Multifacets of lossy compression for scientific data in the Joint-Laboratory of Extreme Scale Computing

Franck Cappello, Sheng Di, Robert Underwood, Dingwen Tao, Jon Calhoun, Yoshii Kazutomo, Kento Sato, Amarjit Singh, Luc Giraud, Emmanuel Agullo, et al.

Journal Pre-proof

Multifacets of lossy compression for scientific data in the Joint-Laboratory of Extreme Scale Computing

Franck Cappello, Sheng Di, Robert Underwood, Dingwen Tao, Jon Calhoun, Yoshii Kazutomo, Kento Sato, Amarjit Singh, Luc Giraud, Emmanuel Agullo, Xavier Yepes, Mario Acosta, Sian Jin, Jiannan Tian, Frédéric Vivien, Boyuan Zhang, Kentaro Sano, Tomohiro Ueno, Thomas Grützmacher, Hartwig Anzt

PII: S0167-739X(24)00255-3
DOI: https://doi.org/10.1016/j.future.2024.05.022
Reference: FUTURE 7323

To appear in: *Future Generation Computer Systems*

Received date : 10 January 2024
Revised date : 9 May 2024
Accepted date : 14 May 2024

Please cite this article as: F. Cappello, S. Di, R. Underwood et al., Multifacets of lossy compression for scientific data in the Joint-Laboratory of Extreme Scale Computing, *Future Generation Computer Systems* (2024), doi: https://doi.org/10.1016/j.future.2024.05.022.

REVISED Manuscript

Multifacets of lossy compression for scientific data in the Joint-Laboratory of Extreme Scale Computing

Franck Cappello, Mario Acosta, Emmanuel Agullo, Hartwig Anzt, Sheng Di, Luc Giraud, Thomas Grützmacher, Sian Jin, Kento Sato, Amarjit Singh, Dingwen Tao, Jiannan Tian, Tomohiro Ueno, Robert Underwood, Frédéric Vivien, Xavier Yepes-Arbós, Kazutomo Yoshii, Boyuan Zhang

Abstract

The Joint Laboratory on Extreme-Scale Computing (JLESC) was initiated at the same time lossy compression for scientific data became an important topic for the scientific communities. The teams involved in the JLESC played and are still playing an important role in developing the research, techniques, methods, and technologies making lossy compression for scientific data a key tool for scientists and engineers. In this paper, we present the evolution of lossy compression for scientific data from 2015, describing the situation before the JLESC started, the evolution of this discipline in the past 8 years (until 2023) through the prism of the JLESC collaborations on this topic and some of the remaining open research questions.

1. Introduction

In this paper, we will describe the compression work mainly developed under the collaborations of JLESC organizations, including Argonne National Laboratory, Barcelona Supercomputing Center (BSC), National Center for Supercomputing Applications (NCSA), Rikagaku Kenkyusho Institute (RIKEN), National Institute for Research in Digital Science and Technology (INRIA), Juelich Supercomputing Center, and the University of Tennesse. The paper presents a collection of compression research in many different scientific domains and several use cases. While more details about the presented works can be found in the corresponding publications, we believe that the overview presented in this paper will introduce readers to the breadth of lossy compression applicability to scientific data. It should be noted that the paper does not cover all scientific domains and possible use cases and, in fact, as the past height years show, the application of lossy compression for scientific data is rapidly expanding to more scientific domains and more use cases.

1.1. Before 2015

Before 2015, scientific data reduction was initially limited to a few specific use cases where generated data was exceeding—sometimes by several orders of magnitude—the communication, storage, and analysis capabilities of that time. The first use case was experiments producing critical data for scientific discoveries. Examples of such experiments are the detectors at the Large Hadron Collider (LHC) [52] and the NASA Mars rovers [87]. In the case of the LHC, detectors are equipped with real-time analysis systems to select the most interesting proton-proton (pp) collisions and reduce the data by up to 1/100000 [52]. For the NASA Mars rovers, the communication bandwidth is constrained by the deep-space communications channel between Mars and Earth [87]: for the Curiosity rover, the data rate direct-to-Earth was 32 kb/s, and the data rate through the Odyssey orbiter was 256 kb/s. To reduce image data, NASA Mars rovers used different image compressors: ICER [110] and JPEG [20]. The second main use case was a visualization of scientific data [58, 96, 96]. The need for lossy compression for scientific visualization was identified in the 1980s [2]. For many other scientific experiments and simulations, data reduction was unnecessary: data resolution and generation rate of instrument detectors and numerical simulation were compatible with contemporary communication and storage technologies. The interest in a broader use of scientific data reduction and compression increased circa 2000, with compression schemes such as ISABELA [90], ISOBAR [130], FPC [24], and fpzip [104].

1.2. From 2015 to 2023

The first successful compressor for scientific data was ZFP from Lawrence Livermore National Laboratory. The first ZFP paper was published in December 2014 [103]. A few months later and independently, the SZ project started at Argonne National Laboratory, and the first version of SZ was published in June 2016 [36]. The rapid success of these two compressors, combined with a realization from the broader scientific community of the necessity to reduce data effectively, inspired many other teams to develop scientific data compression schemes. One difficulty for the early adopters of lossy compressors was to understand the performance of the different compressors from the published papers. Many publications were using different datasets and different metrics to showcase the better performance of their compression schemes. This situation created a problem concerning the scientific methodology: how to fairly compare different compression schemes if comparisons do not use community-established common datasets and metrics. The

SDRBench[1] [162] repository was created to address this problem. SDRBench provides 14 different scientific datasets from different scientific disciplines that compression scheme developers can use to evaluate the performance of their design and compare it with other compressors. In parallel with this effort, the Z-checker [140, 158] software was created to provide users and developers of lossy compressors with a tool offering many (more than 30) different metrics to evaluate the impact of compression schemes on scientific data.

From 2015 to 2023, the need for scientific data reduction and compression increased rapidly, with many different applications and use-cases [26]. This expansion of the compression usage for scientific data is still increasing, with new use cases in artificial intelligence (AI) (e.g., federated learning communication compression, compression of scientific data for training, compression of AI models weights and activations). Several important evolutions of existing compressors or new compression schemes have appeared, increasing sometimes dramatically the performance of previous compressors. For example, MGARD was proposed for lossy compression of scientific data while respecting quantities of interest accuracy [54]. TTHRESH, which relies on high-order singular value decomposition for decorrelation, was designed to reach very high compression ratios on datasets with three or more dimensions [18]. SZ2 was refactored to become a modular and highly customizable lossy compression framework [101]. This redesign is instrumental for the implementation of specialized compressors such as ROIBIN-SZ [150] and MDZ [163]. Bit Grooming [159], Digit Rounding [34], and SPERR [95] were optimized for climate data lossy compression. From 2000, several groups focused on lossy compression using AI models [117], perhaps with less success than initially expected, which motivates further research in this area.

In parallel with a better understanding of compression schemes and their applications to scientific data, users started to express more complex data quality preservation criteria. Beyond preserving quantities of interest, researchers in cosmology stressed the need to preserve the accuracy of analysis results. For example, they required compressors to generate compressed data leading to a maximum deviation on the power spectrum analysis of 1%. They had similar requirements concerning the halo mass distribution analysis. For crystallography, the requirement was to keep intact (no lossy compression) regions of interest surrounding bright peaks and compress the remaining data significantly. For molecular dynamics, users demanded the precise preservation of the radial distribution function. In many cases, these difficult requirements could not be directly translated through mathematical analysis into controls of the quantization (approximation) stage of the lossy compressors. In some cases, such formal analysis of the compression errors could be used to design lossy compressor error controls preserving important data characteristics. Examples are the preservation of critical points in vector fields [99] and the preservation of linear and nonlinear data characteristics [54, 75, 154].

[1] https://sdrbench.github.io

The widening interest in and adoption of lossy compression schemes by the scientific community have also encouraged researchers to investigate the design, implementation, and effects of "ultra-fast lossy compressors" for compressing communications in parallel simulations and even data between the processors (CPUs, GPUs, accelerators) and the memory (internal or external) [42, 41].

The following sections present examples of applications and use cases of lossy compression for scientific data, specifically developed by collaborators within the JLESC.

2. Compression for Climate Simulations

In this section, we describe the lossy compression research related to climate simulations that was conducted by researchers from Argonne National Laboratory (ANL) and Barcelona Supercomputing Center (BSC).

2.1. Research Background

The execution scales of climate simulations such as the Community Earth System Model (CESM) [85, 72] or EC-Earth [38] keep growing because of ever-increasing research problems to study. More and more model output data is being produced, and how to efficiently store and transfer it has become a critical issue for researchers. For example, when running large climate ensembles of high-fidelity 1 km $\times$ 1 km simulations where each instance simulates 15 years, 260 TB of data could be generated for estimating only one ensemble member per simulated day [37]. For the climate simulations with the Coupled Model Comparison Project Phase 6 [112], each model may produce nearly 10 PB of data. More examples of large volumes of data that are handled by climate research computing centers can be found in [15, 17, 71, 89, 159].

2.2. Research Problem Formulation

In this subsection we introduce a pair of Earth system models (ESMs) commonly used in the climate modeling community, the research objective in using lossy compression for these models, and evaluation methods in the context of climate data.

The climate research community has two well-known models. One of them is EC-Earth, developed by a European consortium of national meteorological services and research institutes. Its latest version currently being developed, EC-Earth4, couples various components, including OpenIFS 43R3 as the atmospheric component, NEMO 4.0.1 as the ocean component, and other components coupled using OASIS3-MCT, to understand and predict climate variability and climate change.

The other well-known ESM is the Community Earth System Model (CESM) [85], which is a fully coupled global climate model developed in collaboration with many partners in the research community. CESM provides effective simulations of the past, present, and future climate states of the Earth using hundreds of different analysis fields, including potential temperature, salinity, in situ density, sea surface height, solar shortwave heat flux, abiotic surface pH, and vertical submeso velocity.

The research objective is to explore or develop an efficient error-controlled lossy compressor, which is expected to get high compression ratios on climate datasets while respecting the fidelity very well in climate research. There are three ways to assess the quality of the reconstructed data in general, all of which have been widely used in the community.

- *Statistical Metric*. Quite a few statistical metrics have been used in prior studies [17, 14, 13, 121, 64, 148], including structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR), and KS test.
- *Visualization*. Many existing studies [14, 148] conducted by both compression developers and climate scientists assess the quality of the lossy reconstructed data by visualization of the lossy reconstructed data versus original uncompressed data.
- *Feature Analysis*. Some works [56] have focused on the preservation of specific features of climate data (such as during the lossy compression in particular).

2.3. Related State of the Art

In this subsection, we comprehensively discuss the related works, which mainly include the different lossy compression methods developed and evaluated for climate simulation datasets.

Multiple works have evaluated the effectiveness or impact of lossy compression methods on climate simulation datasets. In 2014, Baker et al. [17] published a paper in which they evaluated the compression ratios for multiple lossless compressors (such as FPC [24]) and lossy compressors (such as FPZIP [105], ISABELA [90], and JPEG2000 [141]), as well as the impact of lossy compressors on climate data analysis. Many statistical analysis metrics were used in that investigation, including compression ratio, PSNR, normalized root mean squared error, and normalized maximum pointwise error. However, much more efficient error-bounded lossy compressors were not yet developed in 2014, so that work missed many important compressors such as SZ [36, 139, 98, 161, 101], ZFP [103], and Digit Rounding [34]. Thereafter, a few papers were published that investigated the effectiveness of error-bounded lossy compressors on climate data, while focusing mainly on two lossy compressors—SZ and ZFP. For example, in [121, 64, 14, 13], the climate researchers provided a series of analyses for lossily compressed climate model data based on some lossy compressors including SZ1.4.13 [139], ZFP0.5.3 [103], and FPZIP [105]. However, these compressors are now relatively outdated because more effective lossy compression algorithms were devised thereafter. For example, the latest version of SZ has been upgraded to 3.1, which adopts a different data prediction method in the SZ compression pipeline, so that the compression quality and results are substantially different [161]. Recently, Underwood et al. [148] performed a comprehensive investigation of the compression quality on climate datasets for 11 state-of-the-art error-bounded lossy compressors including MGARDx [100], SZ3 [101], ZFP 0.5.5 [103], and TTHRESH [18]. This study included many important metrics, such as the KS test and SSIM [16]. The authors concluded that either SZ3 or ZFP has the best compression ratios/qualities in most cases.

In addition to the specific analysis of the effectiveness of lossy compressors for climate datasets, climate data have also been widely used as classic scientific datasets for the design and evaluation of new error-bounded lossy compressors. CESM datasets, for example, have often been used to evaluate SZ-series compressors (from the early version 1.4 [139] to 2.1 [98], 3.1 [161], QoZ [106], and FAZ [107], as well as GPU data compressors [144, 143]) during the development of SZ or performance optimization.

2.4. XIOS and OpenIFS

The use case used by the Barcelona Supercomputing Center to test SZ involved XIOS and OpenIFS. OpenIFS is an atmospheric general circulation model developed and maintained by the European Centre for Medium-Range Weather Forecasts (ECMWF). It is derived from the Integrated Forecasting System (IFS) [19, 40], which is the operational global meteorological forecasting model and data assimilation system of ECMWF. OpenIFS is used as the atmospheric component of the new EC-Earth4 model.

One of the main bottlenecks of OpenIFS was the old sequential I/O scheme used, where all 2D and 3D fields were gathered by the master Message Passing Interface (MPI) process, and the data was then sequentially written onto the storage system. The approach did not scale at high resolutions. To overcome this issue, XIOS was integrated into OpenIFS [155]. The XML Input/Output Server (XIOS) [84, 66, 109] is an asynchronous MPI parallel I/O server used by Earth system models to avoid I/O contention. It focuses on very high scalability with support for high-resolution output. XIOS is developed by the Institute Pierre Simon Laplace. Figure 1 shows the integration between OpenIFS and XIOS. The OpenIFS processes execute the XIOS clients that asynchronously send data to the XIOS servers. These servers are run on independent nodes and are in charge of writing the data into the storage system.

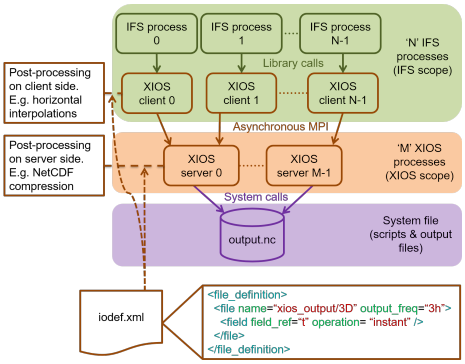


Figure 1: Diagram of the OpenIFS-XIOS integration

Since XIOS uses netCDF4 as the output data format, it is possible to enable lossless compression via HDF5. It uses gzip,

but this default compression filter cannot offer high compression ratios and can take a considerable amount of time. Keeping low compression times is important to avoid increasing the execution time of the model, which would considerably impact the computational efficiency. Also crucial is the high compression of the data without affecting the accuracy needed to perform the scientific studies. This capability will become even more important in the upcoming years when new exascale machines will allow us to run the models at an unprecedented level of resolution that will output a huge amount of data. To explore lossy compression tools, therefore, we integrated SZ into XIOS and evaluated the results.

2.5. XIOS-SZ Integration

The first step was to study how to use SZ from XIOS to minimize the code changes. SZ is registered as a third-party filter of HDF5 [50], so it can be easily enabled from the netCDF4 API.

We developed the HDF5 filter for the SZ compressor (namely, *H5Z SZ filter*) based on the HDF5 filtering mechanism [48]. The design architecture is illustrated in Figure 2. As shown in the figure, the H5Z SZ filter comprises three key components—H5Z_SZ_Init, H5Z_sz_set_local, and H5Z_filter_sz. They are in charge of the initialization (such as memory allocation), HDF5 format analysis and SZ parameter configuration, and execution of the SZ compressor, respectively. The datasets stored in the original HDF5 file would be extracted one by one and passed to the SZ filter for compression processing. After that, another HDF5 formatted file with compressed data stored instead is generated, while the original hierarchical structure of the HDF5 file is not changed at all.

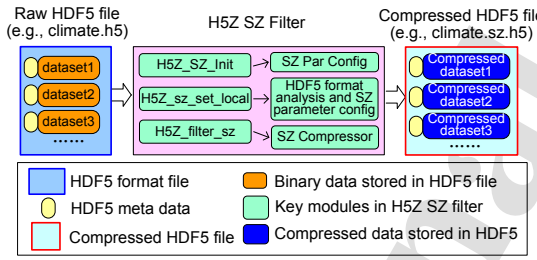


Figure 2: Design architecture of the H5Z SZ filter

Figure 3 shows the usage of the H5Z SZ filter in XIOS: when XIOS creates a new netCDF4 file, the netCDF4 library internally uses HDF5 to structure the different datasets. This feature allows us to use the SZ compressor as a third-party library. In addition, the H5Z SZ filter allows us to define independent compression parameters per field. This is an important feature because in climate modeling the needed accuracy between fields can widely vary. For example, the precision bits for total precipitation are different concerning vorticityas compared to that of vorticity.

The XIOS code was also refactored to make use of the H5Z SZ filter and support independent compression parameters. In

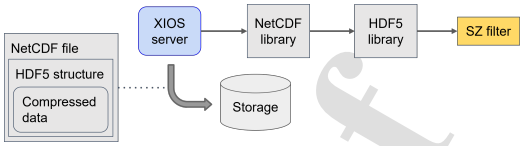


Figure 3: Usage of the H5Z SZ filter in XIOS through the netCDF4 library

XIOS, the output configuration is set up through XML files, which now are also used to specify different compression parameters, as shown in the following listing.

Listing 1: Example of an XML definition for independent compression parameters

```
<file id="x" compression_type="lossy"
    error_bound_mode="abs" error_bound="0.001">
  <field field_ref="t" error_bound_mode="rel"
      error_bound="0.1" />
  <field field_ref="q" error_bound="0.00001" />
  <field field_ref="cc" compression_type="lossless"
      compression_level="4" />
</file>
```

On the other hand, XIOS was also adapted to perform compression in parallel, for both gzip and SZ. This feature is particularly relevant because compression in parallel mode was not available in HDF5 until version 1.10.2 when support was added. It is an interesting feature because for large netCDF files the increase in time due to serialization is high. This can be minimized by enabling compression because it is therefore cheaper in time to write smaller netCDF files. Figure 14 illustrates the two types of writing and compressing files in XIOS through HDF5. XIOS has two different writing modes as illustrated in Figure 4: The pool of servers (a) can be independently used to write the data (multiple file mode), where each server writes an independent netCDF file with a subset of the data; or they (b) can jointly and synchronously write the data (one file mode), where all servers write a single netCDF file containing all the data. XIOS originally could only use compression with the multiple file mode because it is an inherited feature of HDF5. However, from version 1.10.2, HDF5 started to support parallel compression, although it might not be well-tuned yet. Taking advantage of this, XIOS was also adapted to enable compression in the one file mode, for both gzip and SZ. This feature is important to try to mitigate one of the main performance issues of XIOS in using the one-file mode. This mode requires synchronization among all the servers to correctly write into the same file. As a consequence, this serialization implies a degradation that gets worse for large volumes of data. If data is compressed before writing in this mode, the size reduction might considerably help in mitigating the performance degradation.

