh vertex $x[k]$, we approximate its value using the nearest grid node, m_k, such that the residual is derived as $x_2[k] = x[k] - x_1[m_k]$. Due to varied cell sizes, irregularly shaped boundaries, and hollow regions in the mesh geometry, some grid nodes may not be utilized during the residual and reconstruction computation, thus leading to resource waste. To address this, we adaptively coarsen the grid, as detailed in Algorithm 2. The grid coarsen module first traverses the mesh vertices to find their closest grid nodes. This traversal has two objectives: (1) building a grid-to-mesh mapping which can be reused to accelerate residual calculation and data recomposition, and (2) flagging grid nodes that are unvisited in back interpolation. Next, the algorithm scans along each grid axis, checking for unvisited arrays/planes.

Algorithm 2 GRID_COARSEN

Input: mesh vertex coordinates $\{p_i\}$, grid layout S, dimension D
Output: mesh-grid node mapping $\{m_i\}$, updated S

```
1: $G$_node.visited ← False
2: /*loop through $n_v$ mesh vertices*/
3: for $i = 1 \to n_v$ do
4:     $m_i$ ← MAP($p_i$, $S$) /*mapping varies with $g(.)$*/
5:     $G$_node[$m_i$].visit ← True
6: end for
7: /*loop through grid nodes*/
8: for $d = 1 \to D$ do
9:     for $k$ in coord[$d$] do
10:        planes.visited ← check_visit($G$_node.visited, $S$, $k$) /*check if any node in the plane forms by the rest dimensions was visited*/
11:        if planes.visited == False then
12:            update $S$, $\{m_i\}$
13:        end if
14:    end for
15: end for
16: return $\{m_i\}$, $S$
```

Since the AMR-type data cannot be handled by most lossy compressors, coarsening must be performed on all nodes along the corresponding dimensions within the coarsened range. For irregular-shaped meshes, they may yield grids with “blank” regions/nodes that cannot be eliminated due to the continuity

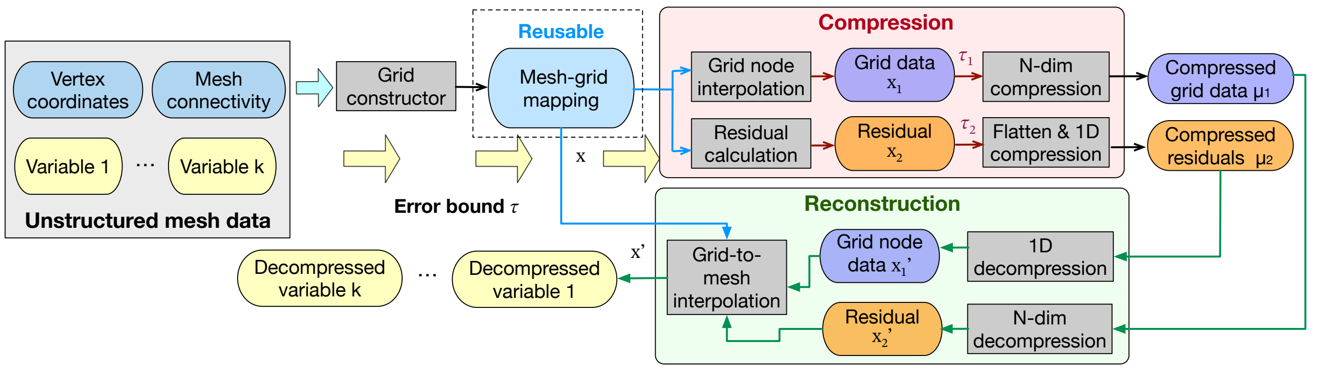


Fig. 3: Workflow of the proposed multi-component, error-controlled compression and decompression algorithm for unstructured mesh data. We assume that there are multiple variables sharing the same set of meshes and the mesh remains static across different timesteps data, such that the mesh-grid mapping can be pre-computed to accelerate compression and decompression.

along grid axes. Hence, our framework implements a variation algorithm where the component x_1 contains only grid nodes visited in Algorithm 2. These nodes are treated as *seeds* for vertices in surrounding regions to interpolate residuals and then serialized for compression. Our framework automatically switches to the variation implementation when the percentage of G_node.*visit* fails to reach a given threshold.

