Fast Algorithms for Scientific Data Compression

Tania Banerjee*, Jaemoon Lee*, Jong Choi†, Qian Gong†, Jieyang Chen‡,
Scott Klasky†, Anand Rangarajan*, Sanjay Ranka*
*University of Florida, USA
†Oak Ridge National Laboratory, USA
‡University of Alabama at Birmingham, USA

***Abstract*—Many scientific simulations and experiments generate terabytes to petabytes of data daily, necessitating data compression techniques. Unlike video and image compression, scientists require methods that accurately preserve primary data (PD) and derived quantities of interest (QoIs). In our previous work, we demonstrated the effectiveness of hybrid compression techniques that combine machine learning with traditional approaches. This paper presents innovative computational techniques aimed at expediting the compression pipeline. Our experiments, conducted on two distinct platforms with a large-scale XGC-based fusion simulation, demonstrate that the overhead incurred by these new approaches is less than one percent of the computational resources needed for the simulation.**

***Index Terms*—Data compression, Machine learning, High-performance computing**

I. Introduction

Scientific applications are producing data at an unprecedented rate, with both volume and velocity expanding rapidly, outpacing advancements in storage, computation, and network capacities [16]. Scientists conduct large-scale simulations to compute derived quantities from primary data, and it is crucial for them to find compression techniques that maintain low or bounded errors on these derived quantities. Bounding the derived quantities or quantities of interest (QoIs) is essential to instill confidence in the data compression methods scientists employ.

This paper presents a multistage algorithmic pipeline for data compression, with the key stages outlined in Figure 1. While the results are showcased for the XGC simulation code [10], designed for large-scale nuclear fusion simulations, the findings are relevant for an extensive array of kinetic codes utilizing a two-dimensional velocity-space mesh and a three-dimensional configuration space mesh. The application also extends to other particle-in-cell plasma physics codes. These principles are equally pertinent to gyrokinetic codes commonly employed in magnetic fusion research. The fundamental concepts outlined in this paper guarantee the preservation of QoIs with floating-point precision, a critical factor for gaining acceptance of compression techniques within the scientific community.

The contributions of this paper to the data compression pipeline in [4] are as follows:

1) Utilizing GPU-based computation, we have finely tuned MGARD, reducing overhead by a factor of ten. This substantial improvement dramatically enhances the overall computational efficiency of the compression pipeline. Unlike previous work [11] that experienced performance degradation with fixed hierarchies, we optimized the multilevel decomposition on the GPU, enabling a self-adaptable and scalable hierarchy for parallel computing. This optimization, along with others, reduced the time requirements by a factor of 4.6.
2) Utilization of L2 Error and SVD for Lagrange Multiplier Computation: We developed a new mathematical approach that utilizes L2 error instead of Bregman divergence, resulting in a direct computation of the Lagrange multipliers. Additionally, we demonstrated the feasibility of using SVD for the computation. These improvements reduced the time required for multiplier computation by a factor of 8. This computation represents a significant portion of the overall computational effort for data compression.
3) Coupled Compression and Simulation Pipeline: While earlier work [4] primarily assessed the computational overhead of compression by post-processing simulation data, our current research introduces a pipeline that enables the concurrent execution of compression and simulation tasks. This pipeline entails distinct sets of simulation and compression nodes operating in tandem. As a result, compression for the previous time step, along with the requisite data writing, is completed in tandem with the simulation progressing to the current time step.

The experimental results are obtained from performance tests on two high-performance supercomputers: the Summit system at ORNL [19] and the Perlmutter system at NERSC. Our findings demonstrate that by utilizing less than one percent of the resources required for data generation, we can achieve a data compression ratio of two orders of magnitude. Furthermore, our approach ensures accurate preservation of the primary data and achieves near-floating-point precision for the QoIs.

In comparison to previous work [4], the new combined approach for primary data compression and post-processing is approximately three times faster, providing improved efficiency in the compression pipeline.

Additionally, we employed SZ and ZFP compression techniques while ensuring comparable bounds for primary data and QoIs. Table I showcases the maximum compression achieved using SZ and ZFP. In contrast, our approach achieves an order of magnitude higher compression for the same error levels.

Contact e-mail: sranka@ufl.edu.

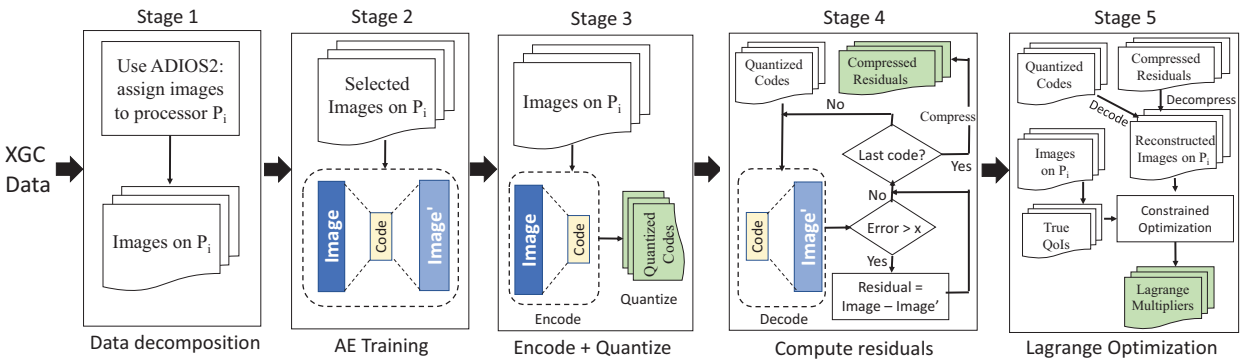


Fig. 1: Data compression pipeline for each processor P_i. Stage 1 uses domain decomposition to divide the dataset into small subdomains. Each processor processes a subdomain independently. An autoencoder (AE) is used in Stage 2 for data compression. This is quantized in Stage 3 to further reduce its size. A lossy compressor that guaranteed primary data error is used in Stage 4 on portions of the dataset. To provide QoI guarantees, Stage 5 uses a novel approach based on nonlinear projection.

QoI Error	Compression Level		
	SZ	ZFP	Our pipeline
10^{-8}	3.35	8.76	232
10^{-15}	-	-	168

TABLE I: A comparison of compression ratios between SZ and ZFP for low QoI errors. Our approach achieves an order of magnitude higher compression than these techniques at the same errors.