2.6. Performance Evaluation

The SZ compressor in XIOS was evaluated on MareNostrum4, the supercomputer of the BSC. Its computing nodes consist of two sockets of an Intel Xeon Platinum 8160 CPU with 24

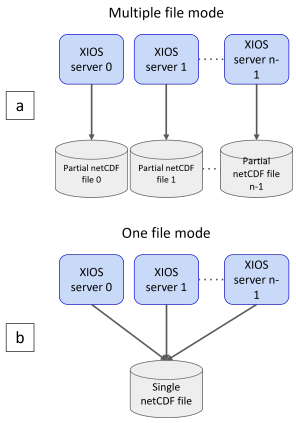


Figure 4: Diagram (a): partial compression where each I/O server compresses its netCDF file; diagram (b): the new compression in parallel where all I/O servers write into a single compressed netCDF fileDiagram (a): multiple file mode where each I/O server writes its netCDF file; diagram (b): one file mode where all I/O servers write into a single netCDF file

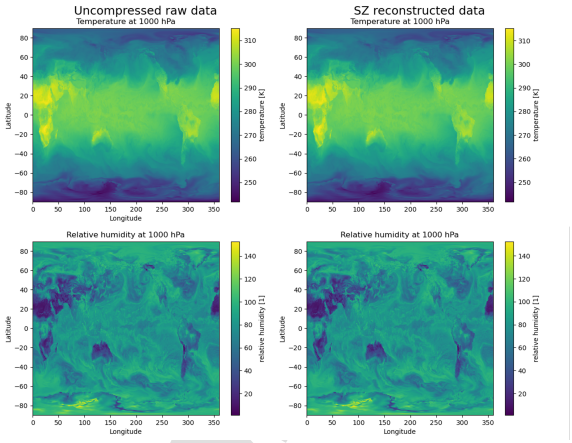


Figure 5: Comparison between Uncompressed raw and SZ reconstructed data (error bound = 0.001) using temperature and relative humidity fields at 1000 hPa

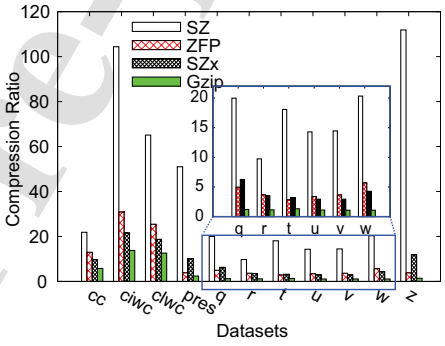


Figure 6: Comparison of compression ratios among different lossy compressors with the same relative error bound of 0.001

cores each for a total of 48 cores per node and 96 GB of main memory distributed with 12x 8 GB 2667 MHz DIMM. The tests use 80 MPI processes for OpenIFS, six OpenMP threads in each MPI task, and two XIOS I/O servers in an exclusive node. For the evaluation, the tests included a simulation of a ten-day forecast running a Tco511L91 configuration, which corresponds to a horizontal resolution of 23 km and with a time step of 900 seconds. We extracted 10+ datasets (climate fields) from the climate data files (.nc) and evaluated the compression ratios of different error-bounded lossy compressors (including SZ, SZx, and ZFP) with the same error bounds as well as Gzip lossless compressor. Since the datasets are all in unstructured format, we can only treat them as 1D arrays.

We first assess the quality of the reconstructed data generated by the error-bounded lossy compression against the original uncompressed raw data to ensure the appropriateness of the error-bound setting we are using. Figure 5 demonstrates the visualization of the two 3D fields, temperature, and relative humidity, at the pressure level of 1000 hPa, based on the raw data and lossy compressed data (using SZ with relative error bound of 0.001). The figure demonstrates a fairly high visual quality of the decompressed data under the relative error bound of 0.001. Thus, we use the same setting in the following test for other lossy compressors on the climate datasets.

We evaluate the compression ratios for different lossy compressors on the climate datasets. As shown in Figure 6, SZ exhibits the highest compression ratio from among all compressors in our evaluation. In absolute terms, SZ's compression ratio is about 1.5-20× and 2-10× as high as that of the lossy compressor ZFP and SZx, respectively.

Figure 7 further shows different tests comparing gzip against SZ in different scenarios: no compression and compressing 19 3D fields combined with one and multiple file modes, which corresponds to the two modes illustrated in Figure 4. For these tests a compression level of 5 was used for gzip and a relative error bound of 0.001 for SZ. Figure 7 shows different tests comparing no compression and SZ compression of 19 3D fields combined with one and multiple file modes of XIOS, which corresponds to the two modes illustrated before in Figure 4. For these tests, a relative error bound of 0.001 was used for SZ.

In Figure 7 we can observe that the compression ratio of SZ is much higher than that of gzip and that SZ takes less time as well: SZ can offer a compression ratio of 20.5 whereas gzip's ratio is only 1.94. The overhead of enabling the SZ compression for the multiple file mode is negligible, but in gzip there is an extra cost of 142%. This becomes even worse with the one-file mode because the serialization due to the needed synchronization of multiple I/O servers is very high when writing large volumes of data. This hugely impacts gzip but not SZ at the same level: while SZ has an overhead of 71.1%, gzip needs up to 520.3% more time.In Figure 7 we can observe that the compression ratio of SZ is high (around 20.5) for both one and multiple file modes of XIOS. However, whereas the execution

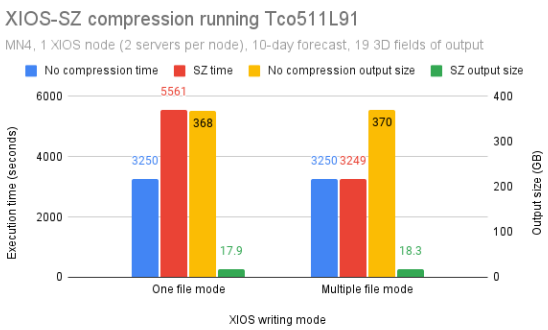


Figure 7: Comparison of the execution time and the output size between gzip and SZ using both normal and parallel I/OExecution time and output size using no compression and the SZ compressor in XIOS for both one and multiple file modes

time does not increase for the multiple file mode, there is considerable performance degradation for the one file mode. This is a preliminary study, so we plan to further analyze the source of this degradation in the future. It might be related to the new parallel compression feature added in HDF5. It might also be caused by the performance issue of the one file mode for large netCDF files described in Section 2.5. However, it is possible to improve this performance degradation by adding more XIOS servers to make use of more computational resources, and therefore, speeding up the processing of events, compression, and netCDF encoding. For instance, instead of using 2 XIOS servers as illustrated in Figure 7, with 6 XIOS servers the execution time is reduced from 5561 to 4696 seconds.

It is also important to note that the compression ratio between the one and multiple file modes is slightly different. This is related to the metadata of the netCDF files. In the multiple file mode metadata is replicated among the different partial netCDF files, whereas in the one file mode metadata is outputted once, in the single netCDF file generated.

2.7. Conclusions and Future Work

We have presented an overview of the integration of the SZ compressor in XIOS, which involved developing the H5Z SZ filter as a third-party filter of HDF5 and refactoring the XIOS source code to make use of SZ as well as defining independent compression parameters per field. This last point is crucial to produce enough accurate data to not invalidate scientific results.

The performance evaluation of SZ is positive compared with the default lossless compression of XIOS offered by gzip. The SZ compressor is faster, and it achieves much higher compression ratios, which are even more outstanding when using the one-file mode of XIOS.The performance evaluation of SZ is positive when using the multiple-file mode of XIOS because the performance remains the same and it achieves a high compression ratio. In addition, the one file mode can be improved when adding extra XIOS servers to make use of more computational resources. However, the performance numbers presented in this paper might vary because we used the same compression parameters for all the fields, so in an experiment using scientifically validated compression parameters these can be different.

An important need in the future is to determine with climate scientists the different acceptable error ranges per field to set the compression parameters accordingly. In addition, further work is needed on the parallel compression offered by the one-file mode of XIOS. In that sense, of interest is the testing of the latest versions of HDF5 to see the impact of the performance improvements introduced in the experimental version 1.13.1.

3. Compression for Instrument Data

In this subsection, we delve into our collaborative research efforts between RIKEN and Argonne National Laboratory on data compression techniques for large experimental scientific instruments such as particle colliders, distributed radio telescopes, and synchrotron radiation facilities. These instruments are pivotal in deepening our understanding of the universe, leading to significant advancements in fields such as semiconductors, medicine, and materials, to highlight just a few areas of impact. To facilitate scientific discoveries, capturing more intricate details both spatially and temporally is essential, which consequently leads to an exponential surge in the volume of data generated. Figure 8 depicts a typical data pipeline configuration of scientific instruments and consists of two distinctive edge computing stages: upstream and downstream. With the ever-increasing volume of data generated by those instruments, transmitting raw data downstream for analysis becomes exceedingly challenging, cost-ineffective, or impossible if the data rate surpasses the capabilities of the network hardware that bridges between stages. Our goal is to understand the nature of data at each stage and investigate optimal compression techniques.

Synchrotron radiation facilities play a crucial role in studying the structures of various materials, from physical to biological domains. These facilities use intense X-rays to capture structural changes over time, providing insights into phenomena such as protein function. As these facilities enhance their capabilities with improved X-ray sources and advanced X-ray imaging detectors, they will produce even more data. For instance, the SPring-8 facility will upgrade its beamlines with the upcoming CITIUS generation detector. In 2025, SPring-8 beamlines are expected to generate a massive 1.3 exabytes of raw data annually. Likewise, with the completion of the upgrades at the Advanced Photon Source at Argonne National Laboratory and the Linac Coherent Light Source at SLAC National Accelerator Center in the United States, these facilities are expected to generate 1 TB/s.

3.1. Research Problem Formulation

First, the necessity for data compression or reduction inside or near the data source has become paramount in order to transfer data efficiently into downstream edge computing. Compressing data directly within the detector ASICs is one of the potential solutions. However, the challenge lies in developing efficient hardware compressor algorithms and devising a design

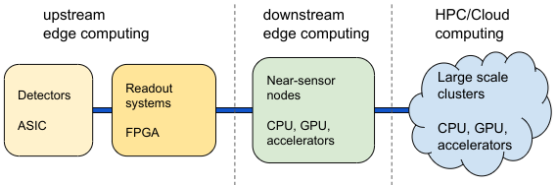


Figure 8: A data pipeline configuration of scientific instruments

strategy for hardware compressor algorithms that not only meet performance and resource requirements but also minimize development costs.

Hundreds to thousands of terabytes of instrument data are processed, archived, visualized, and analyzed each year at light sources[150, 93]. At this volume, data instrument data presents significant data compression of instrument data presents a significant challenge due to the complexity and volume of data involvedoperational challenges to move and store. Compression offers a potential solution, but lossless compression is insufficiently effective to use as the primary means of compression which leads to the consideration of lossy compression. However, with lossy compression, it becomes critical to preserve the scientific integrity of the images which itself is complicated by the complexity and acquisition rate of the data. This data consists of observations of complex materials at short intervals with significant noise. Compression methods which are capable of separating the true data from the noise is critical in order to achieve high compression ratios while preserving the scientific content of the images while meeting operational constraints. We provide details on how each compression approach tackles this differently in Section 3.3 Without preserving the scientific integrity of the images, lossy compression techniques will not be adopted.

Another significant issue is the variability of instrument data, which differ in their structure or data type due to the different kinds of experiments performed at light sources. Serial crystallography, tomography, ptychography, X-ray photon correlation spectroscopy, and coherent surface scattering imaging represent just a fraction of the kinds of methodologies used at light sources, and each produces vastly different structures of data. Compounding the challenges raised by the various experimental methodologies are the different types of detectors with different features including gain-switching sensitivity, various capture rates, and different output precision, leading to myriad data structures that need to be compressed. Therefore, one-type-fits-all compression solutions are not appropriate.

Additionally, the continuous generation of real-time instrument data introduces the need for real-time compression and decompression processes, further complicating the task. Detectors have limited storage capacity as they capture images continuously. If the data is not read off the detector quickly, it will be overwritten by the next frame, resulting in uncontrolled data loss that will inhibit the capture of scientific phenomena. Thus, compression pipelines located at the edge need to operate in real-time, placing constraints on the bandwidth of the compressor.

In order to address these challenges effectively, domain-specific knowledge and innovative adaptations of compression techniques are required to ensure that instrument data remains accessible and valuable for scientific and industrial purposes.

Why It Is Important. Data from light sources informs our understanding of the world around us. Domains as far spread as agriculture, biology, neuroscience, electrochemistry, materials science, and microelectronics leverage this data so that we can better understand the structure and properties and can design better solutions to problems all around us. Improvements in detector technology give us greater resolution allowing us to study progressively smaller phenomena, a larger field of view allowing us to study larger samples, and a higher rate allowing us to study increasingly ephemeral phenomena. However, these capabilities further increase the volume of data required.

To understand why compression for instruments is important, one must understand the consequences of either not using compression at all or using only lossless compression. Storing nearly 1 TB/s continuously would exhaust all of the available storage at Argonne in less than 3 days and all of the storage at Riken in less than 2 days, rendering each system at these sites ineffective for most tasks since they require at least some storage. Lossless compression is of limited effectiveness because of the high entropy of light source data, giving a compression ratio of less than 2, which grants only a brief reprieve from the deluge of data by extending our available capacity to just under a week. However, lossy compression systems such as ROIBIN-SZ can achieve much higher compression ratios—up to 196x—extending the time available to process data from a week to nearly a year by allowing collected data to be stored in a compressed form prior to processing drastically decreasing the rate at which storage capacity is consumed, allowing time to sort through the data that is collected and to transfer only important data to off-site/archival storage, leaving storage capacity for other usages of the machines and reducing costs [150]. Therefore, lossy compression is an essential part of any sustainable storage system for processing data from light sources.

For lossy compression to be a complete solution, however, we need to establish the integrity of the results, the capability to meet bandwidth requirements, and the suitability for a wide variety of experiments.

Potential Impact. Efficiently managing data in large-scale facilities such as synchrotron radiation centers is essential to keep storage costs in check and streamline data transfer operations. Advanced data compression techniques and predictive models can significantly reduce the volume of data needing storage and transmission. Beyond cost reductions from reduced hardware requirements, reducing the volume of data also facilitates real-time analysis by providing more margin to complete tasks within deadlines with available resources. These concrete improvements underscore the transformative impact of these methods on the way data is handled and leveraged in the pursuit of scientific knowledge.

3.2. Related State of the Art

In this segment, we deliver an overview of the previous state of the art that is highly relevant to our work in the context of image and video compression.

Recently, researchers have expressed increasing interest in using convolutional neural networks (CNNs) and recurrent neural networks for data compression. Some of these methods build on the established HEVC (H.265) standard and target specific modules, such as intraprediction [31], [46], [94], interprediction [33], [164], quantization [8], entropy encoding [122], and loop filtering [74]. These efforts have generally led to improvements over the HEVC codec, ranging from 0.5 percent to 5 percent in terms of compression efficiency.

Numerous studies have delved into lossy compression techniques exploring a range of video standards such as JPEG, MPEG-4 (H.264), and HEVC (H.265). Other endeavors have involved deep neural networks operating independently of the HEVC framework. For example, Chen et al. [28] introduced DeepCoder, which achieved results on a par with the x264 decoder, and pixel motion (PixelMotionCNN) extension and hybrid prediction [28], which demonstrated comparable performance to the H.264 codec [133]. Some scientists have investigated the utilization of long short-term memory (LSTM) networks to learn video representations, while others have combined CNN and LSTM [108] to predict future frames. Although these approaches are aimed at improving existing video compression methods, they predominantly belong to the realm of lossy compression. Notably, their bitrate savings over the x264 encoder have generally averaged below 10%. Lossy compression techniques such as SZ and ZFP primarily concentrated on enhancing the compression of data by adhering to an error margin. SZ [36], for example, represents an error-bound lossy compression approach tailored for actual data compression, ZFP [100] stands as a specialized library for high-throughput compression of floating-point arrays. while ZFP [103] is primarily a specialized library for high-throughput compression of floating-point arrays.

TEZip offers cutting-edge solutions that can transform the instrument data. TEZip [127] is addressing vast instrument datasets. PredNet is an example of a deep convolutional recurrent neural network used for this purpose. This innovative approach to temporal relationships within instrument image data offers robust support for integer, floating-point value prediction in datasets. This advancement represents a significant step in the state of the art for data compression, enabling more effective handling of complex, time-varying image data.

When considering compression for light sources specifically, a different line of compressors has been considered. Prior to ROIBIN-SZ and TEZip, there were two primary state-of-the-art approaches: general-purpose error-bounded lossy compressors such as SZ and non-hit rejection. Hadiean-Jazi et al. [62] considered the use of so-called non-hit rejection techniques. Non-hit rejection observes that some frames contain no detected regions of interest. This occurs in serial crystallography because while injecting microcrystals before the detector, it is possible that no crystals are in the field of view of the detector. When such a frame is detected, the storage of the frame can be rejected, reducing the storage and bandwidth needs of the compression. On datasets that we study, this can reduce the volume of data by between 3.22× and 5.81×.

Two decades earlier, in 1998, Ferrer et al. [47] studied compression schemes derived from wavelet transforms (the basis of the modern-day SPERR [95] compressor) and cosine transforms (used by JPEG-style compression algorithms and similar to what is done in ZFP). Since that work, compression designs now support error bounds on the error introduced by compression, which allows for greater confidence in the reconstructed data.

More recently, Leonarski et al. [93] studied the use of SZ. This work found that while large compression ratios are possible, they can be affected substantially by small local intensity fluctuations when using large error bounds, thus limiting the effective compression ratios that can be used in practice with general-purpose error-bounded lossy compressors.

3.3. Overview of Methods Developed in the JLESC

Researchers at RIKEN and Argonne have developed three distinct methods within the JLESC, sharing progress and findings to improve compression for instruments. StreamPressor [157] is a stream compressor hardware generator written in the Chisel hardware construction langauge [12] for evaluating various designs of streaming hardware compressor that has combinations of predefined hardware compressor primitives as well as user-defined primitives, which allows researchers to evaluate efficient hardware compressor algorithms for future computing architecture and detector systems. Riken's compressor TEZip[127] (time evolutionary zip) uses a deep convolutional recurrent neural network to exploit the similarity between consecutive images in instrument data in order to achieve extremely high compression ratios at high speeds separating signal (differences between frames) from noise (similarities between frames). Argonne's compressor ROIBIN-SZ[150] instead relies upon more traditional computing approaches but leverages domain-specific knowledge from serial crystallography to separate the noisy background from features of interest while achieving high compression ratios at extremely high speeds while preserving scientific quality. Together these approaches serve distinct roles in the data life cycle at light sources—ROIBIN-SZ is better suited for extremely high-rate applications early in the life cycle when the data is collected, and TEZip is better suited for extremely high compression applications for archival purposes later in the life cycle.

Streaming Hardware Compressor Generator:. Research and development of on-chip hardware compressors for detector application-specific integrated circuits (ASICs) is a mid to long-term investment, similar to any other custom chip development. Due to the strict timing and resource constraints in detectors, hardware implementation of general-purpose compression algorithms, such as LZ77, is not feasible. Hardware compressors need to process data without relying on a large buffer, ideally with all processing occurring in a data-flow manner, eliminating the need for any buffer. Additionally, the unique characteristics of instrument data, such as sparsity and unusual data precision, may not

fit well with general-purpose compressor algorithms. However, these specific attributes present an opportunity for developing highly effective and simple custom hardware compression algorithms. Previously (outside of the JLESC context), RIKEN, Argonne National Laboratory, and other institutes have proposed custom hardware compression algorithms for scientific instrument data and numerical computation and evaluated their characteristics [63, 134, 135, 147]. In the context of our JLESC collaborative project, we have recently initiated a new research and development activity on hardware compressor generator frameworks in 2023. This activity focuses on hardware compressor generator frameworks, aiming to support future scientific detector co-design efforts. We have introduced StreamPressor [156], a generator framework for designing, verifying, and estimating resources in streaming hardware compressor architectures. This framework is intended to assist compressor developers in exploring different compressor architectures with different compressor building blocks, evaluating their characteristics (latency, throughput, gate counts), and generating RTL code for integrating them into custom accelerator designs. Our motivation is to bridge the gap between software and hardware experts through this proposed framework as a co-design tool, leveraging open-source hardware design tools.

Lossy Compression for Instrument Data by ROIBIN-SZ:. In order to understand ROIBIN-SZ, it is helpful to understand the general serial crystallography analysis workflow. First, a peak finder is used to identify so-called Bragg spots/peaks that represent the positions of electrons during the imaging process of serial crystallography. Second, since the peak-finding process has some false negatives and false positives, indexing is used to estimate the unit cell parameters of the crystalline structure from the positions of the Bragg peaks and the background of the images. Third, the indexed unit cells are merged into a 3D diffraction volume. Fourth, phase retrieval is performed on the merged volume to reconstruct the electron density, which is then studied to understand the structure and proprieties of the material.

The background for serial crystallography data can be noisy and difficult to compress to acceptable levels without very large error bounds. However, without preserving the data around the Bragg peaks carefully, the large error bounds needed to produce acceptable compression levels can cause the indexing and merging processes to fail, leading to failures in the later stages of the process. Given the importance of the peaks to recovering the unit cell and ultimately the electron density, one might guess that only the peaks can be stored. However, without preserving at least some of the background, we may inadvertently destroy some of the false negative peaks, thus limiting our ability to index, merge, and perform phase retrieval. By using knowledge of peak location determined by the peak finder we can assign error tolerances to the regions of interest and to the background, however, we can enable higher levels of compression.