D. Compression

Algorithm 3 details the multi-component compression module, which consists of three parts: grid values interpolation, residual calculation, and independent compression of interpolated values and residuals. We denote the interpolation function used for approximating grid values as $f(.)$. Note that the objective of our research is to maximize the compression ratio of unstructured data under a numerical error bound, rather than to produce the best "visualization". The function $f(.)$ that produces the smoothest approximation may not lead to the smallest magnitude for mesh residuals since the back interpolation $g(.)$ used for residual calculation must be a linear function to ensure error preservation. Considering the overhead incurred to compression throughput, our framework implements two operators for $f(.)$, one based on linear interpolation and another based on cluster. The cluster-based function takes the mean of n vertices data closest to node j (i.e., $x \in C_j$) as the interpolated value, i.e., $x_1[j] = \sum_{x\in C_j}^{n} x[i]/n$. Despite its simplicity, $x_1[j]$ minimizes the residuals as $\sum_{x\in C_j}(x[i]-x_1[j])^2$ is smallest when $x_1[j]$ is the mean value of the $x[i]$ assigned to the cluster C_j. Users can replace $f(.)$ using other customized interpolation functions. The residual computation and independent compression—$\mathbb{C}(x_1)$ and $\mathbb{C}(x_2)$—are described in Section IV-B. Here, $\mathbb{C}$ can be any error-controlled lossy compressor.

E. Reconstruction

Reconstruction follows the inverse procedure of compression, as detailed in Algorithm 4. Similar to multi-component compression, grid approximation and mesh residual values are decompressed independently. Data values are recomposed

Algorithm 3 MULTI-COMPONENT COMPRESSION

Input: mesh-grid mapping $\{m_i\}$, mesh vertices value $\{x_i\}$, grid layout S
Output: compressed residual values μ_2, compressed grid values μ_1

```
1:  $x_1 \leftarrow 0$
2:  /*Obtain grid interpolation values*/
3:  for $i = 1 \rightarrow n_v$ do
4:      $x_1[m_i] \leftarrow f(x_i, x_1[m_i])$
5:  end for
6:  $x_2 \leftarrow x - g(x_1)$
7:  /*multi-component compression*/
8:  $\mu_1 = \mathbb{C}(x_1)$ /*$\mathbb{C}$ as the compression function*/
9:  $\mu_2 = \mathbb{C}(x_2)$
10: return $\mu_1, \mu_2$
```

back to the original mesh vertices through: $x' = g(x_1') + x_2'$. Errors in the recomposed mesh data are guaranteed to stay within the user-prescribed input tolerance, as we proved in Section III.

Algorithm 4 MULTI-COMPONENT RECONSTRUCTION

Input: compressed residual values μ_2, compressed grid values μ_1, grid layout S, mesh-grid mapping $\{m_i\}$
Output: reconstructed mesh values x'

```
1: $x_1' \leftarrow \mathbb{D}(\mu_1)$ /*$\mathbb{D}$ as the decompression function*/
2: $x_2' \leftarrow \mathbb{D}(\mu_2)$
3: for $i = 1 \rightarrow n_v$ do
4:     $x_i' \leftarrow g(x_1'[m_i]) + x_{2i}'$
5: end for
6: return $x'$
```

V. EXPERIMENTS

A. Datasets and evaluation metrics

In this section, we evaluate our multi-component compression framework using 12 variables captured from two synthetic datasets and two real-world simulation datasets. Detailed specifications are presented in Table I, with selective visualization shown in Figure 4. The synthetic datasets were generated by OpenFoam [32] through the PISO algorithm [33], which solves the pressure-velocity calculation for the Navier-Stokes equations. We simulated turbulent flow for a

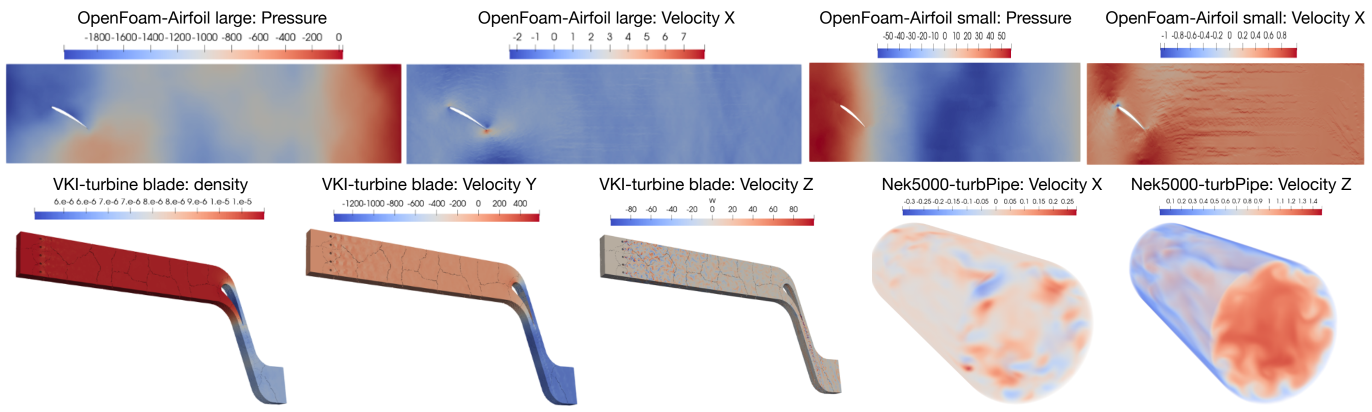


Fig. 4: Visualization of the benchmark mesh data

2D NACA airfoil validation case on a small and large sets of meshes and initialization parameters, denoted as airfoil-small and airfoil-large. The Nek5000-turbPipe dataset was obtained from a simulation of fully-developed turbulent pipe flow [34] generated by Nek5000 [35], a computational fluid dynamics code. The VKI dataset was obtained from a simulation of the VKI (von Karman Institute) gas turbine blade cascade test case [36] generated by GE's GENSIS software.