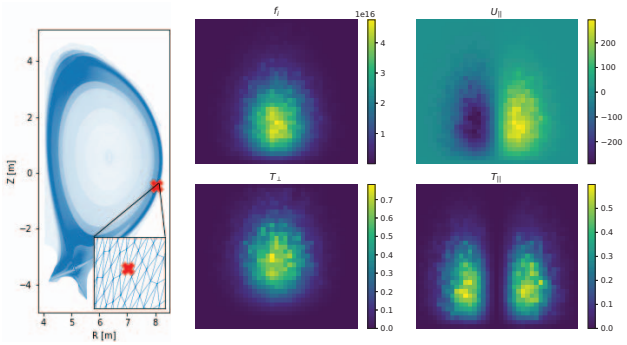


Fig. 2: XGC ITER mesh and the F data and their derived quantities (on the right). The marker shows the cell for which F data was collected. Data for each cell corresponds to a two-dimensional tensor.

The rest of the paper is organized as follows. Section II presents the background. We also provide a brief introduction to the XGC application. Section III describes our compression pipeline and novel aspects of GPU processing and post-processing algorithm. Section III elaborates on our comprehensive data workflow design, encompassing both data generation and real-time data compression. Section V presents our experimental results for different levels of compression and accuracy. Section VI presents related work. Conclusions are provided in Section VII.

II. Background

In this section, we provide a concise overview of the XGC application and introduce MGARD [1, 2, 3], a hierarchical decomposition algorithm that plays a crucial role in ensuring accuracy for primary data.

A. XGC

The X-point gyrokinetic code (XGC) [10, 22] is a fusion physics code designed to simulate plasma behavior in magnetic confinement fusion devices, specifically those known as tokamaks. XGC is based on a particle-in-cell (PIC) method that uses many computational particles that move at every time step. Particles are interpolated onto distinct mesh points by utilizing gyrokinetic equations of motion to resolve the particle distribution functions. This function, "F", in the five-dimensional physical space characterizes the velocities and directions of particle motion. Handling these "F" data in XGC presents a significant challenge due to their size: it can grow from a few gigabytes to several terabytes per simulation iteration, contingent on the resolution of the simulated physics space.

In this paper, we use the ITER simulation [9] of XGC to demonstrate our pipeline. ITER is one of the world's largest tokamaks currently under construction in France. XGC simulations for ITER produce over 80 GB of data every iteration (45 to 60 seconds), making compression necessary to store and accurately reconstruct the data. There are four QoIs that are of interest to the scientists: density, parallel flow velocity, perpendicular temperature, and parallel temperature, which are denoted by n, $u_{\|}$, $T_{\perp}$, and $T_{\|}$, respectively. In

addition, two more QoIs are obtained by manipulating these four.

Since XGC data are available on a discrete mesh. The corresponding constraints are also discretized. For each (x, t), where x corresponds to the location and t corresponds to time, the four QoIs of the 2D histogram $\boldsymbol{f}$ can be written in a linearized fashion as follows:

$$\sum_{i,j} f_{ij} \text{vol}_{ij} = n^c, \tag{1}$$

$$\sum_{i,j} f_{ij} \text{vol}_{ij} v_j^{\parallel} = n^c u_{\parallel}^c, \tag{2}$$

$$\frac{1}{2} m \sum_{i,j} f_{ij} \text{vol}_{ij} \left(v_i^{\perp}\right)^2 = n^c T_{\perp}^c, \tag{3}$$

$$\frac{1}{2} m \sum_{i,j} f_{ij} \text{vol}_{ij} \left(v_j^{\parallel}\right)^2 = n^c \left(T_{\parallel}^c + \frac{1}{2} m \left(u_{\parallel}^c\right)^2\right), \tag{4}$$

where m, vol_{ij}, $v_j^{\parallel}$, and $v_j^{\perp}$ represent particle mass, mesh node volume, parallel velocity, and perpendicular velocity, respectively. These quantities can be obtained from the XGC metadata.

The four QoI constraints can be expressed in the form $B\boldsymbol{f} = \boldsymbol{c}$, where $B = \left[B^1, B^2, B^3, B^4\right]^T$ with $B_{ij}^1 = \text{vol}_{ij}$, $B_{ij}^2 = \text{vol}_{ij} v_j^{\parallel}$, $B_{ij}^3 = \frac{1}{2} m \text{vol}_{ij} \left(v_i^{\perp}\right)^2$, $B_{ij}^4 = \frac{1}{2} m \text{vol}_{ij} \left(v_j^{\parallel}\right)^2$, and $\boldsymbol{b} = \left[n^c, n^c u_{\parallel}^c, n^c T_{\perp}^c, n^c \left(T_{\parallel}^c + \frac{1}{2} m \left(u_{\parallel}^c\right)^2\right)\right]^T$.

Although B is specific to each instance, it has to be computed only once in the first iteration, as it remains constant throughout the simulation. For this reason, we do not include the time in the computational cost for compression.

B. MGARD-based Compression

To effectively reduce the residuals of the autoencoder (AE), it is crucial to utilize a compressor capable of achieving substantial compression ratios while maintaining control over the induced compression errors. Among the state-of-the-art lossy compressors are MGARD [1, 2, 3], SZ [32, 34], and ZFP [26]. Our choice of MGARD is based on its high throughput on GPUs, the ability to tightly control the L^2 norm of errors, and its parameterization options that facilitate the preservation of either higher or lower frequency components [18, 23, 25].

MGARD provides error-controlled lossy compression using a multigrid representation. It transforms floating-point scientific data into multilevel coefficients, followed by quantization and lossless encoding processes, resulting in a self-describing compressed buffer. The decomposition stage aims to convert the input data, denoted as `u`, into a highly compressible representation, referred to as `u_mc`. For detailed information on MGARD, we refer readers to [2, 3]. The decompression algorithm is an inverse process of compression. MGARD supports diverse data topologies, including uniform and non-uniform and structured and unstructured, and it is portable across different computing architectures.

III. Data Compression Pipeline

This work focuses on simulations conducted across multiple cross-sections, referred to as planes, in a toroidal tokamak. Each plane is discretized using a triangular mesh overlay, where each mesh node is considered a cell in this paper. Considering the discretized components of speeds along the parallel and perpendicular directions to the magnetic field, a histogram is calculated using all the particles within each cell. Because the compression is performed independently on each cell, there is no need for communication during the cell-by-cell computation. The particle histograms in the two-dimensional velocity space represent the data that needs to be compressed. This two-dimensional tensor is generated for each cell in every plane.