A typical error-bounded lossy compressor will face challenges in compressing the background data. The reason is that the background is very noisy and appears to be uncorrelated. Since error-bounded lossy compressors such as SZ and ZFP are

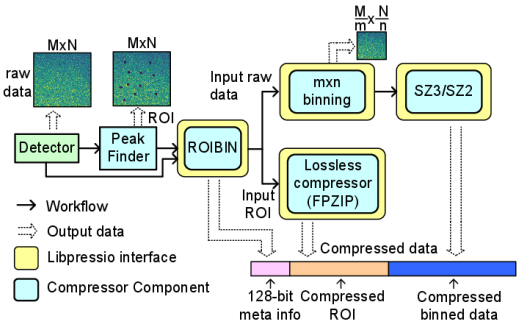


Figure 9: ROBIN-SZ overview diagram [150]

designed to exploit spatial correlations in the data, the lack of correlation results in poor performance. To increase the correlation, we apply a 2x2 binning process to the background. This smooths the data and improves its compressibility while reducing the volume of the data by a factor of 4 on its own. Because we store the regions of interest separately and losslessly prior to binning, we maintain high resolution signal around key features which are critical to the scientific analysis while using lower resolution around regions of non-interest containing noise. As we will demonstrate in section 3.5, this preserves the scientific integrity of the data with very high compression ratios.

Figure 9 presents an overview of ROIBIN-SZ, the meta-compressor we designed that combines lossless compression of key regions of interest with binning, followed by aggressive binning and then error-bounded lossy compression of the remaining background data [150]. The combination of these two approaches allows high levels of compression at the extremely high line rates of modern and future beamlines. Together the pipeline is implemented as a hierarchical parallel code using the LibPressio and SZ3 libraries, where each of the tasks can be parallelized, and potentially in the future accelerated on the GPU, while giving us maximal flexibility to experiment with different compression pipelines. As an added benefit of our design, we can use the OptZConfig framework to autotune the runtime performance of ROIBIN-SZ and set the number of execution threads to use for each level of parallelism.

Lossy Compression for Instrument Data by TEZip:. Time Evolutionary Zip (TEZip) is a specialized tool designed for handling the compression of instrument data. In this context, it is investigated in the domain of lossy compression. TEZip's lossy compression methodology encompasses several key steps. To initiate, TEZip employs the recurrent neural network PredNet for training. PredNet's primary function is to predict expected image frames based on accessible frames. Subsequently, TEZip originates the disparities between these expected (predicted) frames and the real frames, generating d-(delta) frame that exhibits a higher inherent compressibility.

During the compression phase, we use the trained model to predict forthcoming frames. In this context,"Bi" represents

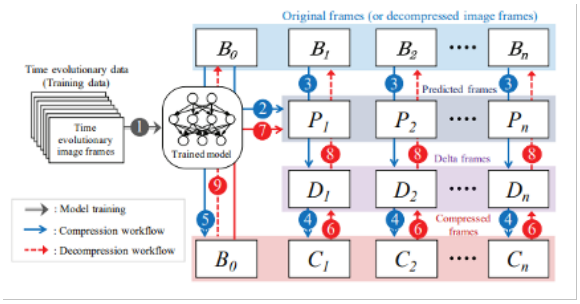


Figure 10: Workflow of TEZip

the *i*th original frame from the time evolutionary data, which serves as a reference before compression, while"Pi" stands for the predicted frame corresponding to "Bi" (2). Subsequently, we calculate the discrepancies between "Bi" and "Pi," resulting in the creation of "Di," which signifies the d-(delta) frame (3). We then apply a series of encoding techniques developed within our framework, culminating in "Ci," which denotes the final compressed frame of "Bi" (4). We note that the first frame, "B0," lacks a previous frame for prediction, so we retain it in its original form (5).

3.4. Performance Evalaluations

TEZip Performance Evaluation. The TEZip performance evaluation includes several types of instrument data, such as KITTI [51], XPCS [73], XCT2K/4K [73], Victoria [60], and Malaga [22]. In figure 10 compares with compression ratio performance TEZip and other compressors. Additionally, the initial evaluation of TEZip is based on the training root measure mean squared error (RMSE), discussed in Section 3. Table 1 lists RMSE values of the tested datasets.

Dataset	RMSE
kitti-1 [51]	0.265
kitti-2 [51]	0.436
xpcs [73]	0.141
xct2k [73]	0.055
malaga [22]	0.308
victoria [60]	0.301
Eumet-19 [3]	0.196

Table 1: RMSE as a % of the value range of prediction mechanism in TEZip with different datasets trained on the KITTI-1 dataset

In our study, we evaluated TEZip predictive model quality by segmenting the dataset into distinct subsets for training and testing. This approach subsequently evaluates prediction models for each training set across all testing datasets. Utilizing the test datasets, we calculated the root mean square error $RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - p_i)^2}$, that measures the average error between the predicted values and validation set on a per-pixel basis. We can further interpret this by dividing the RMSE by the value range (vr) and presenting the outcome as a percentage, that is, $RMSE_{rel} = RMSE/vr$. The original dataset has a value in the range of [0,255]. Tabel-1 shows the $RMSE_{rel}$ percentage for the model trained on the kitti-1 dataset. The results highlighted an error rate of less than 0.5 percent across all datasets which shows the prediction method reduces errors For additional information and results, refer to our previous work[122]. We'll also compare the quality of TEZip with another top compressor, they're tested at the same compression ratio, in Section 3.5. First, we evaluate the quality of the underlying prediction model used by TEZip as follows. First, we divide each of the various datasets into training and test subsets. We then train prediction models for each training set and evaluate it on the test sets from all datasets. With the test set and the prediction models, we can compute the Root Mean Square Error on the test set as follows $RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - p_i)^2}$. The RMSE measures the average error in the predictor vs the validation set per pixel. We can contextualize this further by dividing RMSE by the value range (vr) and expressing the result as a percent, i.e., $RMSE_{rel} = RMSE/vr$. The original dataset has a value in the range of [0,255]. Table 1 presents the $RMSE_{rel}$ percentages for a model trained on the kitti-1 dataset. These results demonstrate less than 0.5% error across all datasets which underscores the effectiveness of the prediction method in minimizing prediction errors. For more details and results please see our prior work[122]. We will consider the comparative quality between TEZip and another leading compressor at the same compression ratio between compressors later in Section 3.5.

ROBIN-SZ Performance Evaluation. ROIBN-SZ utilizes multiple levels of parallelism to scale to the line rate of data produced by serial crystallography. Generally, ROIBIN-SZ scales linearly with the number of nodes each processing different subsets of the detector panels used in an experiment. Since its development, ROIBIN-SZ has been tested at line rate speeds as part of the Exascale Computing Project, and is now used as part of the data reduction pipeline for serial crystallography at SLAC. For more details please see our prior work[150]

3.5. Lossy Compression Quality Performance

In the lossy compression approach, TEZip adapts to different error bounds (α). During experiments, we systematically adjusted α based on a method elaborated later in this section to cater to different datasets. To assess the performance of our lossy TEZip scheme, we conducted comparisons with established lossy compression methods such as SZ and zfp. Based on our time evaluation, we found that the compression time for lossy TEZip is higher. Also, when it comes to decompression, SZ significantly outperforms TEZip in terms of speed.

TEZip Image Quality Comparison. We applied the multiscale structure similarity index measure (MS-SSIM) to compare the image quality of decompressed frames between TEZip and ZFP, considering the maximum error bounds.The MS-SSIM index evaluates structural similarity at various scales between the image and the reference image. TEZip and zfp in term of the image quality of the decompressed frames for the same maximum point-wise relative error bounds. MS-SSIM index is calculated by combining the MS-SSIM between different versions

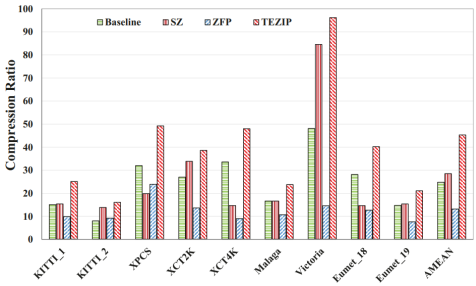


Figure 11: Compression ratios compared with baseline approach [127]

Dataset	TEZip	zfp
kitti-1 [51]	0.9726	0.9553
kitti-2 [51]	0.9669	0.9582
xpcs [73]	0.9988	0.9786
xct4k [73]	0.9692	0.9537
malaga [22]	0.9813	0.9639
victoria [60]	0.9948	0.9840

Table 2: Comparison of TEZip and zfp image quality using MS-SSIM at the same compression ratio

of the image and reference image. We measure the MS-SSIM score for each frame and then calculate the mean across all frames. Table 2 demonstrates that TEZip consistently brings quality images while enhancing compression. We have compared compression ratios with other compressors for different error bounds. TEZip compression ratio is the highest for all datasets at an absolute error bound of 1e-06 and 1e-05. For more information please see our recent work [137].

ROIBIN-SZ Reconstruction Performance. Careful attention must be paid to the tuning of compression pipelines to ensure that the electron density can be reconstructed accurately. We describe the process for tuning these parameters in our work [150] In Figure 12 we show the results of a reconstruction on a sample of Streptavidin. In Figure 13 we show the results of a reconstruction after compression and decompression using ROIBIN-SZ. As can be seen by visual inspection [2] of the two images, the reconstruction quality is high, presenting nearly identical reconstructions indicating that the scientific integrity of the data is preserved—this while reducing the volume of the data 46.44× and up to 154.18× when combined with non-hit rejection.

3.6. Future Work

Our research has been focused on developing compression techniques and frameworks for both upstream and downstream edge computing. Within the JLESC initiative, we have initiated three significant projects: StreamPressor, ROIBIN-SZ, and TEZip.

StreamPressor continues to focus on research for a framework that evaluates hardware-level compressor algorithms. Currently, we assess the behavior of digitally generated circuits for

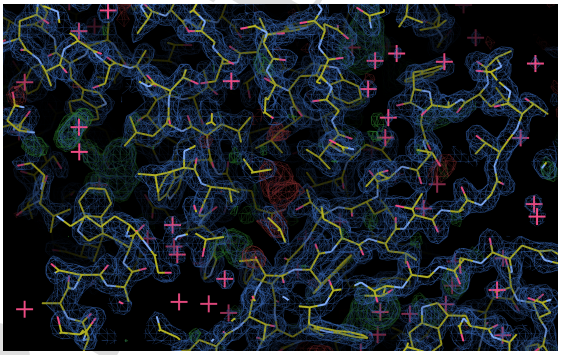


Figure 12: Original Streptavidin electron density reconstruction [150]

compressors using cycle-accurate software simulators. In future work, we plan to include FPGA-specific optimizations in the framework, aiming to create additional computational possibilities in the upstream edge stage. Additionally, we are exploring potential synergies with downstream compressors, such as ROIBIN-SZ and TEZip.

We're upgrading TEZip to predict future data more accurately. Our focus is on understanding what happens next in instrument data. It will make instrument data smaller and faster. By fine-tuning how TEZip predicts, we can manage changes in data more efficiently.

For ROIBIN-SZ we are investigating the performance tradeoffs when porting to the GPU and the appropriateness of different types of masking types to different kinds of datasets from beamlines. Adopting GPUs may allow us to reduce the amount of hardware required to compress the data.

We are improving TEZip to be more user-friendly by integrating it with LibPressio. This is important because many compressors, such as SZ, ZFP, MGARD, and TTHRESH, are written in a computer language called C/C++. By linking TEZip with LibPressio, we're making them work smoothly together. This means TEZip can use compressors like SZ through LibPressio, and other compressors can use TEZip. This enhancement makes TEZip more helpful and adaptable for various compression needs, bringing different compression technologies to-

[2] An astute reader may notice that this is not particularly precise to use visual inspection and it would argue that it is better to use image quality metrics such as PSNR, SSIM, or L_∞ norm, or to present a difference plot. However, there is some amount of natural variation in the reconstructed image derived from both the individual scientist's discretion regarding how to parameterize the reconstruction and the particular versions of software packages used which can result in small translations or distortions in the reconstruction. Therefore, what is acceptable in the reconstruction would be unduly penalized by these measures because they favor pixel-for-pixel reproducibility over semantic similarity. There are more quantifiable measures from earlier stages of the analysis which we present in [150], but we omit them here for space.

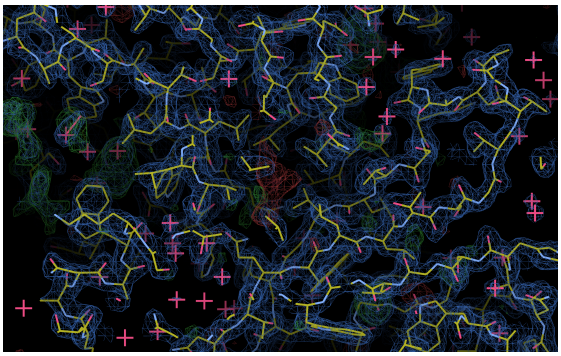


Figure 13: ROIBIN-SZ Streptavidin electron density reconstruction. 46.44× smaller than the original, and 154.18× when combined with non-hit rejection [150]

gether to handle data more effectively.

More broadly, more work is needed to expand the considerations of these kinds of techniques across different kinds of beamlines and experimental methodologies and detector types that produce different structures of data. We encourage the community to consider the compression approaches developed in our work.

4. Compression for Numerical Methods

The work described in this section was initiated as part of a collaboration between ANL and Inria in the context of the JLESC initiative [6] and then extended in [7]. It involves many collaborators who are not co-authors of the present survey but who deserve to be acknowledged; namely, Olivier Coulaud, Martina Innacito, Xin Liang, Gilles Marait, and Nick Schenkels.

Compression is ubiquitous in scientific computing in general and numerical linear algebra in particular. One of the most well-known methods for compression in this latter field is truncated singular value decomposition (TSVD). TSVD allows for compressing matrices in some optimum sense. However, there are fewer techniques for compressing vectors. An old but currently intensively studied method is to design numerical algorithms able to use mixed-precision arithmetic. These methods have been recently reviewed in [69, 5]. Still, data are stored in a way that sticks to the hardware processing capacity, typically in the form of 64-, 32-, and 16-bit words.

The idea of variable-accuracy storage is instead to rely on a compressor such as SZ to compress vectors independently from hardware constraints and apply it to the solution of large sparse linear systems. In [6] we investigated how to exploit that to compress the non-orthogonal basis of flexible GMRES (FGMRES) [128] while maintaining a result in high accuracy. In another study [7] we investigated how to compress the (orthogonal) basis of GMRES [129] to reach a prescribed accuracy (typically lower than the machine precision).

In the first study [6], we examined the FGMRES algorithm, which requires the storage of a basis for the Krylov subspace and for the search space spanned by the solutions of the preconditioning systems. We showed that the vectors spanning this search space Z can be compressed by looking at the combination of FGMRES and compression in the context of inexact Krylov subspace methods. This allowed us to derive a bound on the normwise relative compression error in each iteration. We used this bound to formulate several different practical compression strategies and validate and compare them through numerical experiments.

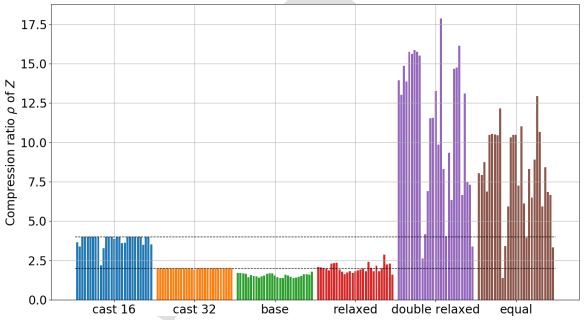


Figure 14: Total compression ratio of the search space Z for various compression strategies; each bar per strategy corresponds to a different matrix (see [6] for details). Horizontal reference lines are added at $\rho = 2$ and 4 corresponding to ideal (no extra iteration needed) 32- and 16-bit uniform storage, respectively.

In the second study [7], we investigated the backward stability of two backward-stable implementations of the GMRES method, namely, the so-called modified Gram–Schmidt (MGS) and the Householder variants. When the data representation of vectors introduces componentwise perturbations, we showed that the existing backward stability analyses of MGS-GMRES [119] and Householder-GMRES [151, 39] still apply. We illustrated this backward stability property in a practical context where the agnostic SZ lossy compressor is employed and enables the reduction of the memory requirement to store the orthonormal Arnoldi basis or the Householder reflectors. We also considered the normwise relative perturbations of the vector storage from an experimental perspective, showing numerically that the backward stability is maintained; that is, the attainable normwise backward error is of the same order as the normwise perturbations induced by the data storage. We illustrated it with numerical experiments in two practical contexts. The first one corresponds to the use of the SZ-agnostic compressor where vector compression is controlled normwise. The second one arises in the solution of tensor linear systems, where low-rank tensor approximations based on the tensor-train approach [118] are considered to tackle the curse of dimensionality.

We refer to the respective reports [6, 7] for further details.

5. Fast Data compression for shared memory mixed precision algorithms

On virtually all modern hardware architectures, the performance of sparse linear algebra is limited by the memory bandwidth. In consequence, to accelerate the algorithms, it is not

important to accelerate the arithmetic operations (e.g., by using low precision), but to accelerate the memory access, e.g., by reducing the data volume. Reducing the data volume can even come at the cost of additional in-register operations as processors are increasingly over-provisioned for FLOP/s. The approach we investigate for shared memory architectures is therefore based on the idea of decoupling the arithmetic precision from the memory precision and using in-register data compression. The algorithms we derive from this concept employ IEEE double precision for the arithmetic operations and then compress the double-precision data in registers to reduce pressure on the main memory bandwidth. Different concepts can be used for compression. Either IEEE standard low-precision formats, such as single or half-precision, or arbitrary storage formats, including compression schemes.

An example of an algorithm using the idea of data compression is the Compressed Basis GMRES (CB-GMRES) solver presented in [9]. GMRES is a Krylov solver that creates and expands an orthogonal basis of Krylov vectors that span a Krylov subspace. The CB-GMRES method, which we review in detail in the next section, compresses the basis vectors to reduce the memory access volume.

5.1. CB-GMRES

CB-GMRES is based on classical Gram-Schmidt with reorthogonalization. Similar to [7], it compresses the basis vectors to decrease the data access volume in the orthogonalization step, but CB-GMRES focuses on accelerating the algorithm's execution time. The compression is handled with the memory accessor introduced in [59], which is a lightweight software interface that allows us to quickly select a suitable memory format without touching the algorithm. In [9] we presented numerical experiments using two separate approaches: (1) Using IEEE formats half-, single- and double-precision as the storage format; and (2) Using a fixed-point format based on 32- and 16-bit integers that store the significand for each element with an additional scalar for each vector. Both of them are lossy compression techniques that compress each vector element individually.

Using IEEE formats as compression formats is easy to implement. We can simply leverage the existing cast functions to transform between the arithmetic double-precision and the chosen storage format. Since these operations are supported by hardware, the in-register conversion routine does not increase the runtime. This is similar to other mixed-precision algorithms with the difference that all arithmetic operations happen in double-precision.

Also, the fixed-point format is a lossy compression format. Conversion is significantly more complicated. Initially, the maximum value for each vector has to be determined. The fixed-point format is a lossy compression format as well. However, the conversion is significantly more complicated. Initially, the maximum value for each vector has to be determined. Then, all vector values are stored as integers by uniformly scaling between zero and the identified maximum value. This requires one additional double-precision value per vector that needs to be read for every value.

The experimental evaluation in [9] demonstrates that using single-precision as the storage format gives the best trade-off between convergence and runtime performance. The analysis is based on 49 test matrices and time-to-convergence analysis for both unpreconditioned and preconditioned GMRES. The experimental results show that convergence is very similar to double precision while providing an average speedup of 1.4. Using half-precision as the basis storage format allows for even faster GMRES iterations but impacts convergence, requiring either more iterations or not converging at all.

The 32-bit fixed-point format has similar convergence characteristics to single- and double-precision, but the conversion introduces a small overhead, making it inferior to single-precision storage overall. The 16-bit fixed-point basis storage is not competitive with the other strategies.

5.2. Evaluating CB-GMRES with compression using LibPressio

As the next step, we want to test the feasibility of using more sophisticated compression algorithms for the basis compression. We use LibPressio [149] to get access to many different compressors and initially only simulate the compression. For that, we compress the Krylov vectors in each iteration, followed by an immediate decompression. This simulates the information loss induced by the compression scheme without writing a time-consuming high-performance implementation first. At the same time, this experimental setup is only a first step, as it does not allow for evaluation of the runtime benefits. The experiment aims to identify compression strategies that successfully preserve the information in the Krylov basis vectors. Once compression strategies have been identified to be suitable suitable compression strategies have been identified, they can be integrated into the accessor for production runs. We use the following lossy compressors with various settings: Bit grooming [159], digit-rounding [34], SZ3 [101] with absolute error bounds, and zfp [103] with absolute error bounds.