We utilize three state-of-the-art error-controlled lossy compressors (detailed in Table II) within our framework to demonstrate the enhancement and generality. As reviewed in Section II, these compressors are designed for compressing scientific data on rectilinear grids, requiring mesh data to be serialize, usually based on their original order in the files, before compression [37], [38]. We denote this 1D serialization-based compression as *default*.

Throughout the evaluation, we demonstrate the impact of different hyper-parameters and how compression ratios can be significantly improved under our multi-component framework. We define the following metrics: (1) Compression ratio (CR): defined in Section IV-A. (2) Achieved compression error ϵ: errors measured in reconstructed data, required to be less than the input compression error bound τ. We choose the relative $l2$ error, denoted as $\epsilon = \sum \sqrt{(x-x')^2} / \sum \sqrt{x^2}$, as the error metric, where $\sum \sqrt{x^2}$ represents the norm of the original data. (3) Improvement ratio: the ratio of the compression ratios achieved by our multi-component framework to the default approach implemented with the same lossy compressor. (4) Throughput overhead: the *extra* time spent using our multi-component framework comparing to the default method.

The experiments were conducted on the Frontier supercomputer [39]. Each Frontier compute node consists of a 64-core AMD 3rd Gen EPYC CPU with access to 512 GB of DDR4 memory. All source code was writeen in C/C++. Although most lossy compressors tested in this work support multi-threading or GPU acceleration, we tested only the single core CPU execution mode as code parallelism is irrelevant to our evaluation metrics.

dataset 1	dim	attributes	# of vertices
Nek5000-turbPipe	3D	velocities	1,451,520
VIK turbine blade	3D	pressure, density, Velocities	209,121,200
OpenFoam airfoil-small	2D	pressure, velocities	527,904
OpenFoam airfoil-large	2D	pressure, velocities	1,994,776

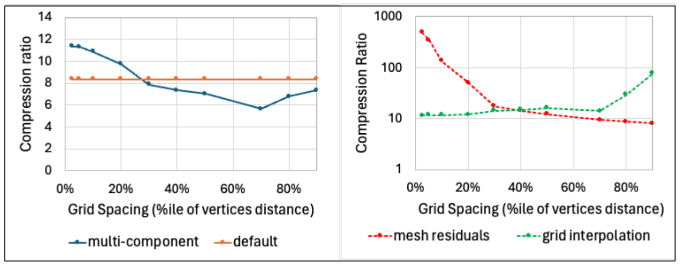
TABLE I: Data sets used for benchmark

Compressor	Version	GitHub repository
MGARD	1.5.2	https://github.com/CODARcode/MGARD
SZ3	3.1.8	https://github.com/szcompressor/SZ3
ZFP	1.0.1	https://github.com/LLNL/zfp

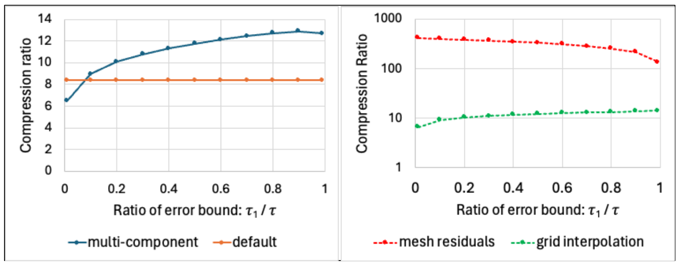
TABLE II: Compressors used in our multi-component compression framework