The key stages of the compression pipeline are now described in detail (see Figure 1).

Stage 1: Domain decomposition: The input cells are divided into smaller chunks and distributed among the nodes of the parallel machines. In line with the approach discussed in [4], we employ a column-wise decomposition of the two-dimensional array (number of planes times number of cells per plane) to leverage spatial homogeneity effectively. This decomposition strategy reduces the training time for the autoencoder, which is built separately on each node in the subsequent step.

Since each subdomain can be compressed independently, we can achieve a high level of parallelism, which also contributes to the overall scalability of the system.

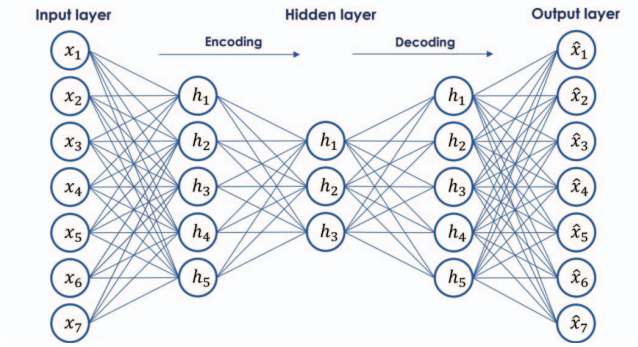


Fig. 3: A multilayer autoencoder. The bottleneck layer represents the compressed version of the input.

Stage 2: Training an Autoencoder: The autoencoder (AE) transforms the data into a lower-dimensional space using a bottleneck layer. By employing one or more hidden layers, the AE learns to map the input data and generate an output with minimal loss. The final hidden layer is the bottleneck layer, effectively learning a compressed representation of the input data.

The compression ratio is determined by the ratio of the size of the input layer to the bottleneck layer. Additionally, further compression can be achieved by discretizing the bottleneck layer. The primary objective of AE training is to optimize the weights in a way that minimizes the error between the input

and output data. Typically, mean squared error is used as the loss function in this process.

Stage 3: Encoding and Quantization: The autoencoder model (refer to Figure 3) reduces the size of each cell through its compression process. The output coefficients from this compression form the latent space, which captures the compressed representation of the input data.

To further reduce data size, we utilize quantization techniques to represent the latent space. In our approach, we employ a product quantization (PQ) scheme [20]. This scheme involves applying k-means clustering in multiple subspaces, where each subspace represents a specific portion of the latent representation. By employing this approach, the overall representation can be viewed as a Cartesian product of these smaller subspaces. The number of bits allocated for representing codewords in the codebook of a PQ dictionary determines the size of the quantized data.

Stage 4: Ensuring Accurate Error Bounds on Primary Data through Residual Post-processing

Utilizing truncated basis function representations, MGARD successfully achieves the assurances above, elaborated in Section II-B. In our pursuit of enhancing the efficiency of the MGARD compression pipeline with minimal overhead, we aim to expedite the process using GPU acceleration. There are three major components of the existing MGARD compression pipeline: multilevel decomposition, levelwise linear quantization, and lossless compression. Previously, the optimization efforts had primarily focused on the multilevel decomposition phase for GPU acceleration[11]. However, in this endeavor, we build a comprehensive GPU-based compression pipeline, which includes a quantization design and a lossless compression kernel.

In the process of levelwise linear quantization, we assign distinct quantization bin sizes to multilevel coefficients at different levels. The quantization operation is represented as $quantized[i] = round_to_nearest(\frac{mc[i]}{quantizer[l]})$, where $mc[i]$ and $quantizer[i]$ are the multilevel coefficient and quantizer at level l. To eliminate redundant quantizer calculations, we precompute the quantizer values for each level on the CPU and cache them into shared memory on the GPU before the quantization step. In the lossless compression step, we leverage the normal distribution of quantized multilevel coefficients and employ Huffman compression for effective data reduction. Drawing from the research by [33], we implement the parallel Huffman codebook generation algorithm [30] on GPUs. To balance encoding performance and compression ratio, we introduce an adaptive codebook storage strategy to enhance codebook accessibility. In cases where the codebook is sufficiently small, we cache it in GPU-shared memory. On the contrary, for data that prove challenging to compress, we dynamically construct and store the larger codebook in GPU global memory. Leveraging our GPU-based levelwise quantization and Huffman lossless compression kernels, we create an end-to-end MGARD compression system on GPUs.

To optimize the compression of XGC data using MGARD, we also present an enhancement to the multilevel decomposition process. In existing parallel multilevel decomposition [11], the degree of parallelism is constrained by the smallest dimension of the input data. This limitation arises from parallelizing the process of solving tridiagonal linear systems, which depends on the total number of linear systems. In cases where the dataset exhibits significantly smaller dimensions, the solving of tridiagonal linear systems can lead to severely degraded performance. For XGC data, which exhibits a shape of $V_x \times N \times V_y$, with V_x and V_y representing the dimensions of the particle histogram, this constraint becomes apparent. The number of nodes denoted as N, can vary from hundreds of thousands to millions. In response, we introduce a dimension decomposition technique, enabling us to decompose the data along the taller dimension, ensuring that the decomposed datasets maintain similar dimensions. With each decomposed dataset amenable to parallel processing, this technique significantly.