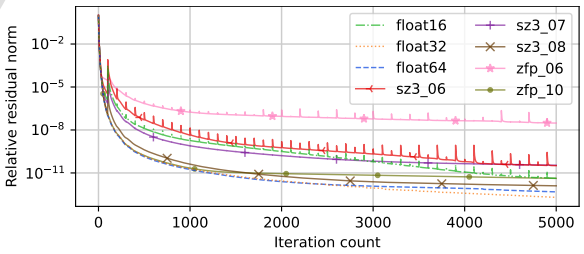


Figure 15: Residual norm plot for different precisions and the zfp and SZ3 compressions for the Transport matrix. The iteration count was fixed to 5000 for this experiment; the average number of bits per value is shown as the suffix of the legend the absolute error bound magnitude of zfp and SZ3 is shown as the suffix of the legend.

We run all mentioned compression strategies with all matrices in the original paper [9]. Both bit grooming and digit-rounding fail for some of the test cases, so we exclude them from further analysis.

Figure 15 shows the residual norm plots exemplary for the Transport matrix with the zfp and SZ3 compressors and the

Name	absolute error bound used	average bits per value	Final relative residual norm
float16	-	16	$8.2 \cdot 10^{-12}$
float32	-	32	$2.0 \cdot 10^{-13}$
float64	-	64	$4.8 \cdot 10^{-13}$
sz3_06	10^{-6}	12	$7.8 \cdot 10^{-11}$
sz3_07	10^{-7}	17	$3.9 \cdot 10^{-11}$
sz3_08	10^{-8}	41	$1.3 \cdot 10^{-12}$
zfp_06	10^{-6}	16	$4.2 \cdot 10^{-8}$
zfp_10	10^{-10}	28	$4.2 \cdot 10^{-12}$

Table 3: Compression details for the benchmark run shown in Figure 15

common IEEE floating-point precisions from the original paper [9]. Table 3 details the most important metrics of this benchmark run. For this experiment, we always run 5,000 iterations of CB-GMRES. Single- and double-precision display almost the same convergence behavior, while half-precision is two orders of magnitude convergence delay, which is a common occurrence. From the shown SZ3 and zfp convergence, only `sz3_41` `sz3_08` (which needs 41 bits per value on average) and `zfp_28` `zfp_10` (which needs 28 bits per value on average) show convergence behavior similar to double-precision, but both of them use roughly the same memory footprint as single-precision storage. When looking at all the results, none of our settings consistently beat the single-precision storage format. For some matrices, the absolute error bound of 10^{-8} with SZ3 results in an average of 24 bits per value and is enough to achieve the same convergence behavior as double-precision, but the same settings yield 60 bits per value for other scenarios requiring tuning for each matrix for the variable bitrate versions of SZ3 and zfp used here. While these particular algorithms perform poorly with respect to concerning the residual norm performance per bitrate, the fact that very low bitrates can perform comparability to double precision suggests that a more customized scheme may be possible for higher residual norm performance with lower bitrates. Additionally, for fast performance on hardware, decompression needs to be possible in registers or shared memory on the GPU and need fine-grained access to individual blocks of values. Neither the variable bitrate modes of zfp nor SZ3 allow for this kind of access as the decoding step requires space to decompress on account of the encoding stages[3]. An algorithm that addresses these concerns would need to be fixed-rate and use a very limited number of both instructions and memory to meet the performance demands compared to floating point truncation methods.

As part of our investigation, we also analyzed the correlation of the values of the Krylov vectors. After the first iteration, no correlation could be identified in the mantissa's of the floating point values, which explains the low compression rates for SZ3 and zfp[4]. Compressors such as SZ3 and zfp decorrelate data using techniques such as prediction or a near orthogonal transform, respectively. On spatially correlated data, the decorrelation stage of these compressors increases the sharpness and decreases the spread of the distribution of input values by recognizing the predictable nature of the data as a function of nearby points allowing the values to be encoded more efficiently. However, when these techniques are applied to random data, they at best have no effect, and at worst introduce overhead to store the metadata used to undo the decorrelation steps (such as prediction type indexes in SZ). This likely is the cause of the poor residual norm performance per bit of storage. Algorithms that respond to this concern would need to be designed not to exploit correlations in the mantissa bits.

[3] While zfp does offer a fixed rate mode, it presents other challenges for this kind of data we will address in the next paragraph.

[4] We do observe correlated patterns in the initial values of the Krylov vectors, but this is attributable to the initialization performed by the solver. After one iteration, the patterns we observe in the initial values disappear.

5.3. Future work

Future work will focus on analyzing the Krylov basis vectors in more detail to derive suitable compression schemes that can operate at very high bandwidths. As the compression happens in processor registers, the chunk size used in the compression needs to be small. At the same time, information loss can be reduced when compressing larger chunks. Additionally, the decompression must be lightweight in order to reach the memory bandwidth. SZ3 and zfp are too complex to achieve that, so we need a more simplistic strategy that might operate on small data blocks. We will investigate how to draw a balance to preserve information while achieving high performance.

6. Compression for AI

6.1. Introduction

Deep neural networks (DNNs) have rapidly evolved into the state-of-the-art technique for various artificial intelligence tasks across multiple science and technology domains, including image and vision recognition [132], recommendation systems [152], and natural language processing [29]. DNNs contain millions of parameters in an unparalleled representation, efficiently modeling complex nonlinearities. Numerous works [88, 136, 68] have suggested that using either deeper or wider DNNs is an effective way to improve analysis quality. Many recent DNNs have become significantly deeper and/or wider [153, 76].

The continually expanding size of DNN models generates a significant volume of intermediate data throughout the training process. This expansion results in a substantial increase in model size, which poses challenges for synchronization and inference. Similar to the challenges faced in scientific applications, the widening performance gap between computation and memory/storage systems presents data-related obstacles in deep learning. These challenges can be viewed primarily from three perspectives: (1) frequent large model/gradient synchronization against limited bandwidth; (2) extremely large intermediate data against limited GPU memory during training; and (3) effective handling of sizable input data for the model within the constraints of limited memory and bandwidth.

6.2. DNN Model Compression

As pointed out by Wang et al. [153], the deep learning community acknowledges that increasing the scales of DNNs can improve the inference accuracy of image recognition tasks. For example, a 9-layer AlexNet, proposed by Krizhevsky et al. [88], won the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [91] with a top-5 accuracy of 83%. In the 2014 ILSVRC, a 22-layer GoogLeNet [136] proposed by Szegedy et al. further improved the top-5 accuracy record to 93.3%. In the 2015 ILSVRC, He et al. [68] proposed a 152-layer ResNet, which set a new record of 96.43%. More recently, models such as GPT-3 [49] have exceeded 100 billion parameters, offering groundbreaking capabilities beyond those of conventional neural networks. This trend suggests that networks will continue to grow larger. Moreover, we note that during training, a model's gradient typically shares a size equivalence with the model itself. Consequently, the substantial model size presents communication challenges when exchanging gradients during training and difficulties in deploying these models on platforms with limited resources. Researchers at JLESC have proposed advanced lossy compression techniques for models aimed primarily at facilitating deployment on resource-limited platforms.

Specifically, Jin et al. [76] proposed DeepSZ, a lossy compression framework for DNNs. DeepSZ adapts the SZ lossy compression framework [36, 139, 98] to the context of DNN compression. This work enhances the efficiency of deploying DNN models on end devices with constrained bandwidth. Additionally, it also enables the storage of a larger number of models on each device, thereby significantly broadening their utility and application scope. In this adaptation, SZ achieves a much higher compression ratio for the compression of nonzero weights compared with other state-of-the-art compressors such as ZFP [103], especially due to its efficient linear-scaling quantization, in contrast to the simple vector quantization applied to the original weights in other related work [57, 65]. The general workflow of the DeepSZ framework is presented in Figure 16. As illustrated, DeepSZ consists of four key steps: network pruning, error-bound assessment, optimization of error-bound configuration, and generation of the compressed model. The first step involves adopting network pruning to reduce network complexity and mitigate the overfitting problem caused by a large number of parameters. The second step applies error-bounded lossy compression to the pruned layers and assesses the impacts of different error bounds on the inference accuracy for each layer. Based on the degradation of inference accuracy, DeepSZ identifies a feasible range of error bounds for each layer, collecting results on inference accuracy degradation and compressed layer size based on these bounds. This step effectively narrows the range of optimal error bounds for each layer. The third step determines the best-fit error bound for each layer based on the narrowed feasible range from the second step, compressing the network as much as possible while meeting the user-set inference accuracy requirement. The fourth step generates the compressed network based on the optimized error bounds and the best-fit lossless compressor.

6.3. Activation Data Compression

Training deep and wide neural networks has become increasingly challenging. While many state-of-the-art deep learning frameworks, such as TensorFlow [4] and PyTorch [120], can provide high training throughput by leveraging massive parallelism on general-purpose accelerators such as GPUs, one of the most common bottlenecks remains the high memory consumption during the training process. This issue is particularly acute considering the limited on-chip memory available on modern DNN accelerators. The primary cause is the ever-increasing size of the activation data computed during the training process.

Training a neural network involves numerous training epochs to update and learn the model weights. Each iteration includes forward and backward propagation. The intermediate activation data (the output from each neuron) generated by every layer is commonly kept in memory until backpropagation reaches that layer again. Several works [27, 53, 126, 30, 43] have highlighted the large gap between the time when activation data is generated in forward propagation and the time when it is reused in backpropagation, especially when training very deep models.

To address these challenges, researchers at JLESC propose a memory-efficient deep neural network training framework [82] named COMET (for Compression Optimized Memory-Efficient Training), which compresses activation data using adaptive error-bounded lossy compression. The key insights explored in this work include (i) the impact of compression errors in the activation data on the gradients and the entire CNN training process under strict error-controlling lossy compression can be theoretically and experimentally analyzed, and (ii) validation accuracy can be well maintained based on adaptive, fine-grained control over error-bounded lossy compression (i.e., compression error). This is the first work to investigate the impact of lossy compression error during CNN training and to leverage this analysis to significantly reduce memory consumption for training large CNNs while maintaining high validation accuracy.

An overview of COMET is shown in Figure 17. COMET iteratively repeats the process shown in the figure for each convolutional layer in every iteration. COMET mainly includes four phases, as shown in the figure from left to right: (1) parameter collection of current training status for adaptive compression, (2) gradient assessment to determine the maximum acceptable gradient error, (3) estimation of compression configuration (e.g., absolute error bound), and (4) compression/decompression of activation data with a modified cuSZ, the CUDA implementation of SZ (detailed in §7.3). Additionally, this research includes an analysis of the error-bound control scheme for lossy compression of activation data.

6.4. Input Data Compression for Inference

Another significant bottleneck in deploying DNNs for real-world applications is managing the extensive data collected from the source. In scenarios such as pedestrian detection, the DNN is typically hosted on cloud-based services, with end devices responsible only for collecting and sending data to the host for inference. A key challenge in such setups is the frequently restricted bandwidth of these end devices. Researchers at JLESC

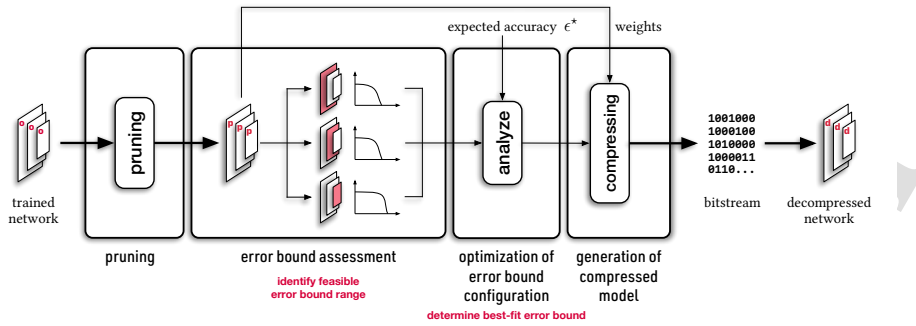


Figure 16: Overview of the DeepSZ framework for neural network compression.

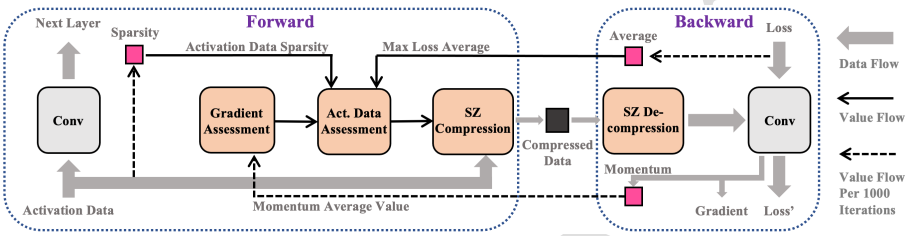


Figure 17: Overview of our proposed memory-efficient CNN training framework COMET.

have introduced frameworks designed to dynamically adjust the lossy compression configuration for the data [124]. This adaptive approach aims to maintain high detection accuracy across a diverse range of environmental conditions.

Rahman et al. [124] proposed a dynamic error-bounded lossy compression strategy to mitigate the bandwidth requirements in real-time vision-based pedestrian safety applications. This study introduced a dynamic feedback control system that adjusts the compression level dynamically to maintain the same detection accuracy as a system communicating raw lossless video data. The approach involves using a video compression unit installed on a roadside video monitoring camera. This unit compresses the raw video stream using a predetermined tolerance level, based on the peak signal-to-noise ratio that is dynamically adjusted in response to varying conditions, such as rain and darkness.

Following compression, the wirelessly transmitted compressed video streams are processed by the roadside edge computing infrastructure, which includes three main components: (1) a collection of calibrated pedestrian detection models tailored for diverse environmental conditions, (2) an active pedestrian detection model designed to process video images, and (3) an environmental condition detection model tasked with identifying the prevailing environmental conditions depicted in the video.

6.5. Performance

In memory compression experiments, results based on the tested neural networks show that DeepSZ [76] can achieve compression ratios of up to 116×, outperforming the second-best approach by up to 1.43×. Our experiments with four Nvidia

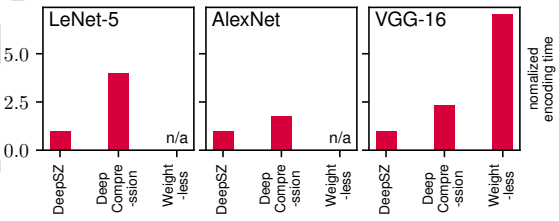


Figure 18: Normalized encoding time with different solutions on GPUs.

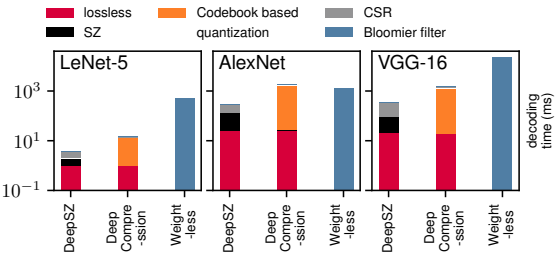


Figure 19: Breakdown of decoding time with different solutions on CPU. Each sub-figure shows the comparison with a given model, including LeNet-5, AlexNet, and VGG-16.

Table 4: Inference accuracy of DeepSZ compressed LeNet-5, AlexNet, and VGG-16

Neural Network	Top-1 Accuracy	Top-5 Accuracy	layer' Size	Compress Ratio
LeNet-300-100 original	98.35%	-	1056 KB	
LeNet-300-100 DeepSZ	98.31%	-	**19.1 KB**	55.8×
LeNet-5 original	99.13%	-	1620 KB	
LeNet-5 DeepSZ	99.16%	-	**28.3 KB**	57.3 ×
AlexNet original	57.41%	80.40%	234.5 MB	
AlexNet DeepSZ	57.28%	80.58%	**5.15 MB**	45.5 ×
VGG-16 original	68.05%	88.34%	494.5 MB	
VGG-16 DeepSZ	67.80%	88.20%	**4.277 MB**	115.6 ×

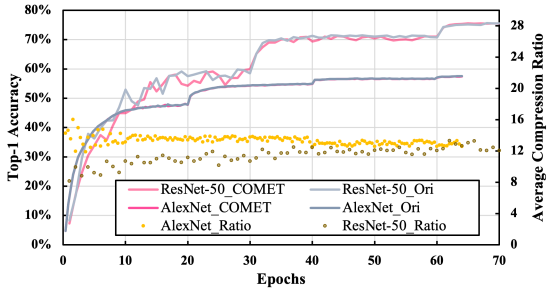


Figure 20: Comparison of validation accuracy between baseline training and COMET (batch size = 256). Lines represent the validation accuracy during training; dots represent the compression ratio of COMET on the secondary y-axis.

Tesla V100 GPUs demonstrate that DeepSZ can obtain a performance improvement in the encoding of 1.8× to 4.0× compared with the previous state of the art. Moreover, DeepSZ can improve the decoding performance by 4.5× to 6.2× compared with the second-best solution. DeepSZ also provides high flexibility to balance the compression ratio and inference accuracy.

Table 4 illustrates the compression ratio achieved by DeepSZ on multiple tested neural networks. Additionally, Figures 18 and 19 show that DeepSZ has lower encoding and decoding time overheads compared with Deep Compression [65] and Weightless [125], the two works prior to DeepSZ.

In activation data compression, COMET [82] significantly reduces memory usage by up to 13.5× with little or no loss in accuracy. Compared with the previous state-of-the-art compression-based approach, COMET offers an improvement in memory reduction of up to 1.8×. By leveraging the saved memory, COMET can enhance end-to-end training performance (e.g., approximately 2× improvement on AlexNet).

Figure 20 illustrates the accuracy curve and the compression ratio for AlexNet and ResNet-50. We observe that these two curves are very close to each other, indicating that COMET does not noticeably affect the validation accuracy. In the early stages of training, the compression ratio can be slightly unstable due to the relatively large changes in the model itself. We note that the compression ratio will change slightly when the learning rate changes since the learning rate is significant only when updating the gradient to the weights.

Table 5 shows the average compression ratio of COMET compared with a JPEG-based solution [43], which employs a hardware-implemented image-based compressor to achieve up to 7× compression ratios. COMET outperforms this solution by 1.5× and 1.8× on ResNet-18 and ResNet-50, respectively.

In the context of inference data compression, Table 6 illustrates the bandwidth reduction rates achieved by the framework introduced by Rahman et al. [124]. The data in the table demonstrates that, across various conditions, this framework consistently delivers high compression ratios for videos, effectively mitigating communication bottlenecks.

Table 5: Comparison of validation accuracy between baseline training and COMET; comparison of compression ratio between JPEG-ACT and COMET.

Neural Nets		Top-1 Accuracy	Peak Mem.	Max Batch	Conv. Act. Size	COMET	JPEG-ACT
AlexNet	b.	57.41%	2.17 GB	512	407 MB		
	c.	57.42%	0.85 GB	2048	**30 MB**	13.5×	-
VGG-16	b.	68.05%	17.29 GB	64	6.91 GB		
	c.	68.02%	5.04 GB	256	**0.62 GB**	11.1 ×	-
ResNet-18	b.	67.57%	5.16 GB	256	1.71 GB		
	c.	67.43%	1.37 GB	1024	**0.16 GB**	10.7 ×	7.3 ×
ResNet-50	b.	75.55%	15.57 GB	128	5.14 GB		
	c.	75.51%	4.40 GB	512	**0.46 GB**	11.0 ×	6.0 ×

b.= baseline, c.= compressed

6.6. Future Work

Our upcoming research of compression for AI focuses mainly on three directions: (1) reduce the model/gradient size to enhance communication performance during training, (2) further reduce memory consumption during training through the compression of activation data, and (3) reduce the cost of loading training data and/or checkpoints to boost overall training performance. s Specifically, in (1), our plans involve introducing multiple frameworks to enhance communication efficiency in various scenarios. First, we target the compression of communication data for deep learning recommendation model training, a process that typically employs a hybrid training infrastructure encompassing data parallelism, model parallelism, and pipelining. We aim to propose an adaptive framework to compress both gradient and tensor data, thereby mitigating communication overhead. Second, we plan to establish a comprehensive compression workflow to enhance the performance of all-to-all communication, a widely employed operation in machine learning. Third, we intend to develop a framework tailored for reducing gradients in K-FAC training to significantly improve training performance. We plan on a compression scheme to control the distribution of compression errors in this task, thereby effectively limiting accuracy degradation for K-FAC training.