B. Hyperparameter optimization

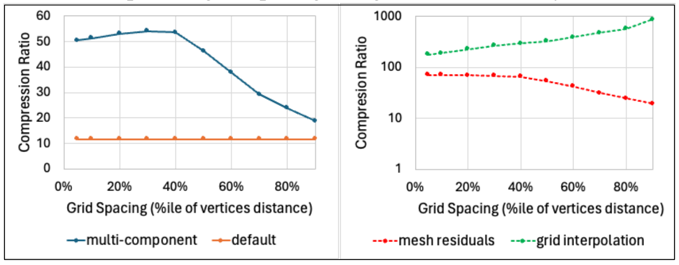
We begin by exploring the impact of grid spacing on multi-component compression ratios. The grid spacing is the node distance along each spatial dimension, which is selected based on the percentile (%ile) of the vertex distance, as discussed in Algorithm 1. It represents the *finest* scale in grid approximation. We evaluate parameter settings using the *highly fluctuated* velocity data from a fully-developed turbulent pipe flow simulated by Nek5000 and the pressure data from a *relatively smooth* region in a VKI turbine blade simulation. Figure 5 illustrates the compression ratios under different parameter settings, with combined compression ratios plotted on the left and the individual compression ratios for grid interpolation and residual data on the right in each subfigure. As shown in Figure 5a and 5c, as the grid spacing coarsens, the compression ratio of grid interpolation data rises due to the reduced number of grid nodes. Meanwhile, the compression ratio of interpolation residuals decreases due to increased approximation errors. The combined compression ratios peak at grid spacing equal to around $30-40\%$ of the vertex distance percentile for VKI pressure data and at 1% of the vertex distance percentile for Nek5000 data. The turbulent Nek5000 data requires fine-grained grid representation, or it would yield



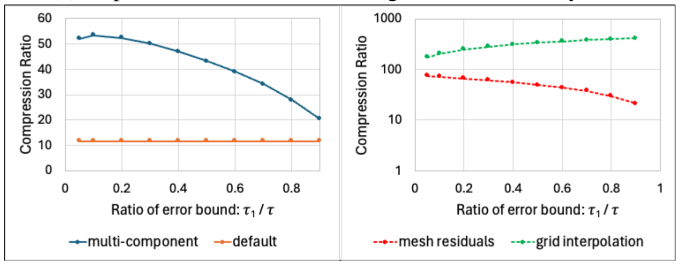
(a) Impact of grid spacing using Nek5000 velocity data



(b) Impact of error allocation using Nek5000 velocity data



(c) Impact of grid spacing using VKI pressure data



(d) Impact of error allocation using VKI pressure data

Fig. 5: Impact of hyper-parameters on achieved compression ratios: demonstrated using data exhibiting different scales of smoothness.

significant interpolation errors (i.e., mesh residuals). The VKI pressure data, in comparison, are relatively smooth, thus can afford to interpolate on coarser grids.

Next, we fix the grid spacing at the 1% percentile and 20% percentile for the Nek5000 and VIK datasets, respectively, and study the impact of error allocation in Figure 5b and 5d. Our implementation requires $\tau_1 + \tau_2 \leq \tau$, where τ is the requested error bound on mesh data, τ_1 and τ_2 are error bounds used for compressing the grid interpolation and mesh residuals, respectively. As the τ_1 increases, the compression ratio of grid data becomes larger, and the compression ratio of interpolation residual drops. However, the combined compression ratios for Nek5000 and VKI datasets exhibit opposite trends because the combined ratio is upper bounded by the compressibility of grid interpolation for Nek5000 data and by interpolation residuals for VKI data.

C. Benchmark Datasets Evaluation

To verify that our framework meets error bounds, we use the Nek5000-turbPipe velocity field and plot in Figure 6 the measured error ϵ as a function of input error bound τ. We denote the data compressed through serialization by "-default" and the proposed multi-component approach by "-mc", with both approaches using MGARD as the underlying compressor. The evaluation shows that the errors in multi-component compressed data stay below the prescribed error bounds and are comparable to the ones derived through standard serialization-based approach.

Using MGARD as the underlying compressor, we plotted in Figure 7 the compression ratios as a function of τ for 12 variables in four unstructured datasets. We chose a grid spacing of 10%, 40%, 30% percentile, and $\tau_1/\tau = 0.99$, 0.15, 0.9 for all variables in Nek5000, VKI, and two OpenFoam

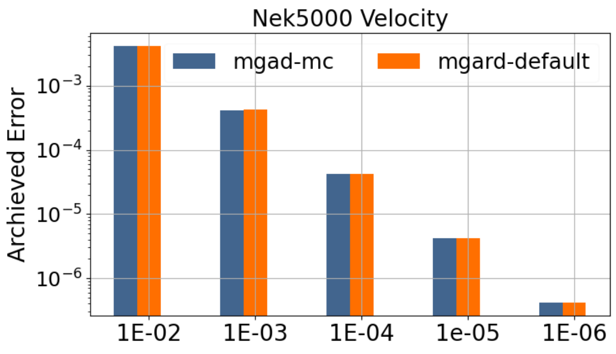


Fig. 6: Compression error analysis, with x-axis representing the requested/input error bound and y-axis representing the archived relative root-of-mean-square error measured in lossy compressed data. The evaluation was conducted using Nek5000-turbPipe's velocity field.