Stage 5: Guaranteeing QoI Error Bounds Even if the PD is forced to satisfy error bounds, this does not guarantee QoI preservation due to the holistic properties of the QoI. However, since the QoI constraints in (1)—(4) are linear in the histogram values f, we can perform standard constrained optimization with linear equality constraints to satisfy them. As shown in (5), the objective function $d : \mathbb{R}^N \to \mathbb{R}$ minimizes a suitable norm based on the difference with the original PD while setting up equality constraints $h : \mathbb{R}^N \to \mathbb{R}^M$ on M different QoI. The loss function $d(\boldsymbol{f}, \boldsymbol{f}_R)$ quantifies the distance between the two vectors. Here, $\boldsymbol{f}$ is the modified histogram, while $\boldsymbol{f}_R$ is the decoder-based reconstruction obtained in Stage 4.

$$\begin{aligned} &\text{minimize} \quad d(\boldsymbol{f}, \boldsymbol{f}_R) \\ &\text{subject to} \quad h(\boldsymbol{f}) = 0. \end{aligned} \tag{5}$$

A standard approach for solving the optimization problem in (5) with linear constraints involves the application of the method of Lagrange parameters [6]. The extrema of a real function d (which include maxima, minima and saddle points) in a D-dimensional space which satisfies M constraints $h_k(f_1, f_2, \ldots, f_D)$ for $k \in \{1, \ldots, M\}$ can be obtained via the Lagrange parameter theorem: at extrema, the gradient of d is a linear combination (using a set of Lagrange parameters) of the constraint gradients of d. The Lagrange parameters $\{\lambda_k\}$ are the coefficients of this linear combination:

$$\nabla d(f_1, f_2, \ldots, f_D) = \sum_{k=1}^{M} \lambda_k \nabla h_k(f_1, f_2, \ldots, f_D), \tag{6}$$

where the gradients ∇d and ∇h_k lie in a D-dimensional space. We now write the Lagrangian for the problem in (5):

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\lambda}) = d(\boldsymbol{f}, \boldsymbol{f}_R) + \boldsymbol{\lambda}^T h(\boldsymbol{f}), \tag{7}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_M]$ is a Lagrange multiplier vector. The optimal solution of the Lagrange multipliers can be obtained using (6). A new reconstruction $\boldsymbol{f}$ can be obtained using the Lagrange parameters which preserves the QoIs, while simultaneously minimizing the distance to the original PD

reconstruction $\boldsymbol{f}_R$.

In previous work [5], the *Bregman divergence* for a convex function (actually an entropy barrier function) [7, 8] was chosen as the objective in (7). The Bregman divergence-based optimization used convex programming since direct solutions for $\boldsymbol{\lambda}$ are unavailable. Consequently, using duality and Newton's method [14], an iterative procedure was used to obtain an optimal dual value.

Using a squared ℓ_2 norm as the distance measure in (7), we show that a direct solution of $\boldsymbol{\lambda}$ can be obtained. Let the original histogram be $\boldsymbol{f}_T$. The true QoI are first obtained via $\boldsymbol{c} = B\boldsymbol{f}_T$. With the set of QoI constraints $B\boldsymbol{f} = \boldsymbol{c}$ in hand, the constraint satisfaction Lagrangian is

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\lambda}) = \frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_R)^T(\boldsymbol{f} - \boldsymbol{f}_R) + \boldsymbol{\lambda}^T(B\boldsymbol{f} - \boldsymbol{c}). \quad (8)$$

The first order Karush–Kuhn–Tucker (KKT) conditions are

$$\boldsymbol{f} - \boldsymbol{f}_R + B^T\boldsymbol{\lambda} = \boldsymbol{O}, \quad (9)$$

$$B\boldsymbol{f} = \boldsymbol{c}. \quad (10)$$

Because we seek $B\boldsymbol{f} = \boldsymbol{c}$, we multiply by B to get

$$\boldsymbol{c} - B\boldsymbol{f}_R + BB^T\boldsymbol{\lambda} = \boldsymbol{O} \Rightarrow \boldsymbol{\lambda} = \left(BB^T\right)^{-1} B(\boldsymbol{f}_R - \boldsymbol{f}_T). \quad (11)$$

Substituting $\boldsymbol{\lambda}$ in (9), we get

$$\boldsymbol{f}_C = \boldsymbol{f}_R - B^T\boldsymbol{\lambda} = \boldsymbol{f}_R - P(\boldsymbol{f}_R - \boldsymbol{f}_T), \quad (12)$$

where $P = B^T\left(BB^T\right)^{-1}B$. The obtained $\boldsymbol{f}_C$ is a vector space projection of $\boldsymbol{f}_R$ onto the subspace spanned by constraints. We have eschewed the satisfaction of non-negativity constraints of each element of $\boldsymbol{f}_C$ during the constraint satisfaction process. Nevertheless, the improved computational complexity makes it suitable for QoI preservation.

The computational overhead of (12) can be further reduced by choosing an order of matrix multiplication that is efficient in the total number of operations and use of SVD. The XGC node for ITER is of size 33×37, resulting in a projection matrix of size (1221×1221). Additionally, this P requires an inverse operation, which can be avoided, as explained below. Instead of explicitly performing the complete set of matrix multiplications and inversion, we use the singular value decomposition of B to simplify the constraint satisfaction process drastically. Let the truncated SVD of B be $U\Sigma V^T$. The columns of U are the left singular vectors, Σ is a diagonal matrix with singular values, and the rows of V^T are the right singular vectors. Using this, we can simplify P as follows:

$$\begin{aligned} P &= B^T\left(BB^T\right)^{-1}B \\ &= V\Sigma U^T\left(U\Sigma V^T V\Sigma U^T\right)^{-1}U\Sigma V^T \\ &= V\Sigma\left(\Sigma^T\Sigma\right)^{-1}\Sigma V^T \\ &= VV^T. \end{aligned} \quad (13)$$

The reconstruction $\boldsymbol{f}_C$, after QoI post-processing, can now be expressed as

$$\boldsymbol{f}_C = \boldsymbol{f}_R - P(\boldsymbol{f}_R - \boldsymbol{f}_T) = \boldsymbol{f}_R - V\left(V^T(\boldsymbol{f}_R - \boldsymbol{f}_T)\right). \quad (14)$$

In the above representation, the order of the computation, as indicated by the parentheses, requires the least amount of computation for the matrix sizes under consideration.

IV. Coupled Compression and Simulation Pipeline

Coupled or co-simulation [12] is a prevalent technique in scientific research, allowing for exploring multiphysics and multiscale simulations. This approach involves running multiple applications simultaneously, either in a loosely connected or tightly synchronized manner, to achieve a unified result.

Incorporating real-time data compression into synchronized workflows is useful because the core objectives of scientific applications can be realized without being impeded by the I/O operations on the generated data. To realize this, we leverage ADIOS [17] that aids in efficiently coordinating data transfer in HPC environments. Our strategy involves real-time data compression in a designated staging area, using dedicated supplementary nodes while XGC operates concurrently. This approach allows us to offload the data compression process.

There are two additional overheads in the workflow: the data transit time, which is nominal, and the cost of compression. We assume that the latter is less than the data generation time of each step of the XGC simulation.