In terms of activation data compression, in (2) we noticed distinct advantages from two techniques: recomputing and data migration, apart from lossy compression for reducing memory consumption. Our goal is to create a hybrid framework that combines all three techniques to improve training performance. Additionally, we plan to explore lossy compression for activation data of a wide range of layers to identify which types of

Weather condition	Baseline Pedestrian Detection Accuracy (no compression)	Dynamic EBLC Framework Pedestrian Detection Accuracy	Improvement in Pedestrian Detection using Dynamic EBLC Framework	Maximum Constant Rate Factor (CRF) (or Minimum PSNR)	Required Bandwidth (MBits/sec)	Bandwidth Reduction
Normal	98%	97%	-1%	30 (41dB)	0.53	18×
Light Dark	93%	97%	4%	30 (41dB)	0.68	14×
Medium Dark	92%	97%	5%	20 (43dB)	1.01	9.5×
High Dark	90%	97%	7%	10 (56dB)	4.05	2.5×
Light Rain	89%	97%	8%	10 (56dB)	5.15	1.5×
Medium Rain	88%	97%	9%	0	9.82	0×
Heavy Rain	83%	97%	14%	0	9.82	0×

Table 6: Maximum compression ratio achieved for different weather conditions.

layers can be efficiently compressed without significant accuracy degradation.

In (3), we aim to implement compression techniques for both training data and checkpoint data. We observed that in numerous AI applications for scientific purposes, the input data often surpasses the scale of traditional AI image processing. As a result, a substantial portion of the model's training time is allocated to loading the extensive input data. Offering frameworks that efficiently reduce the size of input data without compromising convergence speed and accuracy can effectively reduce the expenses associated with transferring large input data.

7. Optimizing Compression on the GPU

7.1. Introduction

Recent advancements in scientific computing, notably marked by the transition to exascale (over 1e18 FLOPS) computing [1], impose data challenges in two ways: the sheer volume of data and the rapid rate of data generation. Data compression has increasingly been researched to meet these challenges. Notably, a lossy compressor designed for scientific data reduction should fulfill three primary design goals: high-fidelity preservation in accordance with specific domain science constraints, a high compression ratio, and robust processing capability. On the other hand, GPUs have demonstrated significantly higher data processing capabilities than CPUs have in modern heterogeneous computing systems, making them well suited to address data generation issues by conducting in situ reduction.

SZ is a state-of-the-art error-bounded lossy compression framework for scientific data, meeting the requirements of high-fidelity preservation and a high compression ratio. However, as shown in prior work [36, 139], SZ suffers from (1) low single-core compression and decompression speeds at hundreds of megabytes per second and (2) limited multicore performance. Therefore, adopting a data-parallel programming model and leveraging GPUs can be an attractive solution to keep pace with the high data production rate. Today, GPUs contribute significantly to compute capabilities thanks to a mature ecosystem; their high programmability, as opposed to other heterogeneous hardware platforms such as FPGAs [144], enables easier parallelization.

Parallelizing SZ requires addressing two main issues. First, a loop-carried data dependency exists among the loop iterations of SZ's prediction and error quantization step—not until iteration i is finished can iteration $(i + 1)$ start. This tight data dependency incurs extraordinarily costly synchronization to ensure correctness.[5] Second, during Huffman coding, a variable-length encoding step in the SZ algorithm, the uneven data access pattern for grouped threads hurts performance by breaking the rigid SIMD parallel pattern (known as warp divergence).

The challenge centers on unleashing GPUs' potential by identifying parallelisms in the existing SZ algorithm and balancing the three primary design goals—achieving a high compression ratio, high throughput, and high data quality. Therefore, over the past five years, JLESC researchers have proposed a series of works to address the low-throughput issue and expand the usability of SZ compression. A summary of the proposed GPU-based compressors is given in Table 6.

Work	Generality	Speed	Quality	Comp. Ratio
cuSZ(+)	general-purpose	baseline	baseline	high
FZ-GPU	general-purpose	high	baseline	balanced medium
cuSZp	special (1D only)	high (end-to-end)	baseline	medium
cuSZ-I	special (3D only)	medium	highest	highest

Table 7: Comparison of cuSZ, FZ-GPU, cuSZp, and cuSZ-I.

7.2. The State of the Art

Given the surge in GPU-based HPC systems and their applications, there has been a proliferation of GPU-based compression methods for scientific data. Notable examples include cuZFP [32], MGARD [55], and nvCOMP's Bitcomp [115]. These GPU-based compressors offer significantly higher compression throughputs compared with CPU-based compressors.

cuZFP, employing a transformation-based approach, allows users to specify their desired bitrate, representing the average number of bits per value after compression. In contrast, Bitcomp (a quantization-based compressor) and MGARD (a multigrid-based compressor) enable users to set the maximum permissible error. Notably, cuZFP in fixed-rate mode consistently deliv-

[5] Preliminary results show 30 to 40 GB/s for the major kernel on V100, which is approximately 1/20 of memory copy speed.

ers superior compression throughput, while MGARD in error-bounded mode tends to achieve a higher compression ratio. Although the principles underlying nvCOMP's Bitcomp remain undisclosed because of its closed-source nature, our evaluations indicate that it employs prediction processes and consistently attains a relatively high compression ratio while maintaining commendable compression throughput.

However, these GPU-based lossy compressors face certain challenges that warrant resolution. For instance, cuZFP exclusively supports the fixed-rate mode, restricting its applicability in scenarios necessitating strict control over the error bound in reconstructed data compared with the original data. MGARD's GPU version, on the other hand, necessitates CPU involvement in the compression process, resulting in performance bottlenecks attributed to frequent data transfers between the CPU and GPU. MGARD's GPU version, on the other hand, integrates the decomposition and Huffman encoding process into the pipeline, which results in the relatively low throughput (single-digit GB/s) [54], which is impractical to fit the HPC real-time data processing requirements. Additionally, nvCOMP's Bitcomp, being proprietary, lacks transparency in its operation, with NVIDIA's provided APIs being overly high-level, thereby limiting its adaptability in practical applications. In light of these limitations, our objective is to develop a new series of GPU-based compressors to address these issues comprehensively.

7.3. Proposed Design

We now introduce our proposed GPU compressors.

cuSZ *Framework.* The cuSZ work The work cuSZ [145] and cusz+ [143] is the first error-bounded lossy compressor on GPUs, adapting the SZ framework for scientific data. The work features a practical framework on GPUs to achieve high throughput. The basic framework contains the fully parallelized Lorenzo-based prediction-quantization and coarse-grain multi-byte symbol-enumerating Huffman encoding, with a further lossless encoding. The lossless encoding can be dropped for the subsequent high throughput. The key novelty of cuSZ/cuSZ+, on top of the practice on GPUs, is the demonstration that the achieved capability originates from both algorithm design and the performance optimization in a hardware-software co-design manner, which are detailed in cuSZ and cuSZ+ work. The key novelty of cuSZ/cuSZ+ lies in demonstrating that the achieved capability originates from both algorithm design and the performance optimization, considering the practical utilization of GPU architecture, which is detailed in cuSZ[145] and cuSZ+ [143] work.

FZ-GPU. The FZ-GPU compressor provides an alternative compression pipeline to broaden its usability based on its data processing capability. Based on the existing prediction-quantization that generates the error-quantization code from the original cuSZ framework, a novel compression pipeline is proposed by incorporating GPU-based bitshuffle, a newly designed high-throughput-oriented lossless coding technique. The prediction and bitshuffle operations result in substantial data sparsity, so FZ-GPU incorporates a fast-scan GPU kernel. This kernel efficiently skips data blocks containing all-zero values and selectively writes non-all-zero blocks, aligning well with the compression pipeline's requirements. Compared with cuSZ/cuSZ+, FZ-GPU utilizes a faster lossless encoder in the pipeline and maintains balanced compression ratios achieves a gracefully degraded compression ratio but higher throughput. Consequently, it manages to achieve a notably higher host-device data transfer rate performance compared with cuSZ/cuSZ+.

cuSZp. cuSZp represents a GPU-based error-bounded lossy compression approach engineered to optimize end-to-end throughput. This method is constructed on a series of lightweight bit-level operations implemented within GPU registers, complemented by finely tuned global synchronization mechanisms. This concerted strategy culminates in the achievement of remarkably swift throughput. The methodology underlying cuSZp can be briefly outlined through four key steps. (1) Quantization and prediction: In cuSZp, the quantization and prediction stages resemble those of other SZ-based compressors. Quantization introduces the sole lossy element within the compression pipeline, introducing errors while concurrently enhancing data redundancy to facilitate greater compressibility. The prediction stage serves to decorrelate input data based on dimensional patterns. (2) Fixed-length encoding: This step entails preserving a fixed-length bit representation, further augmenting compression ratios. (3) Global synchronization: In GPU-based lossy compression methodologies global synchronization is essential. In cuSZp, this process leverages a high-performance prefix sum technique for local offset calculations and a global lock mechanism for global offset computations. (4) Block bitshuffle: cuSZp harnesses the bitshuffle technique to introduce additional redundancy at the bit level, thereby enhancing the compressibility of processed data.

cuSZ-*I.* Based on the cuSZframework and CPU QoZ[106], cuSZ-I is using cubic spline interpolation in place of the generic Lorenzo prediction to achieve a higher compression ratio and improve the data reconstruction quality. Also, with more accurate prediction before the lossless encoding stage, the error-quantization codes are much more condensed so that higher data compressibility can be better utilized in the lossless encoding stage. With the original Huffman encoding, the proposed cubic spline interpolation shows its advantage in the compression ratio, even with the data-representation-restricted Huffman encoding.[6] In addition, by appending another stage of lossless encoding, cuSZ-I can steadily achieve an even higher compression ratio, outperforming any other known GPU-based compressors. This work profiled the best-performing lossless encoding—in this case, the proprietary Bitcomp from NVIDIA's nvCOMP collection. Another dictionary-like lossless encoding can also leverage the found compressibility. We note that in the data-movement use scenario, cuSZ-I does not provide the best-in-class processing speed because of its significantly higher compression ratio; however, it shows a notable advantage in the equivalent data-transfer rate. Also, based on its improved data quality, rate-distortion or transfer-rate distortion profiling demonstrates that cuSZ-I could

[6] Huffman encoding needs at least 1 bit to represent each quantization code.

be the preferable solution based on the knowledge of data in specific scientific applications.

7.4. Performance

In our evaluation, we use the representative scientific dataset to demonstrate the capability of the above-mentioned GPU-based compressors, namely JHTDB, Miranda, NYX, QMCPack, RTM, and S3D (detailed in Table 8). They have been widely used in prior works [11, 14, 19, 27] and are representative in the production-level simulations. Most of them are open data from the Scientific Data Reduction Benchmarks suite [34]. Also, we employ a set of metrics to assess the performance of the compression techniques. These metrics encompass the following aspects:

1. Compression ratio: This metric is a fundamental measure widely used in compression research. It quantifies the ratio between the size of the original data and the size of the compressed data. Higher compression ratios indicate a more efficient aggregation of information in comparison with the original data. We note that in our evaluations, we specifically calculate the bitrate, which represents the average number of bits required per value after compression. Given that our evaluation datasets consist of single-precision floating-point data, the bitrate is computed as 32 (bits) divided by the compression ratio. We utilize this metric to evaluate how each compressor performs in terms of data quality at a specific bitrate. Figure 22 effectively displays the compression ratio as bitrate.

2. Distortion: Assessing distortion is essential for evaluating the quality of data reconstruction in lossy compression. In our work, we primarily utilize the peak signal-to-noise ratio to quantify the quality of distortion. Similar to prior research, we construct rate-distortion curves to facilitate a fair comparison among different compressors and their various compression modes. These curves enable a comparison of distortion quality at equivalent bitrates. The PSNR is depicted in a rate-distortion map, as illustrated in Figure 22.

3. Compression and decompression throughput: This metric gauges the rate at which a compressor can process data within a unit of time. It represents a key advantage of utilizing GPU-based compression over CPU-based alternatives. Figure 21 presents the result of throughput for both compression and decompression processes.

These evaluation metrics collectively provide a comprehensive assessment of the compression techniques' performance, encompassing compression efficiency, data quality, and processing speed.

7.5. What's Next

Workflow Optimization. The recently updated cuSZ-I demonstrates that the prediction-quantization step creates compressibility for subsequent lossless encoding. However, the actual reduction in data size is achieved by our in-house Huffman encoding and Nvidia's proprietary Bitcomp. Therefore, the workflow needs to be optimized in the following ways: (1) replace the proprietary component (e.g., with a customized Finite State Entropy encoder) to benefit the compressor development in an end-to-end manner and facilitate deployment; and (2) introduce a compressibility model to further optimize the efficiency of lossless encoding, utilizing the same intermediate data from the prediction-quantization step.

JHTDB	10 files	dim: 512×512×512	total size: 5 GB
	Domain: turbulence; the Johns Hopkins Turbulence Database [97] with direct numerical simulation (DNS) turbulence data. We extracted 512^3 data blocks of pressure field over time steps.		
Miranda	7 files	dim: 256×384×384	total size: 1 GB
	Domain: Large-eddy simulation of multi-component turbulent flows via a radiation hydrodynamics code [114].		
NYX	6 files	dim: 512×512×512	total size: 3.1 GB
	Domain: cosmology; cosmological hydrodynamics simulation based on adaptive mesh [10].		
QMCPack	1 files	dim: 288×115×69×69	total size: 612 MB
	Domain: quantum; an open source ab initio quantum Monte Carlo package for the electronic structure of atoms, molecules, and solids [123].		
RTM	37 files	dim: 449×449×235	total size: 6.5 GB
	Domain: seismic wave; reverse time migration for seismic imaging [86].		
S3D	11 files	dim: 500×500×500	total size: 5.1 GB
	Domain: combustion; a simulation dataset of combustion process [**?**].		

Table 8: Datasets used in GPU compressor evaluations.

Multiple Backends. In-production and upcoming supercomputers are equipped with GPUs from multiple vendors, posing challenges in developing and maintaining GPU compressors for them. With the parallel algorithm in data processing developed, we can adapt compressors, originally based on NVIDIA's CUDA programming model, to other backends. This adaptation may include creating our portability layer or adopting Kokkos.[7] Furthermore, the presence of multiple backends implies the need to study performance models to maximize the usability of the proposed GPU compressors.

8. Scheduling Compressed I/Os

The work [77] described in this section was conducted as part of a collaboration between ANL and Inria. In addition to the coauthors of the present survey this work involved Daoce Wang and Yves Robert.

8.1. Introduction

Scientific applications typically use parallel I/O libraries such as the Hierarchical Data Format 5 (HDF5) [142] for reading and storing their data. HDF5 offers dynamically loaded filters [67], including lossless and lossy compression [35], which enable the automatic storage and access of data in compressed formats.

Asynchronous I/O from parallel I/O libraries can help alleviate I/O bottlenecks by overlapping I/O operations with computations, enhancing the end-to-end performance of HPC applications. However, the current implementation of asynchronous

[7] Kokkos is a C++ performance portability programming ecosystem that addresses two major concerns regarding multiple backends: parallel execution description and memory abstraction across heterogeneous hardware.

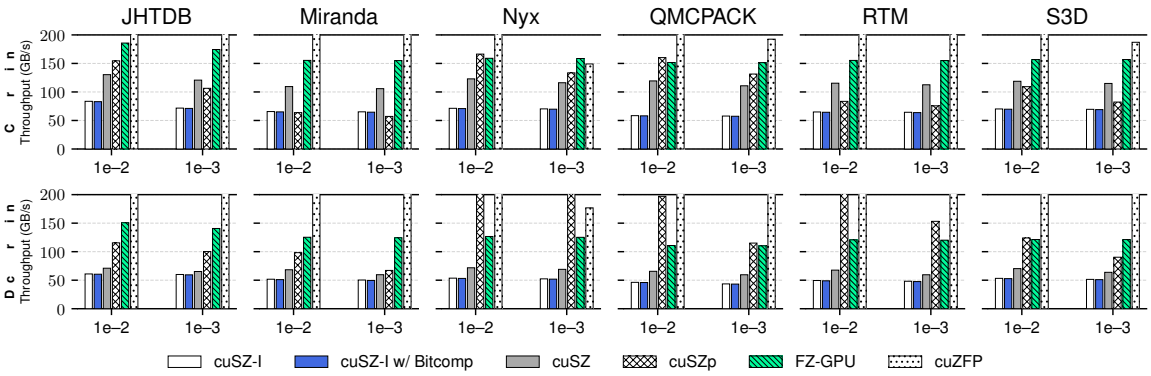


Figure 21: Compression and decompression throughputs, with all-kernel combined, for GPU compressors in GB/s on NVIDIA A100.

I/O has certain limitations. It supports either (1) asynchronous compression and I/O [81] or (2) asynchronous I/O and computation [165]. In the former scenario, the data writing and computation are still executed sequentially. In the latter scenario, one would miss the opportunity to utilize lossy compression. Moreover, asynchronous I/O typically occurs in a background thread to prevent interference with the main computational thread responsible for running the simulation. Nevertheless, it can still potentially result in performance degradation by introducing interference with other background tasks related to communication within the simulation, especially without proper scheduling [138, 146].

In this section we present a design that enables concurrent execution of compression, I/O, and computation. As scientific applications increasingly adopt GPUs [10, 45, 21], it is crucial to ensure that the GPUs are utilized continuously with minimal interruptions. However, the main and background threads experience periods of idle time while processing computational tasks. Our goal is to harness these idle periods to perform data compression and I/O tasks. To ensure that compression and I/O tasks do not impact overall execution, we carefully schedule compression tasks during the idle times of the corresponding computation thread. Avoiding any delays in existing processing tasks is essential, since such delays would hinder GPU work and potentially result in a slowdown of the entire execution. Our framework includes three key components: fine-grained compression, a compressed data buffer, and a shared tree for Huffman coding. The fine-grained compression algorithm allows for independent compression of data blocks, thereby enhancing task-scheduling efficiency. The compressed data buffer enables the overlapping of compression and I/O tasks. The shared Huffman tree minimizes the overhead associated with building the Huffman tree for compressing small data blocks.

8.2. Task Scheduling with Lossy Compression

Problem formulation. In most scientific simulations, each process is responsible for handling a segment of the entire data space. We focus on iterative HPC applications. We assume that the execution pattern is highly similar between consecutive iterations. Therefore, for scheduling one iteration we use the recorded characteristics of the previous iteration (number, starting and ending times of computation tasks, etc.). We note that slight variations in the required task lengths and dates between neighboring iterations may result in some performance degradation, since the proposed task scheduling takes imperfect data as input. As demonstrated in the evaluation section, however, these variations do not significantly impact the effectiveness of the proposed solution.

During each iteration, the HPC application executes a series of tasks on both the CPU and GPU. Usually, each node operates multiple processes, with each process assigned to a dedicated GPU along with a subset of CPU cores. Within each process, there are multiple threads: one associated with the GPU (for computation) and others with the CPU cores (for computation, compression, I/O). Parallel I/O operations are collaboratively executed by the background thread in each process. Our design choice aims to prevent any interference with the GPU's simulation/computation performance. We ignore the GPU and concentrate on the computing, compression, and I/O tasks performed on the main thread and on the background thread by the CPU.

Periodically, either at each iteration or every l iterations for some fixed value l, the application generates a large amount of data that needs to be written to the storage system. We apply lossy compression to such data. A process is responsible for managing a certain number of data fields, and each of these fields undergoes our proposed fine-grained compression. Task scheduling must ensure that compression tasks do not interfere with computing tasks. In particular, this means that the execution of each computation task should start and end at the same times whether we perform lossy compression or not.

Each compression task is followed by a corresponding I/O task that writes the compressed data to the storage system. We call a *job* the pair formed by a compression task and the corresponding I/O task. We assume that the I/O operations and communications of the application all occur on the background thread. By construction, these tasks are separated from and cannot interfere with either the computing tasks or the compression

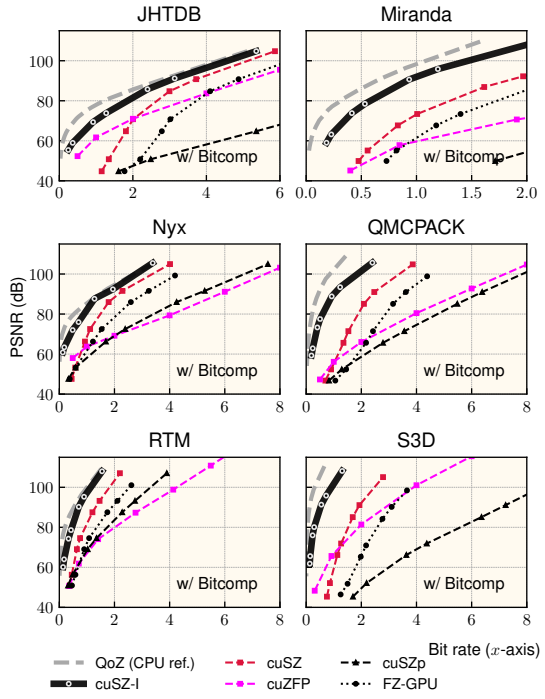


Figure 22: Rate distortion of the proposal GPU compressors, with an extra lossless encoder enabled to show the best possible rate-distortion performance.

tasks. However, there are core tasks associated with computing that are executed on the background thread. Like the computing tasks, these tasks have known execution intervals that must be respected when scheduling the I/O tasks.