airfoil datasets, respectively. Different variables from the same the dataset (e.g., density, pressure, and three velocities in VKI turbine blade) share one set of parameter settings. In this way, the cost of grid construction can be amortized, not counting towards overhead. Several general trends were noted. First, the multi-component compression (blue curves) almost always outperforms the serialization approach (orange curves). Second, the improvement ratio is not constant but generally larger at high error ranges. This is expected as large error bounds would quantize more residual data into zeros, leading to higher improvements in combined compression ratios. Third, the improvement ratio is typically higher for more "compressible" data, where a compression ratio of $10\times$ or more can be achieved using the serialization approach under

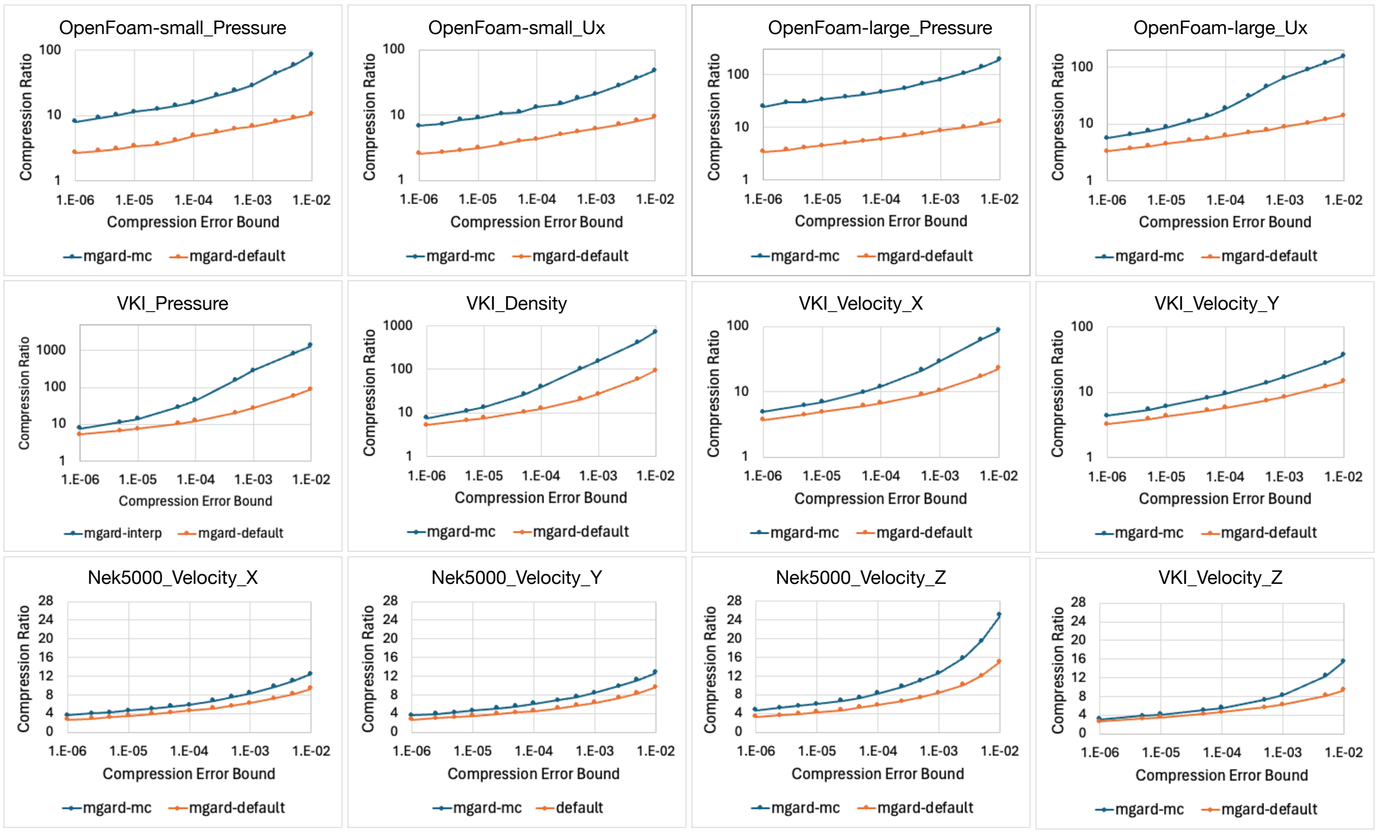


Fig. 7: Compression ratios comparison between default and the proposed multi-component approach, exemplified using MGARD as the underlying lossy compressor.

a relative small error bound. This usually indicates strong spatial correlation, which our multi-component compression can better leverage. For example, the velocity Y vs. velocity Z values in VKI turbine blade dataset and the velocity Z vs. velocity X values in Nek5000 turbPipe dataset.