For data transfer between the simulation and the compression nodes within the ADIOS framework, we have tested the two available methods; the first one is traditional and uses the file system, while the second one is more efficient and employs a direct memory copy using Sustainable Staging Transport (SST) engine.

V. Experiments

This section presents the experimental results of compressing the simulation data of ITER through our pipeline. Section V-A provides a comprehensive overview of the experimental setup, including the platform configuration, a concise description of the data to be compressed, the input/output mechanism, and the properties and reference points utilized for evaluating the compression performance and accuracy. Section V-B show trends of primary data (PD) and quantity of interest (QoI) errors concerning compression ratio for ions and electrons in the ITER simulation, leading to the final Section V-C, which presents the performance results.

A. Experimental Setup

Our experiments utilize US-DOE supercomputers: Summit at ORNL and Perlmutter at NERSC. The Summit node consists of two IBM POWER9 CPUs with 512 GB of DDR4 memory, along with six NVIDIA Volta GPUs, each featuring 16 GB of HBM2 memory. These nodes are interconnected via a dual-rail Mellanox EDR InfiniBand. On the other hand, each GPU-accelerated node of Perlmutter features a single AMD EPYC 7763 CPU, equipped with 256 GB of DDR4 memory, along with four NVIDIA A100s, each carrying 40-GB HBM2 memory and connected via NVLink-3. Each node in this system also includes up to 4 NICs, one for each GPU.

The HPE Slingshot 11 interconnect facilitates communication between all nodes.

To handle large-scale input/output (I/O) operations, we employ an IBM Spectrum Scale GPFS parallel file system on Summit and an all-flash LUSTRE parallel filesystem on Perlmutter. For the compression of ITER data in this study, we utilize 8 to 16 Summit nodes, utilizing six GPUs per node, resulting in 48 to 96 GPUs, with each process allocated one GPU. On Perlmutter, we use 8 to 16 nodes, utilizing four GPUs per node, resulting in 32 to 64 GPUs.

The XGC ITER simulation was performed on 1024 Summit nodes (equipped with 6144 GPUs) and 512 Perlmutter nodes (equipped with 2048 GPUs). The key output of each simulation step is the particle distribution function represented by F. In the XGC ITER run, the behavior of 34 billion virtual electrons and ions is emulated. The size of the F data for both electrons and ions is approximately 81 GB per step, corresponding to a three-dimensional mesh representing the torus. The first dimension of the mesh is equal to the number of planes (8). Each plane corresponds to $1.1\text{x}10^6$ cells. Each cell corresponds to a velocity space tensor of size 33-by-37.

ADIOS-2 [17, 29] was used for parallel I/O. ADIOS-2 offers APIs for consumer-producer workflows, metadata collection, transportation, and compression.

The compression quality is evaluated using a normalized root mean square error (NRMSE) metric as a relative error measure, assessing the PD and the QoI errors. The definition of NRMSE for primary data is as follows:

$$\text{NRMSE}(\boldsymbol{f}, \boldsymbol{f}_C) = \frac{\sqrt{\sum_{i=0}^{N} \|f_i - (f_C)_i\|_2^2 / N}}{\max(\boldsymbol{f}) - \min(\boldsymbol{f})}, \qquad (15)$$

where the variables $\boldsymbol{f}$ and $\boldsymbol{f}_C$ represent the original data and reconstructed data, respectively. The variable N corresponds to the number of degrees of freedom in the original data. For QoI error computation, the corresponding quantities derived from $\boldsymbol{f}$ and $\boldsymbol{f}_C$ are used. We have established the specified error bounds as 0.001 for PD errors and 0.0001 for QoI errors. Our method consistently achieves QoI errors significantly below the specified bounds. In the case of image NRMSE computation, Equation (15) is applied. The value of N is obtained by multiplying the cardinality of the velocity coordinates, where $V_x = 33$ and $V_y = 37$ for the ITER test case.

The improvement in compression ratio is directly proportional to the number of images that MGARD can encode, which requires a significantly more significant number of bits per image to guarantee PD.

The network architecture of the AE includes a single fully connected layer for both the encoder and decoder, employing linear activation functions. In order to enhance stability and reduce the model size, we transpose the model weights from the encoder to the decoder. This design choice allows us to utilize a single set of model weights. An Adam optimizer [21] with a learning rate of 0.001 was used for training. Following our previous work, the training process followed a paradigm where the model is initially trained for 100 epochs in the first step. For each subsequent step, the model is updated using only two epochs [4]. This approach is motivated by the observation that because the underlying tensors do not change substantially from one iteration to the subsequent, minimal model updates are necessary.

B. Quality and Level of Compression

At time step 170 of the ITER simulation, Figure 4 displays the trends of PD and QoI errors as a function of the compression ratio for ions and electrons. The time step 170 was chosen to present our experimental results as it is representative of instances where the compression rates were lower compared to other instances. In Figure 4, it can be observed that when the λs are stored in double precision, the error on the QoIs computed from post-processed reconstructed data is minimal (10^{-15}). However, truncating the λs to single precision increases the error to 10^{-8}. Setting the PD error at 0.001, the AE-based pipeline (Figure 1) utilizing single precision λs achieves a compression ratio of around 204 for ions and 158 for electrons. At time step 170, the compression ratio is 1.7 times higher for ions and 1.13 times higher for electrons than MGARD with post-processing.

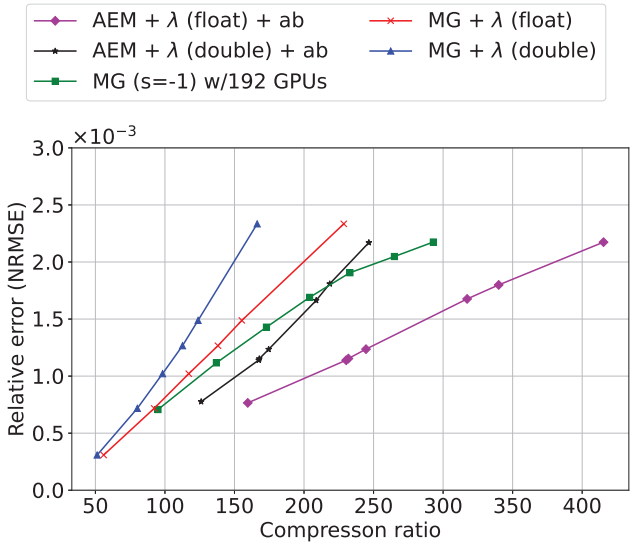
For the rest of the paper, we show results using λs in single precision for all the experimental results.