All the scheduling constraints can be summarized as follows. On the compute thread, the compute tasks are immovable obstacles that the compression tasks must avoid. Similarly, on the background thread, the core tasks are immovable obstacles that the I/O tasks must avoid. The two threads are interdependent, since an I/O task cannot start until the corresponding compression task completes. This is the only constraint on the order of compression and I/O tasks, which can otherwise be scheduled in any order compatible with these dependence constraints. The goal of the task-scheduling algorithm is to find an ordering for compression tasks and for I/O tasks to minimize the overall execution time of the current iteration.

Problem Analysis. In scheduling theory nomenclature, the problem of scheduling the compression and I/O tasks is a flow-shop problem with two machines: The first one corresponds to the computation thread and the second to the background thread. Our problem is exactly a flow-shop problem with two machines with *deterministic unavailability intervals* on both machines and with *non resumable* jobs. The flow-shop problem with two machines, in the absence of unavailability intervals, is optimally solved by Johnson's algorithm [83]. The problem with non-resumable jobs becomes NP-complete as soon as there is at least one unavailability interval [92]; it becomes non-approximable by any constant factor (unless P=NP) as soon as there are at least two unavailability intervals on one machine [23].

Adaptation of Johnson's Algorithm. To schedule tasks, we adapt Johnson's algorithm, which builds an optimal solution in the absence of unavailability intervals. This algorithm works as follows. The jobs are partitioned in two sets $\mathcal{M}_1$ and $\mathcal{M}_2$ where jobs in $\mathcal{M}_1$ are exactly the jobs whose compression task is not longer than its I/O task. Jobs in $\mathcal{M}_1$ are sorted by non-decreasing duration of their compression task and jobs in $\mathcal{M}_2$ by non-increasing duration of their I/O task. Then Johnson's algorithm executes first all jobs in $\mathcal{M}_1$ followed by all jobs in $\mathcal{M}_2$, starting each task as soon as possible. The first adaptation is straightforward: we execute tasks in the same order as in Johnson's algorithm, starting each one as soon as possible after already scheduled tasks and while respecting the unavailability intervals. The second adaptation is called backfilling [102], a technique that takes advantage of intervals of idleness. Then a new task, rather than being mandatorily scheduled *after* the completion of all already scheduled tasks, can be scheduled in an idleness interval if this does not delay the start of any already scheduled task.

8.3. Proposed Approach

We now present our framework that deeply integrates lossy compression with parallel I/O libraries, allowing for the overlap of compression and I/O with computation using our task-scheduling algorithms. Figure 23 shows the overview of our proposed framework. The goal is to dump the data generated by one iteration during the next iteration. The runtime design consists of three main components: fine-grained compression, compressed data buffer, and shared tree for Huffman coding.

Fine-Grained Compression. The data generated by HPC applications often consists of multiple data fields. For instance, data from a Nyx cosmological simulation may include density, temperature, and velocity information. While it may seem intuitive to compress each data field separately to achieve granularity in compression tasks for our task-scheduling algorithms, most scientific applications have only a few data fields (e.g., 6~12 data fields). This limits the number of compression tasks and I/O tasks, leading to low task-scheduling efficiency. To address this challenge, we propose slicing each data field into smaller data blocks and independently compressing each data block. Fine-grained compression may introduce two potential issues: (1) a degradation in compression ratio compared with compressing the data together and (2) a decrease in compression and I/O throughput. In practice, we use offline profiling to evaluate compression and I/O performance to identify the point at which compression and I/O throughput start to deteriorate with small data block sizes. To mitigate the impact on I/O throughput, we propose the use of a compressed data buffer. Moreover, to minimize overhead caused by degradation in compression throughput, we propose the use of a shared Huffman tree.

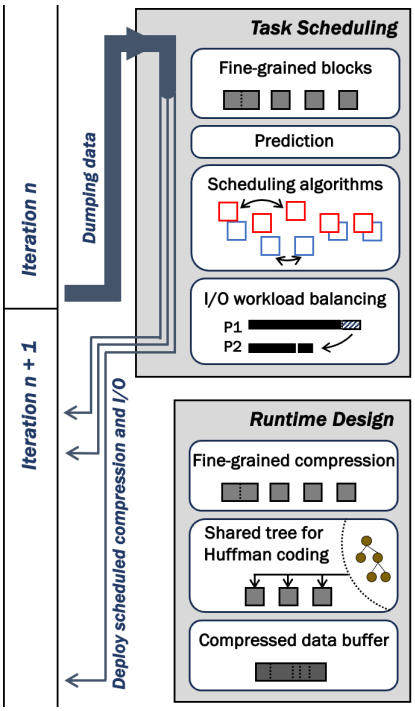


Figure 23: Overview of our proposed framework. We propose three runtime designs to facilitate task scheduling to handle the data-dumping process from iteration n and execute asynchronous compression, I/O operations, and computation tasks during iteration $n+1$.

Compressed Data Buffer. Based on our observations, the size of compressed data blocks can be relatively small. However, writing small data blocks can significantly reduce I/O throughput. To optimize I/O efficiency, we introduce an additional compressed data buffer that allows us to initiate I/O tasks as soon as possible while maintaining high throughput. The policy of writing the compressed data to the buffer and subsequently writing the buffer to the storage system is determined after the task scheduling. During the execution phase, once the execution orders for compression and I/O tasks are determined, we start placing the compressed data into the buffer when the background thread is engaged with I/O tasks or core tasks.

Huffman Coding with Shared Tree. During the prediction-based lossy compression process, one of the crucial steps is Huffman encoding of the quantization codes after the prediction and quantization steps. When compressing small data blocks with high compression ratios, building the Huffman tree can become a bottleneck for compression throughput. The reason is that building the Huffman tree takes nearly constant time regardless of the size of the input data. Additionally, data of similar types often results in similar Huffman trees: the Huffman tree built from the data of one iteration is highly similar to the tree built from the data of the next iteration.

To improve compression throughput for small data blocks, we propose using shared Huffman trees across different iterations and data blocks on the same process. Prediction-based lossy compression accommodates outliers, which allow us to include values that defy coding by this shared Huffman tree. Building and using a shared Huffman tree for each process based on the data of the current iteration are impractical, because doing so would require synchronization of all compression tasks before any I/O task for compressed data could proceed. Instead, we build the shared Huffman tree based on the quantization code from the previous one or few iterations and utilize it for the data of the current iteration.

8.4. Performance Evaluation

System configuration. We implement our approach using HDF5 [25] and SZ3 [101]. Experiments are conducted on the Summit supercomputer [44] at Oak Ridge National Laboratory. We use 16 nodes, each including two IBM POWER9 processors with 42 physical cores and 512 GB DDR4 memory and 64 GPUs.

Studied applications. We consider two I/O-intensive scientific applications, Nyx [10] and WarpX [45], which have been used in numerous previous I/O studies. Nyx is an adaptive mesh hydrodynamics code designed to model astrophysical reacting flows on HPC systems [11, 10]. Nyx uses MPI for the long-range force calculation and architecture-specific programming language for the short-range force algorithms, such as OpenMP and CUDA. According to prior studies [10, 61, 131], it can run on tens of thousands of GPUs. WarpX is a highly parallel and highly optimized code that utilizes AMReX [160], runs on GPUs and multicore CPUs, and features load-balancing capabilities. WarpX can scale up to the world's largest supercomputer and was the recipient of the 2022 ACM Gordon Bell Prize [116].

Compression Configuration. We evaluate our approach using different scales of Nyx and WarpX applications. In all our evaluations, we utilize both GPUs and CPUs. While GPUs serve as the primary compute unit, CPUs handle compression and I/O tasks. Based on previous work [78, 80], we use absolute error bounds of $(0.2, 0.4, 1e+3, 2e+5, 2e+5, 2e+5)$ to compress the six Nyx data fields (baryon density, dark matter density, temperature, velocity x, velocity y, velocity z), respectively, to achieve an average PSNR of 78.6 dB, resulting in a compression ratio of approximately 16×. The problem size of the Nyx application used in our evaluation is $4096 \times 4096 \times 4096$ with three additional fields: particle_vx, particle_vy, and particle_vz. We compress these additional fields with a compression ratio of 16x to ensure the post hoc analysis quality. For the WarpX application, we compress the data fields with a compression ratio of 273.9×, as suggested by the application developers based on their post hoc analysis.

Real-System-Based Evaluation. We compare our solution to the baseline and to the previous solution that uses only asynchronous I/O without compression or task scheduling techniques. We report the *overhead* per iteration, that is, the total execution time of an iteration minus the time required to perform

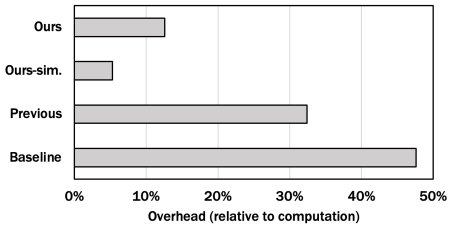
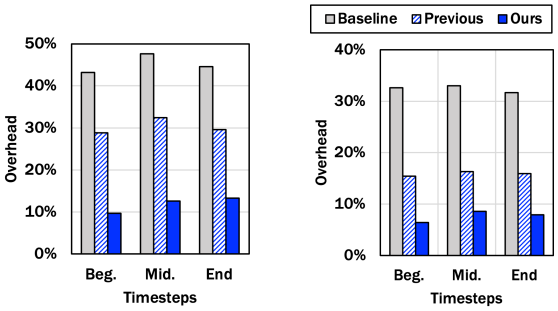


Figure 24: Time overheads (compared with computation time) of the baseline, asynchronous I/O, and our solution (with simulation for reference) with Nyx using 16 nodes and 64 GPUs



(a) Nyx (b) WarpX

Figure 25: Time overheads (compared with computation time) between the baseline, asynchronous I/O, and our solution across different timesteps.

only computations. Figure 24 shows the overall performance improvement of our solution in comparison with the baseline and previous solutions on the Nyx application. Additionally, we provide simulation results for reference. We note that the real implementation results in slightly larger overhead compared with the simulation results. The reason is that the real implementation encounters more unexpected task interference due to (1) uncertainty of computing application intervals and (2) inaccurately predicted compression ratio, compression time, and/or I/O time for compressed data. Nonetheless, our solution still achieves a significant performance improvement, with a 3.78× and 2.57× improvement over the baseline and previous solutions, respectively. Figure 25 presents the performance of our solution across different stages of the application: the beginning, middle, and end. Our solution consistently outperforms the previous solution across all stages of the application.

8.5. Conclusion and Future Work

Lossy compression and asynchronous I/O are two efficient solutions for reducing storage overhead and improving I/O performance in large-scale HPC/scientific applications. However, existing implementations have limitations that hinder the full utilization of lossy compression and can lead to task collisions, limiting the overall application performance. To address these challenges, we proposed an optimization approach for the task-scheduling problem involving application computation, compression, and I/O. Experimental results with up to 64 GPUs on Summit demonstrate that our solution reduces I/O overhead by up to 3.8× and 2.6× compared with the non-compression and asynchronous I/O solutions, respectively. Note that although we focus on parallel write to a single file, our proposed solutions are versatile and equally applicable to writing data to multiple files, enhancing performance in each scenario. Future work will need to consider other parallel I/O libraries.

9. Open problems

While the interest in lossy compression of scientific data started 7 to 8 years ago, the topic is still continuously developing, with more application domains, technique diversity, use cases, and users. We are not observing any slowdown in the growth of scientific data production from instrument facilities and simulation. On the contrary, the emergence of AI and large transformer models as new tools to advance science demands the production of even more data. Training a trillion-parameter transformer requires tens if not hundreds of trillion tokens. Generating this volume of data will take time, but it will also require significant storage space. Unfortunately, the storage, communication, and computing capabilities are not improving at the same pace. Scientific data compression will be even more needed in the future than it is today.

Beyond the need for scientific data compression, users also demand better lossy compression schemes that are faster, preserving more complex features and reducing the data further. Improving lossy compression after 7–8 years of dramatic progress is not trivial, and we summarize below some of the research topics that we consider important for this domain to continue its impressive improvement.

Lossy compressibility bound A fundamental driver for the research of compression has been the definition of compressibility limits expressed by compression bounds. The definition of Shannon entropy led to the development of a large research effort that produced optimal coding algorithms (Huffman and arithmetic coding). Such bounds are necessary to understand the effectiveness of existing algorithms and to motivate new research. However, despite years of research in information theory and improvements in the Shannon rate-distortion theory, there is no defined theoretical bound of lossy compressibility for scientific data. Establishing methods and techniques to compute such bounds (considering dataset properties and user quality constraints) is critical because it gives compressor developers an idea about the capability of existing compressors and informs users about the feasibility of using lossy compression for their use cases. This would serve all use cases of lossy compression for scientific data.

Compression of communications and memory accesses For many applications, including numerical simulation and AI model training, current parallel systems cannot reach high computing efficiency because of communication and memory bandwidth bottlenecks, despite decades of efforts in developing algorithms reducing communications and memory accesses. Research in mixed-precision computation has shown that for some applications the reduction of the numerical representation (e.g.,

from double precision to simple precision or from simple precision to half precision) still produces correct results while opening opportunities to improve performance by reducing the required communication and memory bandwidth. This is the main motivation for the current research on compressing MPI communications [167, 70] and data between the processors (CPU, GPU, accelerators) and memory. One of the main critical applications of this research is accelerating AI training of very large models by compressing all reduce/all-to-all communications [166]. Tests have also been performed on a wide range of numerical simulations: heat propagation, lattice Boltzmann simulation of airflow and fluid flow, simulation of the two-particle orbit problem, financial stock option price forecasting model, and weather forecasting models [42].

Homomorphic compression In simple terms, a homomorphism is a function that preserves the properties of mathematical structures (group, ring, or vector space) while mapping elements from one structure to another. For example, a ring homomorphism preserves addition, multiplication, and distributivity. This property is critical in the context of encryption where the goal is to perform operations on encrypted data without decrypting it. Homomorphic compression aims at a similar goal: being able to compute operations (e.g., algebraic) directly on the compressed data version without decompressing it first. Homomorphic compression could reduce dramatically the decompression/compression overhead in the case where many operations must be performed on the compressed data, for example, if a numerical simulation needs to compute on lossy compressed data because the non-compressed version of the data would not fit in memory. The community has started working on this nontrivial problem [111]. To be useful, this approach requires the co-design of compression scheme(s) with the operations to perform on compressed data because it must retain all the qualities of lossy compression: achieve desired ratios, speed, and accuracy (feature preservation).

Compression for AI in science An important direction to continue and potentially accelerate scientific discovery is to include artificial intelligence (AI) models in the scientific workflow. AI can be used for many use cases including surrogate models approximating numerical simulations, automatic online experiment steering, protein folding, and self-driving laboratories, and more generally as a researcher assistant. Current results show that larger models, in particular in the large language model (LLM) case, are more accurate, leading to a dramatic inflation of the AI model size. The most advanced LLMs use on the order of a trillion parameters. The largest models are so large that their parallel training takes several weeks or months on extreme-scale parallel systems. The most advanced training techniques combine multiple forms of parallelism including data, model, and pipeline parallelisms leading to data replication (data parallelism) and extreme volumes of internode communications (model and pipeline parallelism). The training duration far exceeds the mean time between failure of the parallel systems. In addition, large model training often fails, showing spikes of inaccuracy and divergence after convergence. The current method to mitigate system and training failures is checkpointing. Checkpoints can be very large, aggregating model weights [76], gradients, and potentially other elements such as the activation [79] and learning rate. Lossy compression can play an important role in accelerating large AI model training by reducing training set size, communication overhead [166], and checkpointing time.

Combining lossy compression and encryption Lossy compression and encryption are two forms of data transformations that are used for different applications. New use cases need both lossy data compression and encryption. These two techniques transform the data in fundamentally different ways: lossy compression aims at reducing the data as much as possible, keeping only the information necessary to produce correct results (from the user's perspective), while encryption aims at protecting the data against attacks, making it extremely difficult to decipher. Because lossy compression aims at reducing as much as possible redundancy in the compressed format of the data, it also makes it more fragile to attacks. Encryption can protect the data, but it can be slow and reduce the performance benefit of lossy compression. This situation raises the research question of how to best combine them for different use cases, considering the criteria of performance, protection against attack, and compressed version size. New results in this domain will find applications in federated learning, the secured storage of scientific data, and data communications in Integrated Research Infrastructures [113].

10. Conclusion: General observations from eight years of lossy compression for science.

Lossy compression for scientific data is in rapid expansion as an increasing number of scientific domains and modi operandi (observation, experimentation, simulation) require reducing data while preserving the same opportunity for scientific discovery. From the diversity of the research work in lossy compression for scientific data done in the joint laboratory of extreme-scale computing, and discussions with scientific users in Climate, Fluid Dynamics, Molecular dynamics, Cosmology, Fusion, Instrument, Numerical methods, and AI, we can highlight the following general observations:

- Scientific researchers are initially reluctant to reduce their data sets using lossy compressors while they may already use other methods such as sampling, decimation, resolution reduction, dimensionality reduction, and filtering. When resource limitations (storage footprint, memory occupation, network bandwidth) mandate the use of data reduction techniques, lossy compression, and in particular error-bounded lossy compression, becomes a competitive data reduction method often outperforming other methods regarding the trade-off between reduction speed, ratio, and preservation of quantities of interest.
- Scientific users trust lossy compression techniques only if the compression technique provides some guarantees of scientific integrity preservation. In other words, users require the same scientific outcomes from non-compressed and lossy compressed data. Often, formal guarantees of

scientific information preservation cannot be obtained. In many cases, only empirical analysis that compares analysis results from decompressed and uncompressed data is possible.

- As more scientific users adopt lossy compression for their data, we are observing a sharp increase in the application of lossy compression to more scientific domains and more use cases.
- A very important part of the methodology in this domain is missing: there is currently no method to compute an upper bound of lossy compressibility for any given datasets and user-specified distortion (loss) specifications. This lack should be addressed to provide lossy compressor developers with solid algorithmic objectives and users with well-informed expectations.
- Most scientific users need to preserve their observation/simulation/experimental data to be able to perform analysis after data generation. This is important to rerun the analysis to confirm a result for example. In some cases, new analyses, not planned or even not known at the time of generation will be run on the generated data. Lossy compression may not enable such unplanned analysis as the loss introduced by the compressor at the time of data generation was defined from the planned analysis.
- The paper has shown a panel of successes accomplished over the last eight years. As discussed in Section 4, the use of compressors within the core of numerical methods has also shown tremendous potential. That latter approach remains to be developed and assessed on additional algorithms.

11. Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709/2303064, OAC-2104023/2247080, OAC-2311875/2311876/2311877, OAC-2312673, and OAC-2034169. We acknowledge the computing resources provided on Bebop (operated by the Laboratory Computing Resource Center at Argonne). Some of the experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr). TEZip - This work has been supported by the COE research grant in computational science from Hyogo Prefecture and Kobe City through the Foundation for Computational Science. XIOS-SZ - Mario Acosta and Xavier Yepes-Arbós have received co-funding from the State Research Agency through OEMES (PID2020-116324RA-I00).