Finally, we plotted in Figure 8 the improvement ratios achieved for three lossy compressors—SZ, MGARD, and ZFP—using our multi-component framework. The improvement ratio for three compressors exhibits a trend similar – the multi-component implementation delivers larger improvement ratios at high error ranges, but their level of improvement varies across datasets. We suspect the distinction is due to different de-correlation and quantization functions used by the three underlying compressors. Across the entire set of variables and error bounds, our multi-component compression consistently outperforms the serialization-based implementation, with improvement ratios ≥ 1 for all three lossy compressors, except for ZFP at $\tau = 1 \times 10^{-4}$ with the velocity Z data in VKI turbine blade dataset. We notice that VKI-velocityZ data field is extremely difficult to compress, with values following almost a random distribution with a mean of zero, as illustrated in Figure 4. ZFP exploits data redundancies through breaking data into 4^d blocks and performing orthogonal transformations. Given that the compression of residual data is conducted in 1D rather than the $4 \times 4 \times 4$ space preferred by the transformation algorithm, ZFP is expected to obtain less significant improvement than the other two compressors. Across the 12 mesh variables and five error tolerance ranges, MGARD, SZ, and ZFP achieve an average improvement ratio of $3.5\times$, $3.1\times$, and $2.3\times$ respectively.

D. *Runtime overhead*

Our evaluation excludes the cost of mesh-to-grid translation, as the resulting map can be reused across multiple timesteps and data fields, providing that the mesh structure remains unchanged. Hence, the overhead of the multi-component compression comes from the compression of grid approximation, mesh-to-grid data interpolation (i.e., $f(.)$), and the back interpolation (i.e., $g(.)$) for residual calculation. Given $f(.)$ and $g(.)$, the computation overhead is closely linked to the number of grid points used for interpolation, which is controlled by the hyperparameter of grid spacing. We therefore evaluate the overhead of multi-component compression as a function of grid spacing and plot the results in Figure 9. We denote the *overhead* as $(\mathrm{T}_{\mathrm{mc}} - \mathrm{T}_{\mathrm{default}})/\mathrm{T}_{\mathrm{default}}$, where $\mathrm{T}_{\mathrm{default}}$ represents the time spent on compressing using the same underlying compressor through the serialization approach. For grid spacing of $10\%, 30\%$, and 40%, which are parameters used for the evaluations conducted in Section V-C, the computational overhead ranged from 26% to 101%. This overhead can be compensated considering the speedup for I/O and storage