C. Performance

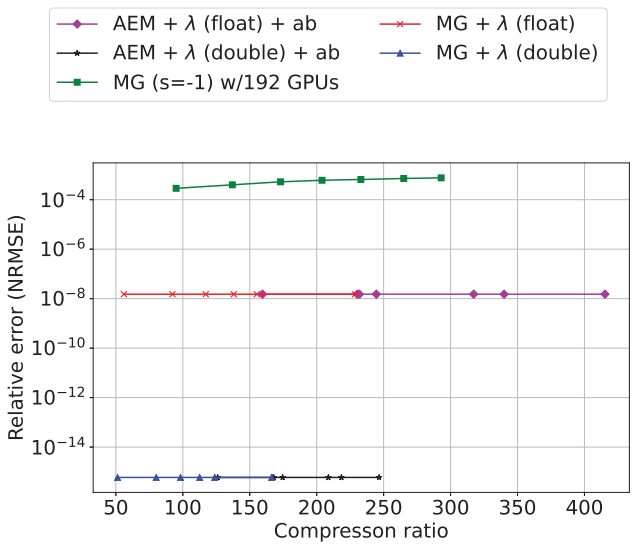
In this section, we analyze the trends of compression ratio and errors across time steps and discuss the timing and resource requirements. The ITER simulation version used in this study consists of 125 time steps, ranging from 2 to 250, with increments of 2. We also provide an overview of the results of MGARD and post-processing optimizations. Finally, we evaluate the pipeline on two different platforms.

1) Compression Ratios and Errors across Time Steps: Comparing the results of both the AE pipeline and the non-AE pipeline, Figure 5(a) presents the individual compression ratios achieved for ions and electrons at each time step. In each case, the PD error is maintained close to 0.001. Furthermore, Figure 5(b) demonstrates that the rounded PD error remains consistently near 0.001 for all scenarios. Additionally, Figure 5(c) displays the QoI errors achieved using single-precision floating-point representation, which consistently stays below 10^{-8}. The compression level is inversely proportional to variability across the images as the autoencoder cannot as effectively model the underlying dataset and requires the use of MGARD for guaranteeing PD. In this simulation, the variability increases because of the underlying turbulence. Even under this variability, we achieved a compression ratio of over 100, showing the benefits of our approach. Our approach generally does not guarantee compression ratio but the PD and QoI bounds.

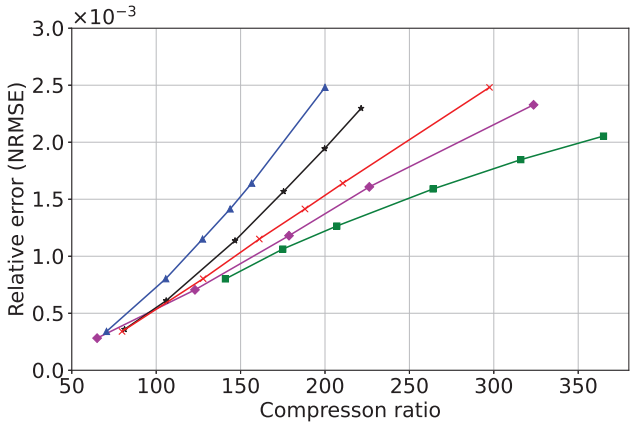
2) MGARD Optimization Results: We use 50,000 mesh points on a single plane to compare the old version of our MGARD GPU implementation with the new one. The performance of both MGARD compression pipelines is shown in Figure 6.



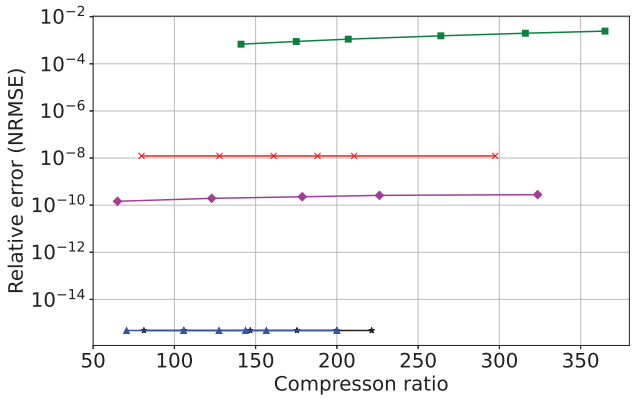
(a) PD Errors for ions



(b) QoI Errors for ions



(c) PD Errors for electrons



(d) QoI Errors for electrons

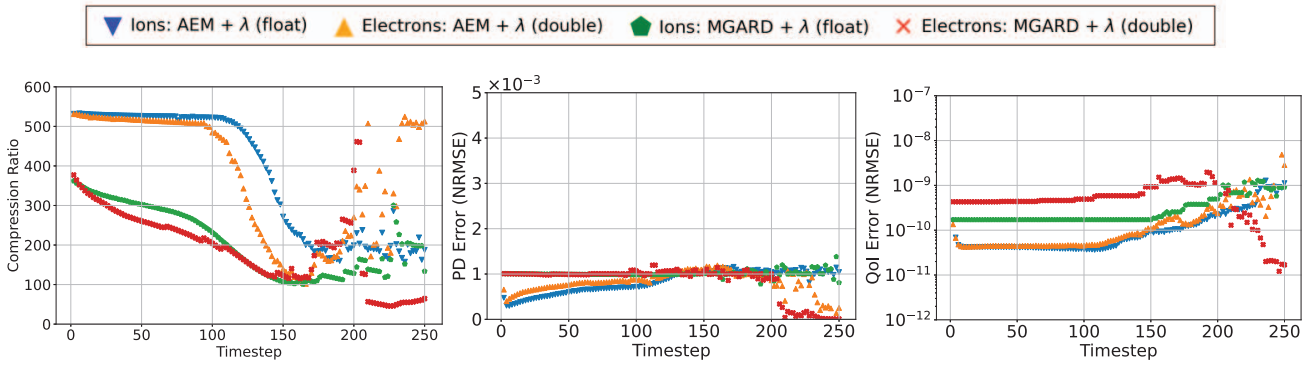
Fig. 4: NRMSE errors for different methods for variable compression ratio for ions (a) and electrons (c) for a representative simulation time step (170). QoI errors using different methods are given for ions (b) and electrons (d). We use double or a float for storing the Lagrange parameter (i.e., $\boldsymbol{\lambda}$). Post-processing reduces the error in QoIs by 6 to 10 orders of magnitude for float versus double storage space for each $\boldsymbol{\lambda}$.