References

[1] Exascale computing project. `https://www.exascaleproject.org/about/`. (Accessed on 11/24/2023).
[2] *SIGGRAPH Comput. Graph.*, 21(6), 1987.
[3] A year of weather. 2019.
[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
[5] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, J. Loe, P. Luszczek, S. Pranesh, S. Rajamanickam, T. Ribizel, B. F. Smith, K. Swirydowicz, S. Thomas, S. Tomov, Y. M. Tsai, and U. M. Yang. A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. *The International Journal of High Performance Computing Applications*, 35(4):344–369, 2021.
[6] E. Agullo, F. Cappello, S. Di, L. Giraud, X. Liang, and N. Schenkels. Exploring variable accuracy storage through lossy compression techniques in numerical linear algebra: a first application to flexible GMRES. Research Report RR-9342, Inria Bordeaux Sud-Ouest, May 2020.
[7] E. Agullo, O. Coulaud, L. Giraud, M. Iannacito, G. Marait, and N. Schenkels. The backward stable variants of GMRES in variable accuracy. Research Report RR-9483, Inria, Sept. 2022.
[8] M. M. Alam, T. D. Nguyen, M. T. Hagan, and D. M. Chandler. A perceptual quantization strategy for HEVC based on a convolutional neural network trained on natural images. In *Applications of Digital Image Processing XXXVIII*, volume 9599, page 959918. International Society for Optics and Photonics, 2015.
[9] J. I. Aliaga, H. Anzt, T. Grützmacher, E. S. Quintana-Ortí, and A. E. Tomás. Compressed basis GMRES on high-performance graphics processing units. *The International Journal of High Performance Computing Applications*, pages 1–18, Aug. 2022.
[10] A. Almgren, V. Beckner, C. Daley, B. Friesen, Z. Lukic, A. Myers, J. Sexton, and W. Zhang. Nyx, 2023. `https://github.com/AMReX-Astro/Nyx`.
[11] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel. Nyx: A massively parallel AMR code for computational cosmology. *The Astrophysical Journal*, 765(1):39, 2013.
[12] J. Bachrach, H. Vo, B. Richards, and Y. L. D. a. d. . Design. Chisel: constructing hardware in a Scala embedded language. *DAC Design Automation Conference*, pages 1212–1221, 2012.
[13] A. H. Baker. On preserving scientific integrity for climate model data in the HPC era. *Computing in Science and Engineering*, 23(6):16–24, 2021.
[14] A. H. Baker, D. Hammerling, and T. L. Turton. Evaluating image quality measures to assess the impact of lossy data compression applied to climate simulation data. *Eurographics Conference on Visualization (EuroVis)*, pages 517–528, 2019.
[15] A. H. Baker, D. M. Hammerling, S. A. Mickelson, H. Xu, M. B. Stolpe, P. Naveau, B. Sanderson, I. Ebert-Uphoff, S. Samarasinghe, F. De Simone, F. Carbone, C. N. Gencarelli, J. M. Dennis, J. E. Kay, and P. Lindstrom. Evaluating lossy data compression on climate simulation data within a large ensemble. *Geoscientific Model Development*, 9(12):4381–4403.
[16] A. H. Baker, A. Pinard, and D. M. Hammerling. On a structural similarity index approach for floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–13, 2023.
[17] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd International Symposium on High-Performance*

Parallel and Distributed Computing, HPDC '14, page 203–214, New York, NY, USA, 2014. Association for Computing Machinery.

[18] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. TTHRESH: Tensor compression for multidimensional visual data. *IEEE transactions on visualization and computer graphics*, 26(9):2891–2903, 2019.

[19] S. Barros, D. Dent, L. Isaksen, G. Robinson, G. Mozdzynski, and F. Wollenweber. The IFS model: A parallel production weather code. *Parallel Computing*, 21(10):1621–1638, 1995.

[20] J. F. Bell III, A. Godber, S. McNair, M. A. Caplinger, J. N. Maki, M. T. Lemmon, J. Van Beek, M. C. Malin, D. Wellington, K. M. Kinch, M. B. Madsen, C. Hardgrove, M. A. Ravine, E. Jensen, D. Harker, R. B. Anderson, K. E. Herkenhoff, R. V. Morris, E. Cisneros, and R. G. Deen. The Mars science laboratory Curiosity rover Mastcam instruments: Preflight and in-flight calibration, validation, and data archiving. *Earth and Space Science*, 4(7):396–452, 2017.

[21] R. Bird, N. Tan, S. V. Luedtke, S. L. Harrell, M. Taufer, and B. Albright. VPIC 2.0: Next generation particle-in-cell simulations. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):952–963, 2021.

[22] J.-L. Blanco-Claraco, F.-A. Moreno-Dueñas, and J. González-Jiménez. The Malaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. *The International Journal of Robotics Research*, 33(2):207–214, 2014.

[23] J. Breit, G. Schmidt, and V. A. Strusevich. Non-preemptive two-machine open shop scheduling with non-availability constraints. *Mathematical Methods of Operations Research*, 57:217—-234, 2003.

[24] M. Burtscher and P. Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Transactions on Computers*, 58(1):18–31, Jan 2009.

[25] S. Byna, M. S. Breitenfeld, B. Dong, Q. Koziol, E. Pourmal, D. Robinson, J. Soumagne, H. Tang, V. Vishwanath, and R. Warren. ExaHDF5: delivering efficient parallel i/o on exascale computing systems. *Journal of Computer Science and Technology*, 35(1):145–160, 2020.

[26] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications*, 33(6):1201–1220, 2019.

[27] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[28] Z. Chen, T. He, X. Jin, and F. Wu. Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[29] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM, 2008.

[30] H. Cui, H. Zhang, G. R. Ganger, P. B. Gibbons, and E. P. Xing. Geeps: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 4. ACM, 2016.

[31] W. Cui, T. Zhang, S. Zhang, F. Jiang, W. Zuo, Z. Wan, and D. Zhao. Convolutional neural networks based intra prediction for HEVC. In *2017 Data Compression Conference (DCC)*, pages 436–436, 2017.

[32] cuZFP. `https://github.com/LLNL/zfp/tree/develop/src/cuda_zfp`, 2019. Online.

[33] Y. Dai, D. Liu, and F. Wu. A convolutional neural network approach for post-processing in HEVC intra coding. In *International Conference on Multimedia Modeling*, pages 28–39, 2017.

[34] X. Delaunay, A. Courtois, and F. Gouillon. Evaluation of lossless and lossy algorithms for the compression of scientific datasets in NetCDF-4 or HDF5 formatted files. preprint, Numerical Methods, Nov. 2018.

[35] S. Di. H5Z-SZ, 2023. `https://github.com/disheng222/H5Z-SZ`.

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

[37] S. Di, D. Tao, X. Liang, and F. Cappello. Efficient lossy compression for scientific data based on pointwise relative error bound. *IEEE Transactions on Parallel and Distributed Systems*, 30(2):331–345, 2019.

[38] R. Döscher, M. Acosta, A. Alessandri, P. Anthoni, T. Arsouze, T. Bergman, R. Bernardello, S. Boussetta, L.-P. Caron, G. Carver, M. Castrillo, F. Catalano, I. Cvijanovic, P. Davini, E. Dekker, F. J. Doblas-Reyes, D. Docquier, P. Echevarria, U. Fladrich, R. Fuentes-Franco, M. Gröger, J. v. Hardenberg, J. Hieronymus, M. P. Karami, J.-P. Keskinen, T. Koenigk, R. Makkonen, F. Massonnet, M. Ménégoz, P. A. Miller, E. Moreno-Chamarro, L. Nieradzik, T. van Noije, P. Nolan, D. O'Donnell, P. Ollinaho, G. van den Oord, P. Ortega, O. T. Prims, A. Ramos, T. Reerink, C. Rousset, Y. Ruprich-Robert, P. Le Sager, T. Schmith, R. Schrödner, F. Serva, V. Sicardi, M. Sloth Madsen, B. Smith, T. Tian, E. Tourigny, P. Uotila, M. Vancoppenolle, S. Wang, D. Wårlind, U. Willén, K. Wyser, S. Yang, X. Yepes-Arbós, and Q. Zhang. The EC-Earth3 Earth system model for the Coupled Model Intercomparison Project 6. *Geoscientific Model Development*, 15(7):2973–3020, 2022.

[39] J. Drkosova, A. Greenbaum, M. Rozloznik, and Z. Strakoš. Numerical stability of GMRES. *BIT Numerical Mathematics*, 35(February 1994):309–330, 1995.

[40] ECMWF. Modelling and prediction.

[41] A. Eldstål-Ahrens, A. Arelakis, and I. Sourdis. L2c: Combining lossy and lossless compression on memory and i/o. *ACM Trans. Embed. Comput. Syst.*, 21(1), jan 2022.

[42] A. Eldstål-Ahrens and I. Sourdis. Memsz: Squeezing memory traffic with lossy compression. *ACM Trans. Archit. Code Optim.*, 17(4), nov 2020.

[43] R. D. Evans, L. Liu, and T. M. Aamodt. JPEG-ACT: Accelerating deep learning via transform-based lossy compression. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 860–873. IEEE, 2020.

[44] O. R. L. C. Facility. Summit supercomputer, 2023. `https://www.olcf.ornl.gov/summit/`.

[45] L. Fedeli, A. Huebl, F. Boillod-Cerneux, T. Clark, K. Gott, C. Hillairet, S. Jaure, A. Leblanc, R. Lehe, A. Myers, C. Piechurski, M. Sato, N. Zaim, W. Zhang, J. Vay, and H. Vincenti. Pushing the frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, Los Alamitos, CA, USA, Nov. 2022. IEEE Computer Society.

[46] L. Feng, X. Zhang, X. Zhang, S. Wang, R. Wang, and S. Ma. A dual-network based super-resolution for compressed high definition video. In *Pacific Rim Conference on Multimedia*, pages 600–610, 2018.

[47] J. L. Ferrer, M. Roth, and A. Antoniadis. Data compression for diffraction patterns. *Acta Crystallographica. Section D, Biological crystallography*, 54(Pt 2):184–199, March 1998.

[48] H. Filter. `https://docs.hdfgroup.org/hdf5/develop/_f_i_l_t_e_r.html`. Online.

[49] L. Floridi and M. Chiriatti. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[50] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases - AD '11*, pages 36–47, New York, New York, USA, 2011. ACM Press.

[51] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[52] V. Gligorov. Real-time data analysis at the LHC: present and future. In G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, editors, *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, volume 42 of *Proceedings of Machine Learning Research*, pages 1–18, Montreal, Canada, 13 Dec 2015. PMLR.

[53] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, pages 2214–2224, 2017.

[54] Q. Gong, J. Chen, B. Whitney, X. Liang, V. Reshniak, T. Banerjee, J. Lee, A. Rangarajan, L. Wan, N. Vidal, Q. Liu, A. Gainaru, N. Podhorszki, R. Archibald, S. Ranka, and S. Klasky. MGARD: A multigrid framework for high-performance, error-controlled data compression and refactoring. *SoftwareX*, 24:101590, 2023.

[55] Q. Gong, J. Chen, B. Whitney, X. Liang, V. Reshniak, T. Banerjee, J. Lee, A. Rangarajan, L. Wan, N. Vidal, Q. Liu, A. Gainaru, N. Podhorszki, R. Archibald, S. Ranka, and S. Klasky. Mgard: A multigrid framework for high-performance, error-controlled data compression and refactoring. *SoftwareX*, 24:101590, 2023.

[56] Q. Gong, C. Zhang, X. Liang, V. Reshniak, J. Chen, A. Rangarajan,

S. Ranka, N. Vidal, L. Wan, P. Ullrich, N. Podhorszki, R. Jacob, and S. Klasky. Spatiotemporally adaptive compression for scientific dataset with feature preservation — a case study on simulation data with extreme climate events analysis. In *2023 IEEE 19th International Conference on e-Science (e-Science)*, pages 1–10, 2023.

[57] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[58] M. H. Gross, L. Lippert, and O. G. Staadt. Compression methods for visualization. *Future Generation Computer Systems*, 15(1):11–29, 1999.

[59] T. Grützmacher, H. Anzt, and E. S. Quintana-Ortí. Using Ginkgo's memory accessor for improving the accuracy of memory-bound low precision BLAS. *Software - Practice and Experience*, (September):1–18, 2021.

[60] J. Guivant, J. Nieto, and E. Nebot. Victoria park dataset, 2012.

[61] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka, V. Venkatram, L. Zarija, S. Saba, and W.-k. Liao. HACC: Simulating sky surveys on state-of-the-art supercomputing architectures. *New Astronomy*, 42:49–65, 2016.

[62] M. Hadian-Jazi, A. Sadri, A. Barty, O. Yefanov, M. Galchenkova, D. Oberthuer, D. Komadina, W. Brehm, H. Kirkwood, G. Mills, R. de Wijn, R. Letrun, M. Kloos, M. Vakili, L. Gelisio, C. Darmanin, A. P. Mancuso, H. N. Chapman, and B. Abbey. Data reduction for serial crystallography using a robust peak finder. *Journal of Applied Crystallography*, 54(5):1360–1378, Oct. 2021.

[63] M. Hammer, K. Yoshii, and A. Miceli. Strategies for on-chip digital data compression for x-ray pixel detectors. *Journal of Instrumentation*, 16(01):P01025, 2021.

[64] D. M. Hammerling, A. H. Baker, A. Pinard, and P. Lindstrom. A collaborative effort to improve lossy compression methods for climate data. In *2019 IEEE/ACM 5th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-5)*, pages 16–22, 2019.

[65] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[66] M. Hanke, J. Biercamp, C. O. Escamilla, T. Jahns, D. Kleberg, P. Selwood, and S. Mullerworth. Deliverable 7.3 – Reference implementations of parallel I/O and of I/O server. Technical report, DKRZ, 2013.

[67] HDF Group and others. Hierarchical data format version 5, filter, 2000.

[68] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[69] N. J. Higham and T. Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, May 2022.

[70] J. Huang, S. Di, X. Yu, Y. Zhai, J. Liu, K. Raffenetti, H. Zhou, K. Zhao, Z. Chen, F. Cappello, Y. Guo, and R. Thakur. C-Coll: Introducing error-bounded lossy compression into MPI collectives, 2023.

[71] N. Hübbe, A. Wegener, J. M. Kunkel, Y. Ling, and T. Ludwig. Evaluating lossy compression on climate data. In J. M. Kunkel, T. Ludwig, and H. W. Meuer, editors, *Supercomputing*, pages 343–356, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[72] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay, et al. The Community Earth System Model: a framework for collaborative research. *Bulletin of the American Meteorological Society*, 94(9):1339–1360, 2013.

[73] T. Ishikawa et al. Spring-8-ii conceptual design report. Technical report, RIKEN SPring-8 Center, Hyogo, Japan, 2014.

[74] C. Jia, S. Wang, X. Zhang, S. Wang, and S. Ma. Spatial-temporal residue network based in-loop filter for video coding. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, 2017.

[75] P. Jiao, S. Di, H. Guo, K. Zhao, J. Tian, D. Tao, X. Liang, and F. Cappello. Toward quantity-of-interest preserving lossy compression for scientific data. *Proc. VLDB Endow.*, 16(4):697—710, dec 2022.

[76] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello. DeepSZ: A novel framework to compress deep neural networks by using error-bounded lossy compression. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 159–170, 2019.

[77] S. Jin, S. Di, F. Vivien, D. Wang, Y. Robert, D. Tao, and F. Cappello. Concealing compression-accelerated I/O for HPC applications through in situ task scheduling. In *EuroSys 2024*, Athens, Greece, 2024.

[78] S. Jin, P. Grosset, C. M. Biwer, J. Pulido, J. Tian, D. Tao, and J. Ahrens. Understanding GPU-based lossy compression for extreme-scale cosmological simulations. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 105–115. IEEE, 2020.

[79] S. Jin, G. Li, S. L. Song, and D. Tao. A novel memory-efficient deep learning training framework via error-bounded lossy compression, 2020.

[80] S. Jin, J. Pulido, P. Grosset, J. Tian, D. Tao, and J. Ahrens. Adaptive configuration of in situ lossy compression for cosmology simulations via fine-grained rate-quality modeling. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, pages 45–56, 2020.

[81] S. Jin, D. Tao, H. Tang, S. Di, S. Byna, Z. Lukic, and F. Cappello. Accelerating parallel write via deeply integrating predictive lossy compression with hdf5. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2022.

[82] S. Jin, C. Zhang, X. Jiang, Y. Feng, H. Guan, G. Li, S. L. Song, and D. Tao. Comet: a novel memory-efficient deep learning training framework by using error-bounded lossy compression. *arXiv preprint arXiv:2111.09562*, 2021.

[83] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.

[84] S. Joussaume, A. Bellucci, J. Biercamp, R. Budich, A. Dawson, M. Foujols, B. Lawrence, L. Linardikis, S. Masson, Y. Meurdesoif, G. Riley, K. Taylor, and P. Vidale. Modelling the Earth's climate system: data and computing challenges. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 2325–2356. IEEE, nov 2012.

[85] J. E. Kay and et al. The Community Earth System Model (CESM) large ensemble project: A community resource for studying climate change in the presence of internal climate variability. *Bulletin of the American Meteorological Society*, 96(8):1333–1349, 2015.

[86] S. Kayum et al. GeoDRIVE – a high performance computing flexible platform for seismic applications. *First Break*, 38(2):97–100, 2020.

[87] A. Kiely and M. Klimesh. Preliminary image compression results from the mars exploration rovers. *Interplanetary Network Progress Report*, pages 1–8, 02 2004.

[88] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[89] M. Kuhn, J. Kunkel, and T. Ludwig. Data compression for climate data. *Supercomputing Frontiers and Innovations*, 3(1):75–94, June 2016.

[90] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Ku, C. Chang, S. Klasky, R. Latham, R. B. Ross, and N. F. Samatova. ISABELA for effective in situ compression of scientific data. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, 2013.

[91] Large Scale Visual Recognition Challenge. `http://www.image-net.org/challenges/LSVRC/`, 2019. Online.

[92] C.-Y. Lee. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters*, 20(3):129–139, 1997.

[93] F. Leonarski, A. Mozzanica, M. Brückner, C. Lopez-Cuenca, S. Redford, L. Sala, A. Babic, H. Billich, O. Bunk, B. Schmitt, and M. Wang. JUNGFRAU detector for brighter X-ray sources: Solutions for IT and data science challenges in macromolecular crystallography. *Structural Dynamics*, 7(1):014305, 2020.

[94] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao. Fully connected network-based intra prediction for image coding. *IEEE Transactions on Image Processing*, 27(7):3236–3247, 2018.

[95] S. Li, P. Lindstrom, and J. Clyne. Lossy scientific data compression with SPERR. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1007–1017, 2023.

[96] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs. Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum*, 37(6):422–447, 2018.

[97] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink. A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, (9):N31, 2008.

[98] 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.

[99] 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 *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 81–90, 2020.

[100] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. R. Pugmire, M. Wolf, N. Podhorszki, et al. MGARD+: optimizing multilevel methods for error-bounded scientific data reduction. *IEEE Transactions on Computers*, 2021.

[101] 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.

[102] D. A. Lifka. The ANL/IBM SP scheduling system. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 295–303, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

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

[104] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

[105] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

[106] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello. Dynamic quality metric oriented error bounded lossy compression for scientific datasets. In *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 892–906. IEEE Computer Society, 2022.

[107] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello. FAZ: A flexible auto-tuned modular error-bounded compression framework for scientific data. In *Proceedings of the 37th International Conference on Supercomputing*, pages 1–13, 2023.

[108] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[109] E. Maisonnave, I. Fast, T. Jahns, J. Biercamp, S. Sénési, Y. Meurdesoif, and U. Fladrich. CDI-pio & XIOS I/O servers compatibility with HR climate models. Technical report, CERFACS, 2017.

[110] J. N. Maki, J. F. Bell III, K. E. Herkenhoff, S. W. Squyres, A. Kiely, M. Klimesh, M. Schwochert, T. Litwin, R. Willson, A. Johnson, M. Maimone, E. Baumgartner, A. Collins, M. Wadsworth, S. T. Elliot, A. Dingizian, D. Brown, E. C. Hagerott, L. Scherr, R. Deen, D. Alexander, and J. Lorre. Mars exploration rover engineering cameras. *Journal of Geophysical Research: Planets*, 108(E12), 2003.

[111] M. Martel. Compressed matrix computations. In *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pages 68–76. IEEE, 2022.

[112] G. A. Meehl, R. Moss, K. E. Taylor, V. Eyring, R. J. Stouffer, S. Bony, and B. Stevens. Climate model intercomparisons: Preparing for the next phase. *Eos, Transactions American Geophysical Union*, 95(9):77–78, 2014.

[113] W. L. Miller, D. Bard, A. Boehnlein, K. Fagnan, C. Guok, E. Lançon, S. Ramprakash, M. Shankar, N. Schwarz, and B. L. Brown. Integrated research infrastructure architecture blueprint activity (final report 2023). 7 2023.

[114] Miranda Radiation Hydrodynamics Data. `https://wci.llnl.gov/simulation/computer-codes/miranda`, 2019. Online.

[115] NVIDIA. `https://developer.nvidia.com/nvcomp`. Online.

[116] Oak Ridge Leadership Computing Facility. WarpX, granted early access to the exascale supercomputer Frontier, receives the high-performance computing world's highest honor, 2023. `https://www.olcf.ornl.gov/2022/11/17/plasma-simulation-code-wins-2022-acm-gordon-bell-prize/`.

[117] A. Olmo, A. S. Zamzam, A. Glaws, and R. King. Physics-driven convolutional autoencoder approach for CFD data compressions: Preprint. Technical Report 1969095, U.S. Department of Energy Office of Scientific and Technical Information, 4 2023.