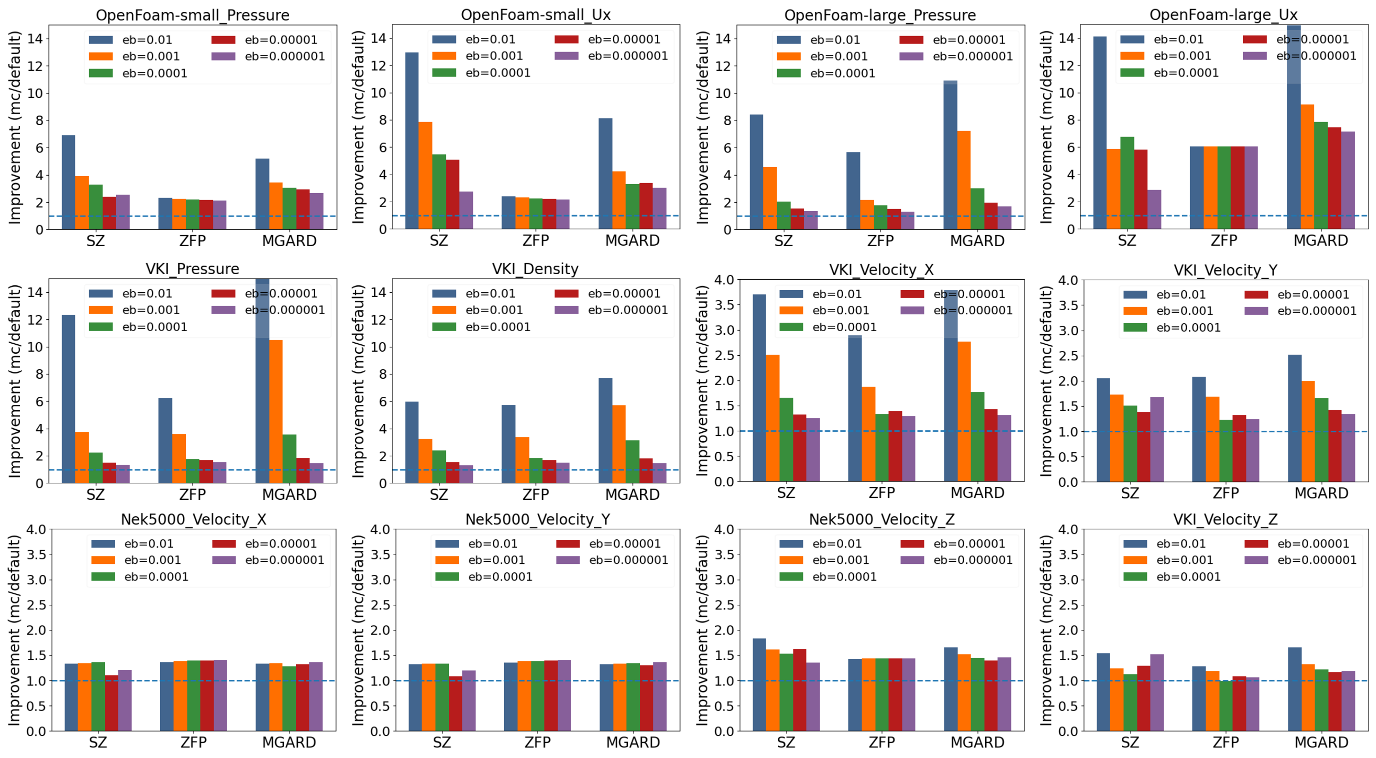


Fig. 8: Improvement ratios achieved by the proposed multi-component approach, demonstrating using three lossy compressors across different error bounds, compared to the serialization-based compression conducted through the same underlying compressors.

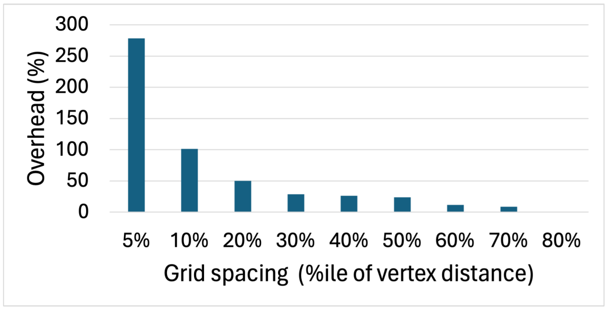


Fig. 9: Overhead of throughput performance as a function of the grid spacing.

savings with the greatly reduced data size obtained using our approach. Users can also trade off the compression ratios for smaller computational overhead when throughput is a major concern.

VI. Conclusion

In this paper, we develop a multi-component, error-bounded compression framework tailored for scientific data on unstructured meshes, addressing the limitations of existing methods that primarily target rectilinear grid data. Our framework interpolates mesh data onto a rectilinear grid, then independently compresses both the grid representation and the approximation errors on meshes (i.e., residuals). Reconstruction is achieved by combining the decompressed grid data and mesh residuals at the original mesh vertices. Our solution is effective and general, seamlessly integrating into existing I/O and compression pipelines to enhance the compression of unstructured mesh data with minimal code changes. Extensive experiments using OpenFoam synthesized and real simulation datasets demonstrate that our approach achieves, on average, a $2.3 - 3.5\times$ improvement in compression ratios when used with three state-of-the-art lossy compressors, within an error bound range of $1 \times 10^{-6} - 1 \times 10^{-2}$. In the future, we plan to investigate potential improvements offered by using different interpolation functions (i.e., $f(.)$ and $g(.)$ as discussed in the paper) and assess their impact on throughout. We will also evaluate the combined time spent on compression and the I/O of reduced data.

Acknowledgment

This research was supported by the SIRIUS-2 ASCR research project, the Scientific Discovery through Advanced Computing (SciDAC) program, specifically the RAPIDS-2 SciDAC institute, and the GE-ORNL CRADA data reductoin project. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility.

References

[1] M. A. Park, A. Loseille, J. Krakos, T. R. Michal, and J. J. Alonso, “Unstructured grid adaptation: status, potential impacts, and recommended

investments towards cfd 2030," in *46th AIAA fluid dynamics conference*, 2016, p. 3323.

[2] R. Sakai, D. Sasaki, and K. Nakahashi, "Parallel implementation of large-scale cfd data compression toward aeroacoustic analysis," *Computers & Fluids*, vol. 80, pp. 116–127, 2013.

[3] "General electric research on olcf's summit," https://www.olcf.ornl.gov/2020/08/10/sparks-fly-in-marriage-of-ges-genesis-code-and-the-summit-supercomputer/, accessed: 2024-05-27.

[4] Q. Gong, J. Chen, B. Whitney, X. Liang, V. Reshniak, T. Banerjee, J. Lee, A. Rangarajan, L. Wan, N. Vidal *et al.*, "Mgard: A multigrid framework for high-performance, error-controlled data compression and refactoring," *SoftwareX*, vol. 24, p. 101590, 2023.

[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] ——, "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2146–A2171, 2019.

[7] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE, 2016, pp. 730–739.

[8] 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.

[9] 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.

[10] 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.

[11] P. Lindstrom, "Error distributions of lossy floating-point compressors," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2017.

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

[13] 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.

[14] J. Lee, A. Rangarajan, and S. Ranka, "Nonlinear-by-linear: Guaranteeing error bounds in compressive autoencoders," in *Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing*, 2023, pp. 552–561.