Upon comparison, it becomes evident that the original decomposition and recomposition steps, lacking parallelism, account for most of the pipeline's processing time. However, with our decomposition optimization, we achieved a remarkable 5.7x acceleration in this process, translating to an overall 2.8x speedup for the entire compression pipeline. Furthermore, integrating our fused compression-decompression pipeline obviates the need for lossless decompression (as indicated by the yellow bars) in the sequence, resulting in an additional speed enhancement of 1.6x. As a result, the cumulative speedup achieved through all these optimizations for MGARD amounts to approximately 4.6x.

3) Post-processing Optimization Results: The experimental results showed that our new algorithm, Section III, was eight times faster than the iterative approach in [5].

4) Evaluations on Summit and Perlmutter: To assess the efficiency of the enhanced MGARD implementation for the AE, along with the novel post-processing scheme, we conducted experiments on both the Summit and Perlmutter supercomputers.

The performance trends, depicted in Figure 7, showcase the overall efficiency across different node configurations for online compression and post-processing on both the Summit and Perlmutter systems. We employ multiple distributed processes, each utilizing a set of resources, including a single GPU and



(a) Compression ratios (b) PD errors (c) QoI errors

Fig. 5: (a) Compression ratio for each of the ITER simulation time steps. The PD error bound is set at 10^{-3}, while QoI errors are in the range of 10^{-8}. The λs are truncated to a single precision floating point. We assume incremental training is used at each time step. (b) Achieved errors show that PD errors are within the 10^{-3} bound for each time step. (c) Achieved QoI errors show that very low errors are achieved for each time step.

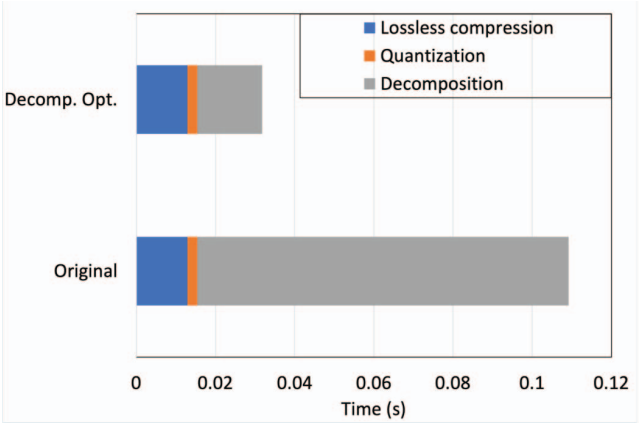


Fig. 6: Comparing the performance of the new MGARD compression pipeline. The results are presented on a single NVIDIA V100 GPU using 50,000 mesh points for different stages. These results show that our new approach requires a factor of 2.8 times lower than the previous approach.

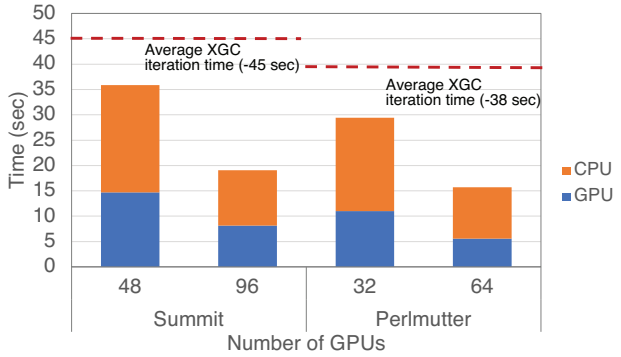


Fig. 7: Online compression performance in a staging area on Summit and Perlmutter.

multiple CPU cores, to perform parallel compression and post-processing steps shown in Figure 1.

The AE training and MGARD residual compression are executed on the GPU, while the post-processing phase primarily runs on the CPU. To enable parallel processes, we utilize MPI for parallelization across multiple nodes and OpenMP for utilizing multiple CPU cores within each process. This implementation strategy ensures efficient utilization of GPUs and all available cores on both Summit and Perlmutter.

Our findings demonstrate that the compression procedure can be efficiently carried out with just eight nodes on both Summit and Perlmutter systems, corresponding to using 48 GPUs on Summit and 32 GPUs on Perlmutter when operating an XGC simulation using 1024 nodes. Additionally, the time required for compression is less than the duration needed for a simulation step—roughly 45 seconds on Summit and 38 seconds on Perlmutter.

VI. Related Work

While lossless compression techniques [28, 13, 27] are desirable due to their ability to preserve data integrity, they often achieve limited compression ratios, making them less suitable for large-scale scientific applications.

Specific lossy techniques, such as prediction-based methods in SZ [15, 32, 24] and block transform-based methods in ZFP [26], as well as multilevel decomposition techniques in MGARD [1, 2, 3], are popular in scientific domains. However, when these techniques are applied directly, they do not offer guarantees on the quality of the output data, limiting their usefulness for downstream scientific applications. While MGARD can bound errors on linear quantities of interest (QoIs), it tends to achieve lower compression ratios (5–10) when the QoI errors need to be limited to 10. On the other hand, SZ [15, 32, 24] and ZFP [26] suffer from reduced fidelity

for the reconstructed data when the compression ratios are larger than 100.

VII. Conclusions

This paper presents a hybrid pipeline (with a novel post-processing algorithm) that obtains very low error on QoIs while guaranteeing the error on primary data. The pipeline was demonstrated using an XGC-based simulation of ITER [31] on two different platforms. Our approach achieves a compression level of 295 for the entire simulation, spanning 125 time steps. The error achieved on PD and QoI was less than 10^{-3} and 10^{-8}, respectively, for each time step.

In terms of performance, the post-processing technique improves the performance of the post-processing step by a factor of 8, while the MGARD optimizations improve the performance of MGARD operations by a factor of 4.6. Additionally, a software approach was developed to enable the concurrent execution of compression and simulation on separate nodes, with communication between them. The performance of online compression and post-processing was evaluated on the Summit and Perlmutter systems by varying the number of engaged nodes. Multiple distributed processes were utilized, leveraging a single GPU and multiple CPU cores per process for parallel compression and post-processing. The training of the autoencoder (AE) and the MGARD residual compression were performed on the GPU, while the post-processing phase primarily utilized CPU resources.