[15] 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.

[16] C. Ren, X. Liang, and H. Guo, "A prediction-traversal approach for compressing scientific data on unstructured meshes with bounded error," *arXiv preprint arXiv:2312.06080*, 2023.

[17] D. El-Rushaidat, R. Yeh, and X. M. Tricoche, "Accurate parallel reconstruction of unstructured datasets on rectilinear grids," *Journal of Visualization*, vol. 24, pp. 787–806, 2021.

[18] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the unstructured case," *SIAM Journal on Scientific Computing*, vol. 42, no. 2, pp. A1402–A1427, 2020.

[19] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. R. Pugmire, M. Wolf, N. Podhorszki, and S. Klasky, "Mgard+: Optimizing multilevel methods for error-bounded scientific data reduction," *IEEE Transactions on Computers*, 2021.

[20] 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.

[21] 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.

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

[23] "Zfp: Fast, accurate data compression for modern supercomputing applications," https://ipo.llnl.gov/sites/default/files/2023-08/Final_zfp-rnd-100-award.pdf/, accessed: 2024-05-27.

[24] M. Salloum, N. D. Fabian, D. M. Hensinger, J. Lee, E. M. Allendorf, A. Bhagatwala, M. L. Blaylock, J. H. Chen, J. A. Templeton, and I. Tezaur, "Optimal compressed sensing and reconstruction of unstructured mesh datasets," *Data Science and Engineering*, vol. 3, pp. 1–23, 2018.

[25] M. Salloum, N. Fabian, D. M. Hensinger, and J. A. Templeton, "Compressed sensing and reconstruction of unstructured mesh datasets," *arXiv preprint arXiv:1508.06314*, 2015.

[26] Y. Liu, Y. Wang, L. Deng, F. Wang, F. Liu, Y. Lu, and S. Li, "A novel in situ compression method for cfd data based on generative adversarial network," *Journal of Visualization*, vol. 22, pp. 95–108, 2019.

[27] Y. Zhang, H. Guo, L. Shang, D. Wang, and T. Peterka, "A multi-branch decoder network approach to adaptive temporal data selection and reconstruction for big scientific simulation data," *IEEE Transactions on Big Data*, vol. 8, no. 6, pp. 1637–1649, 2021.

[28] M. Berger and I. Rigoutsos, "An algorithm for point clustering and grid generation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 5, pp. 1278–1286, 1991.

[29] D. Wang, J. Pulido, P. Grosset, J. Tian, J. Ahrens, and D. Tao, "Analyzing impact of data reduction techniques on visualization for amr applications using amrex framework," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 263–271.

[30] H. Luo, J. Wang, Q. Liu, J. Chen, S. Klasky, and N. Podhorszki, "zmesh: Exploring application characteristics to improve lossy compression ratio for adaptive mesh refinement," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2021, pp. 402–411.

[31] Y. Li, H. Luo, F. Li, J. Wang, and K. Li, "Lamp: Improving compression ratio for amr applications via level associated mapping-based preconditioning," *IEEE Transactions on Computers*, 2023.

[32] H. Jasak, "Openfoam: Open source cfd in research and industry," *International Journal of Naval Architecture and Ocean Engineering*, vol. 1, no. 2, pp. 89–94, 2009.

[33] D. Jang, R. Jetli, and S. Acharya, "Comparison of the piso, simpler, and simplec algorithms for the treatment of the pressure-velocity coupling in steady flow problems," *Numerical Heat Transfer, Part A: Applications*, vol. 10, no. 3, pp. 209–228, 1986.

[34] S. Rezaeiravesh, R. Vinuesa, and P. Schlatter, *A statistics toolbox for turbulent pipe flow in Nek5000*. KTH Royal Institute of Technology, 2019.

[35] E. Merzari, P. Fischer, M. Min, S. Kerkemeier, A. Obabko, D. Shaver, H. Yuan, Y. Yu, J. Martinez, L. Brockmeyer *et al.*, "Toward exascale: overview of large eddy simulations and direct numerical simulations of nuclear reactor flows with the spectral element method in nek5000," *Nuclear Technology*, vol. 206, no. 9, pp. 1308–1324, 2020.

[36] T. Arts and M. Lambert de Rouvroit, "Aero-thermal performance of a two-dimensional highly loaded transonic turbine nozzle guide vane: A test case for inviscid and viscous flow computations," 1992.

[37] 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*, 2022.

[38] 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.

[39] S. Atchley, C. Zimmer, J. R. Lange, D. E. Bernholdt, V. G. M. Vergara, T. Beck, M. J. Brim, R. Budiardja, S. Chandrasekaran, M. Eisenbach *et al.*, "Frontier: exploring exascale the system architecture of the first exascale supercomputer," in *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2023, pp. 1–16.