Overall, the computational overhead of compression was less than one percent, a significant improvement over our earlier work [4] with an overhead of three percent. Other approaches, such as SZ and ZFP, can be replaced by MGARD. However, the latter had small PD and QoI errors as compared to the other two.

In summary, we have shown that due to novel mathematical concepts developed in this paper, our compression pipeline is scalable and has low resource requirements, which makes it highly suitable for on-the-fly compression.

Acknowledgements

This research was partially supported by DOE DE-SC0022265 and DOE DE-SC0021320 RAPIDS2.

References

[1] 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*, 19(5-6):65–76, 2018.

[2] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky. Multilevel techniques for compression and reduction of scientific data—the multivariate case. *SIAM Journal on Sci. Computing*, 41(2):A1278–A1303, 2019.

[3] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky. Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities. *SIAM Journal on Scientific Computing*, 41(4):A2146–A2171, 2019.

[4] T. Banerjee, J. Choi, J. Lee, Q. Gong, J. Chen, S. Klasky, A. Rangarajan, and S. Ranka. Scalable hybrid learning techniques for scientific data compression. *arXiv preprint arXiv:2212.10733*, 2022.

[5] 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 *International Conference on High Performance Computing, Data, and Analytics (HiPC)*, 2022.

[6] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 2014.

[7] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[8] Y. Censor and A. Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34:321–353, 1981.

[9] C. S. Chang, S. Ku, R. Hager, R. M. Churchill, J. Hughes, F. Köchl, A. Loarte, V. Parail, and R. A. Pitts. Constructing a new predictive scaling formula for ITER's divertor heat-load width informed by a simulation-anchored machine learning. *Physics of Plasmas*, 28(2):022501, Feb 2021.

[10] C.-S. Chang, S.-H. Ku, P. H. Diamond, M. Adams, R. Barreto, Y. Chen, J. Cummings, E. D'Azevedo, G. Dif-Pradalier, S. Ethier, et al. Whole-volume integrated gyrokinetic simulation of plasma turbulence in realistic diverted-tokamak geometry. In *Journal of Physics: Conference Series*, volume 180, 1, page 012057. IOP Publishing, 2009.

[11] J. Chen, L. Wan, X. Liang, B. Whitney, Q. Liu, D. Pugmire, N. Thompson, J. Y. Choi, M. Wolf, T. Munson, et al. Accelerating multigrid-based hierarchical scientific data refactoring on GPUs. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 859–868. IEEE, 2021.

[12] J. Y. Choi, C.-S. Chang, J. Dominski, S. Klasky, G. Merlo, E. Suchyta, M. Ainsworth, B. Allen, F. Cappello, M. Churchill, et al. Coupling exascale multiphysics applications: Methods and lessons learned. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 442–452. IEEE, 2018.

[13] Y. Collet and M. S. Kucherawy. Zstandard Compression and the 'application/zstd' Media Type. *RFC*, 8878:1–45, 2021.

[14] J. Dennis and J. Moré. Quasi-Newton Methods, Motivation and Theory. *SIAM Review*, 19(1):46–89, 1977.

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

[16] I. Foster, M. Ainsworth, J. Bessac, F. Cappello, J. Choi, S. Di, Z. Di, A. M. Gok, H. Guo, K. A. Huck, et al. On-

line data analysis and reduction: An important co-design motif for extreme-scale computers. *The International Journal of High Performance Computing Applications*, 35(6):617–635, 2021.

[17] W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck, et al. ADIOS 2: The adaptable input output system. a framework for high-performance data management. *SoftwareX*, 12:100561, 2020.

[18] Q. Gong, X. Liang, B. Whitney, J. Y. Choi, J. Chen, L. Wan, S. Ethier, S.-H. Ku, R. M. Churchill, C.-S. Chang, et al. Maintaining trust in reduction: Preserving the accuracy of quantities of interest for lossy compression. In *Smoky Mountains Computational Sciences and Engineering Conference*, pages 22–39. Springer, 2021.

[19] J. Hines. Stepping up to Summit. *Computing in Science & Engineering*, 20:78–82, 03 2018.

[20] H. Jégou, M. Douze, and C. Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[21] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] S.-H. Ku, C.-S. Chang, and P. H. Diamond. Full-f gyrokinetic particle simulation of centrally heated global ITG turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry. *Nuclear Fusion*, 49(11):115021, 2009.

[23] J. Lee, Q. Gong, J. Y. Choi, T. Banerjee, S. Klasky, S. Ranka, and A. Rangarajan. Error-bounded learned scientific data compression with preservation of derived quantities. *Applied Sciences*, 12:6718, 07 2022.

[24] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, 2018.

[25] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. Pugmire, M. Wolf, N. Podhorszki, et al. MGARD+: Optimizing multilevel methods for error-bounded scientific data reduction. *IEEE Transactions on Computers*, 71(7):1522–1536, 2021.

[26] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[27] P. Lindstrom. Error distributions of lossy floating-point compressors. Technical report, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), October 2017.

[28] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE TVCG*, 12(5):1245–1250, 2006.

[29] J. Logan, M. Ainsworth, C. Atkins, J. Chen, J. Y. Choi, J. Gu, J. M. Kress, G. Eisenhauer, B. Geveci, W. Godoy, et al. Extending the publish/subscribe abstraction for high-performance I/O and data management at extreme scale. *Bulletin of the IEEE Technical Committee on Data Engineering*, 43(1), 2020.

[30] S. Ostadzadeh, B. M. Elahi, Z. Tabrizi, M. A. Moulavi, and K. Bertels. A two-phase practical parallel algorithm for construction of Huffman codes. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 284–291. CSREA Press, 2007.

[31] P.-H. Rebut. ITER: the first experimental fusion reactor. *Fusion Engineering and Design*, 30(1):85–118, 1995.

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

[33] J. Tian, C. Rivera, S. Di, J. Chen, X. Liang, D. Tao, and F. Cappello. Revisiting Huffman coding: Toward extreme performance on modern GPU architectures. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 881–891. IEEE, 2021.

[34] 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 ICDE*, pages 1643–1654. IEEE, 2021.