[118] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[119] C. C. Paige and Z. Strakoš. Residual and backward error bounds in minimum residual Krylov subspace methods. *SIAM Journal on Scientific Computing*, 23(6):1898–1923, January 2002.

[120] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[121] A. Poppick, J. Nardi, N. Feldman, A. H. Baker, A. Pinard, and D. M. Hammerling. A statistical analysis of lossily compressed climate model data. *Computers and Geosciences*, 145:104599, 2020.

[122] S. Puri, S. Lasserre, and P. L. Callet. CNN-based transform index prediction in multiple transforms framework to assist entropy coding. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 798–802, 2017.

[123] QMCPACK: many-body ab initio Quantum Monte Carlo code. `http://vis.computer.org/vis2004contest/data.html`, 2019. Online.

[124] M. Rahman, M. Islam, J. C. Calhoun, and M. Chowdhury. Dynamic error-bounded lossy compression (EBLC) to reduce the bandwidth requirement for real-time vision-based pedestrian safety applications. *arXiv preprint arXiv:2002.03742*, 2020.

[125] B. Reagan, U. Gupta, B. Adolf, M. Mitzenmacher, A. Rush, G.-Y. Wei, and D. Brooks. Weightless: Lossy weight encoding for deep neural network compression. In *International Conference on Machine Learning*, pages 4321–4330, 2018.

[126] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler. vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, page 18. IEEE Press, 2016.

[127] R. Roy. Compression of time evolutionary image data through predictive deep neural networks. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 41–50, 2021.

[128] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal Scientific Computing*, 14:461–469, 1993.

[129] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[130] E. R. Schendel, Y. Jin, N. Shah, J. Chen, C. Chang, S.-H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. ISOBAR preconditioner for effective and high-throughput lossless data compression. In *2012 IEEE 28th International Conference on Data Engineering*, pages 138–149, 2012.

[131] A. Siegel, E. Draeger, J. Deslippe, T. Evans, M. Francois, T. C. Germann, D. F. Martin, and W. Hart. Application results on early exascale hardware. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2022.

[132] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[133] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015.

[134] S. Strempfer, K. Yoshii, M. Hammer, D. Bycul, and A. Miceli. Designing a streaming data coalescing architecture for scientific detector ASICs with variable data velocity. In *2021 3rd Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (XLOOP)*, pages 8–14. IEEE, 2021.

[135] S. Strempfer, T. Zhou, K. Yoshii, M. Hammer, A. Babu, D. Bycul, J. Weizeorick, M. J. Cherukara, and A. Miceli. A lightweight, user-configurable detector ASIC digital architecture with on-chip data compression for MHz X-ray coherent diffraction imaging. *Journal of Instrumentation*, 17(10):P10042, 2022.

[136] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[137] I. Talukdar, A. Singh, R. Underwood, K. Sato, and W. Yu. Integrating tezip into libpressio: A case study of integrating a dynamic application into a static c environment. 2023.

[138] H. Tang, Q. Koziol, J. Ravi, and S. Byna. Transparent asynchronous parallel I/O using background threads. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):891–902, 2021.

[139] 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*, pages 1129–1139. IEEE, 2017.

[140] D. Tao, S. Di, H. Guo, Z. Chen, and F. Cappello. Z-checker: A framework for assessing lossy compression of scientific data. *Int. J. High Perform. Comput. Appl.*, 33(2):285–303, March 2019.

[141] D. S. Taubman, M. W. Marcellin, and M. Rabbani. JPEG2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286–287, 2002.

[142] The HDF Group. Hierarchical data format version 5. `http://www.hdfgroup.org/HDF`.

[143] J. Tian, S. Di, X. Yu, C. Rivera, K. Zhao, S. Jin, Y. Feng, X. Liang, D. Tao, and F. Cappello. Optimizing error-bounded lossy compression for scientific data on GPUs. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 283–293, 2021.

[144] J. Tian, S. Di, C. Zhang, X. Liang, S. Jin, D. Cheng, D. Tao, and F. Cappello. WaveSZ: A hardware-algorithm co-design of efficient lossy compression for scientific data. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP'20, page 74–88, New York, NY, USA, 2020. Association for Computing Machinery.

[145] J. Tian, S. Di, K. Zhao, C. Rivera, M. H. Fulp, R. Underwood, S. Jin, X. Liang, J. Calhoun, D. Tao, and F. Cappello. cusz: An efficient gpu-based error-bounded lossy compression framework for scientific data. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, PACT '20, page 3–15, New York, NY, USA, 2020. Association for Computing Machinery.

[146] S.-M. Tseng, B. Nicolae, F. Cappello, and A. Chandramowlishwaran. Demystifying asynchronous I/O Interference in HPC applications. *The International Journal of High Performance Computing Applications*, 35(4):391–412, 2021.

[147] T. Ueno, K. Sano, and S. Yamamoto. Bandwidth compression of floating-point numerical data streams for FPGA-based high-performance computing. *ACM Transactions on Reconfigurable Technology and Systems*, 10(3):1–22, July 2017.

[148] R. Underwood, J. Bessac, S. Di, and F. Cappello. Understanding the effects of modern compressors on the Community Earth Science Model. In *2022 IEEE/ACM 8th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD)*, pages 1–10, 2022.

[149] R. Underwood, V. Malvoso, J. C. Calhoun, S. Di, and F. Cappello. Productive and performant generic lossy data compression with libpressio. In *2021 7th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-7)*, pages 1–10. IEEE, 2021.

[150] R. Underwood, C. Yoon, A. Gok, S. Di, and F. Cappello. ROIBIN-SZ: Fast and science-preserving compression for serial crystallography, 2022.

[151] H. F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM Journal on Scientific Computing*, 9(1):152–163, 1988.

[152] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244. ACM, 2015.

[153] L. Wang, J. Ye, Y. Zhao, W. Wu, A. Li, S. L. Song, Z. Xu, and T. Kraska. Superneurons: dynamic GPU memory management for training deep neural networks. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 41–53. ACM, 2018.

[154] L. Yan, X. Liang, H. Guo, and B. Wang. TopoSZ: Preserving topology in error-bounded lossy compression. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–11, 2023.

[155] X. Yepes-Arbós, G. van den Oord, M. C. Acosta, and G. D. Carver. Evaluation and optimisation of the I/O scalability for the next generation of Earth system models: IFS CY43R3 and XIOS 2.0 integration as a case study. *Geoscientific Model Development*, 15(2):379–394, 2022.

[156] K. Yoshii. Streampressor: Stream compressor hardware generator. `https://github.com/hwspec/StreamPressor`, 2023.

[157] K. Yoshii, T. Ueno, K. Sano, A. Miceli, and F. Cappello. Streaming hardware compressor generator framework. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 289–297, 2023.

[158] X. Yu, S. Di, A. M. Gok, D. Tao, and F. Cappello. cuZ-Checker: A GPU-based ultra-fast assessment system for lossy compressions. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 307–319, 2021.

[159] C. S. Zender. Bit grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF operators (NCO, v4.4.8+). *Geoscientific Model Development*, 9(9):3199–3211, 2016.

[160] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, et al. AMReX: a framework for block-structured adaptive mesh refinement. *The Journal of Open Source Software*, 4(37):1370, 2019.

[161] 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)*, pages 1643–1654, 2021.

[162] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello. SDRBench: Scientific data reduction benchmark for lossy compressors. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2716–2724, Los Alamitos, CA, USA, dec 2020. IEEE Computer Society.

[163] K. Zhao, S. Di, D. Perez, X. Liang, Z. Chen, and F. Cappello. MDZ: An efficient error-bounded lossy compressor for molecular dynamics. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 27–40, 2022.

[164] L. Zhao, S. Wang, X. Zhang, S. Wang, S. Ma, and W. Gao. Enhanced CTU-level inter prediction with deep frame rate up-conversion for high efficiency video coding. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 206–210, 2018.

[165] H. Zheng, V. Vishwanath, Q. Koziol, H. Tang, J. Ravi, J. Mainzer, and S. Byna. HDF5 Cache VOL: Efficient and scalable parallel I/O through caching data on node-local storage. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 61–70. IEEE, 2022.

[166] Q. Zhou, Q. Anthony, L. Xu, A. Shafi, M. Abduljabbar, H. Subramoni, and D. K. D. Panda. Accelerating distributed deep learning training with compression assisted allgather and reduce-scatter communication. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 134–144, 2023.

[167] Q. Zhou, P. Kousha, Q. Anthony, K. Shafie Khorassani, A. Shafi, H. Subramoni, and D. K. Panda. Accelerating MPI all-to-all communication with online compression on modern GPU clusters. In A.-L. Varbanescu, A. Bhatele, P. Luszczek, and B. Marc, editors, *High Performance Computing*, pages 3–25, Cham, 2022. Springer International Publishing.

Author Biography

Franck Cappello is senior computer scientist and R&D lead at Argonne National laboratory. From 2016 and with the support of ECP (US Exascale Computing Project), he started exploring lossy compression for scientific computing. This research produced the SZ lossy compressor, the Libpressio unifying API, the Z-checker tool to assess the nature of lossy compression errors and the SDRbench repository of reference scientific datasets. Franck is IEEE fellow and recipient of the 2024 IEEE Charles Babbage Award, the 2024 Europar achievement award, the 2022 HPDC achievement award, the 2018 IEEE TCPP Outstanding Service Award, the 2021 IEEE Transactions in Computers Award for Editorial Service and Excellence and two prestigious R&D100 awards (2019 and 2021).

Frédéric Vivien graduated in Computer Sciences and received his PhD from École Normale Supérieure de Lyon, France, in 1997. From 1998 to 2002, he was an associate professor at the Louis Pasteur University in Strasbourg, France. He is currently an INRIA senior researcher at ENS Lyon, France. His main research interests include parallel computing, scheduling, and resilience techniques. He is the author of two books.

Kazutomo Yoshii is a principal experimental systems specialist at Argonne National Laboratory. He earned an M.S. in computer science from Toyohashi University of Technology, Japan, in 1994. His career began with developing medical imaging analysis software for functional MRI images at Hitachi's research facility in Japan. In 1998, he joined Turbolinux, contributing to the Linux operating system in Japan and Santa Fe, New Mexico. Later, in 2002, he focused on dynamic provisioning systems for cluster systems at Mountain View Data. Since 2004, he has been with Argonne National Laboratory, actively engaged in co-design activities for supercomputers and experimental systems. His recent work involves investigating on-chip processing digital logic for scientific detectors in collaboration with domain scientists at Argonne. His extensive research interests include power/thermal-aware computing, reconfigurable dataflow computing, edge computing, AI accelerators, and hardware specialization.

Xavier Yepes-Arbós is a research engineer of the Earth Sciences Department at the Barcelona Supercomputing Center (BSC) where he co-leads the High-Performance Computing for Earth Sciences Team. He holds a bachelor's degree in computer engineering with specialization in computer architecture and a master's degree in Innovation and Research in Informatics (MIRI) with specialization in HPC from the Universitat Politècnica de Catalunya (UPC). He joined BSC in 2015 to work on different research lines, including scalability and performance analysis of Earth system models (ESMs), optimization of the computational and I/O processes, data compression and explore novel hardware architectures.

Mario C. Acosta is a leader researcher the Computational Group of the Earth Sciences Department at the Barcelona Supercomputing Center. He received his PhD in Computer Science from University of Granada in 2015. He integrated the BSC eight years ago, assuming the leadership of the computational group one year ago. Dr Acosta's research lines are well embodied in those related to High Performance Computing (HPC) applied to Earth System Models He is the coordinator of the HorizonE ESiWACE3: Center of excellence for weather and climate phase 3 and WP leader in the development of the climate digital twin in Destination Earth.

Amarjit Singh is working as Postdoctoral researcher at High Performance Big Data Research Team in the Center for Computational Science at RIKEN (RIKEN R-CCS) . He received his Ph.D. in Computer Science

from IKG Punjab Technical University, INDIA in 2022 and M.Sc. from Guru Nanak Dev University, INDIA in 2008. His main research interests include high performance computing, data compression, deep learning.

Kento Sato is a team leader of High-Performance Big Data Research Team in the Center for Computational Science at RIKEN (RIKEN R-CCS). His research area is distributed systems and parallel computing, particularly in High Performance Computing (HPC). Major focuses of his research are artificial intelligence, machine learning and deep learning in HPC, application reproducibility (MPI reproducibility, and Validation), scalable fault tolerance (Scalable checkpoint/restart, Fault tolerant MPI, Resilient system design), and I/O optimization (NVRAM, Burst buffer, and Big data), co- designing and cloud computing. He received his Ph.D. in the Dept. of Mathematical & Computing Sciences at Tokyo Tech in 2014, his M.S. in the Dept. of Mathematical & Computing Sciences at Tokyo Tech in 2010, and his B.S. in the Dept. of Information Science at Tokyo Tech in 2008.

Dingwen Tao is an associate professor at Indiana University Bloomington, where he directs the High-Performance Data Analytics and Computing Lab. He received his Ph.D. in Computer Science from University of California, Riverside in 2018 and B.S. in Mathematics from University of Science and Technology of China in 2013. He is the recipient of various awards including NSF CAREER Award (2023), Amazon Research Award (2022), Meta Research Award (2022), R&D100 Awards Winner (2021), IEEE Computer Society TCHPC Early Career Researchers Award for Excellence in HPC (2020), NSF CRII Award (2020), and IEEE CLUSTER Best Paper Award (2018). He is serving as an Associate Editor of IEEE Transactions on Parallel and Distributed Systems.

Jiannan Tian is a PhD student in Intelligent Systems Engineering at Indiana University Bloomington. His research interests include scientific data compression, error analysis, and GPU-centric computing. His ongoing projects include developing GPU-accelerated compression algorithms and system design optimization for scientific compression. He has been in a long-term internship at Argonne National Laboratory.

Boyuan Zhang is a PhD student in Intelligent Systems Engineering at Indiana University Bloomington. His research interests encompass GPU computing, data compression, and quantum computing. He has completed a summer internship at Pacific Northwest National Laboratory in 2023.

Sheng Di (ANL) is a Computer Scientist in the Mathematics and Computer Science (MCS) division of Argonne National Laboratory. His current research interest includes lossy compression for scientific datasets, high performance computing, scalable computing, and fault tolerance. He is a senior member of IEEE, institute fellow of NAISE, also the scientist at Large through the Consortium for Advanced Science and Engineering (CASE) at the University of Chicago. He is the DOE 2021 Early Career Research Program Award Winner, and the recipient of 2018 IEEE-Chicago Distinguished Mentoring Award and 2019 IEEE-Chicago Distinguished R\&D Award. See https://www.mcs.anl.gov/~shdi for further information.

Sian Jin is an assistant professor at Temple University. He received his Ph.D. in Computer Engineering from Indiana University in 2023 and B.S. in Physics from Beijing Normal University in 2018. His research interests include high performance computing, data compression, neural networks, and parallel

computing. He has published several papers in major journals and international conferences including the SC, PPoPP, VLDB, EuroSys, HPDC, IPDPS, and ICDE. Email: sian.jin@temple.edu.

Robert Underwood is a Post-Doctoral Appointee in the Mathematics and Computer Science Division at Argonne National Laboratory focusing on using data compression to accelerate I/O for large-scale scientific applications including AI for Science. His library LibPressio, which allows users to experiment and adopt advanced compressors quickly, has over 200 average unique monthly downloads, is used in over 17 institutions worldwide, and is a contributor to the R&D100 winning SZ family of compressors and other compression libraries. He regularly mentors students and is the early career ambassador for Argonne to the Joint Laboratory for Extreme Scale Computing.

Tomohiro Ueno is currently a research scientist in the processor research team at the Riken Center for Computational Science. He received his master's and Ph.D. degrees from the graduate school of information sciences at Tohoku University in 2016. Since then, he has been a postdoctoral researcher at the Graduate School of Engineering, Tohoku University, as a member of the ImPACT project until 2017 and RIKEN Center for Computational Science until 2022. His research interests include hardware architectures, hardware algorithms, data compression, and communication networks for high-performance computing.

Hartwig Anzt is a Full Professor at the School of Computation, Information and Technology at the Technical University of Munich. He also holds a Research Professor Position at the University of Tennessee in Knoxville. He preciously served as the director of the University of Tennessee's Innovative Computing Lab after holding a Junior Professor at the Karlsruhe Institute of Technology. Hartwig Anzt's research focuses on iterative methods and preconditioning techniques for the next generation hardware architectures. He also has a long track record of high-quality development. He is author of the MAGMA-sparse open-source software package and managing lead of the Ginkgo math software library. He also is a PI in the EuroHPC Project MICROCARD.

Thomas Grützmacher is a Ph.D. researcher at the Karlsruhe Institute of Technology. He received a bachelor's degree from the KIT in 2015 with an emphasis on software development for mobile computing and IoT. In 2018 he completed his Master's studies with an emphasis on High-Performance Computing and unconventional precision formats. Thomas Grützmacher's research focus is on designing a modular precision ecosystem. He also is among the core developer team of the Ginkgo open-source library.

Emmanuel Agullo is a researcher at Inria in the Concace joint project with Airbus CR&T and Cerfacs. His main research interests are parallel computing, parallel task programming, parallel composability and numerical linear algebra.

Luc Giraud is a researcher at Inria in the Concace joint project with Airbus CR&T and Cerfacs. His main research interests are in paralel scientific computing and numerical linear algebra for large scale applications.

photos

Luc Giraud, Emmanuel Agullo,

Franck Cappello 

Sheng Di 

Robert Underwood 

Frederic Vivien 

Kazutomo Yoshii 

Xavier Yepes 

Mario Acosta 

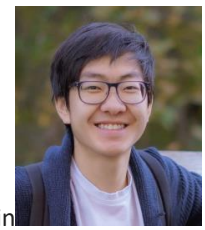
Amarjit Singh 

Kento Sato 

Dingwen Tao 

Boyuan Zhang 

Jiannan Tian 

Sian Jin 

Tomohiro Ueno 

Hartwig Anzt 

Thomas Grützmache 

Luc Giraud 

Emmanuel Agullo

Franck Cappello
Argonne National Laboratory

Sheng Di
Argonne National Laboratory

Robert Underwood
Argonne National Laboratory

Dingwen Tao
Indiana University Bloomington Jon Calhoun Clemson University Yoshii

Kazutomo Argonne
National Laboratory

Kento Sato
RIKEN Center for Computational Science

Amarjit Singh
RIKEN Center for Computational Science

Luc Giraud
National Research Institute for Computing and Automation

Emmanuel Agullo
National Research Institute for Computing and Automation

Xavier Yepes
Barcelona Supercomputing Center Mario

Acosta Barcelona Supercomputing Center

Sian Jin
Indiana University Bloomington

Jiannan Tian
Indiana University Bloomington

Frédéric Vivien
National Research Institute for Computing and Automation

Boyuan Zhang
Indiana University Bloomington Kentaro Sano RIKEN Center for Computational Science

Tomohiro Ueno
RIKEN Center for Computational Science

Thomas Gruetzmacher
Karlsruhe Institute of Technology

Hartwig Anzt
Karlsruhe Institute of Technology

The Joint Laboratory on Extreme Scale Computing (JLESC) was initiated at the same time lossy compression for scientific data became an important topic for the scientific communities. The teams involved in the JLESC played and are still playing an important role in developing the research, techniques, methods, and technologies making lossy compression for scientific data a key tool for scientists and engineers. In this paper we present the evolution of lossy compression for scientific data from 2015, describing the situation before the JLESC started, the evolution of this discipline in the past 8 years (until 2023) through the prism of the JLESC collaborations on this topic, and some of the remaining open research questions.

Franck Cappello, lead writer, contributed to intro, open questions.
Mario Acosta, contributed to the section on compression for climate simulation.
Emmanuel Agullo, contributed to the section on compression for linear algebra.
Hartwig Anzt, contributed to the section on compression for linear algebra.
Sheng Di, contributed to several sections.
Luc Giraud, contributed to the section on compression for linear algebra.
Thomas Grutzmacher, contributed to the section on compression for linear algebra.
Sian Jin, contributed to the section on compression for I/O.
Kento Sato, contributed to the section on compression for instruments.
Amarjit Singh, contributed to the section on compression for instruments.
Dingwen Tao, contributed to the section on compression for AI and GPU for compression.
Jiannan Tian, contributed to the section on compression for AI and GPU for compression.
Tomohiro Ueno, contributed to the section on compression for instruments.
Robert Underwood, contributed to the section on compression for instruments.
Frédéric Vivien, contributed to the section on compression for I/O.
Xavier Yepes-Arbós, contributed to the section on compression for climate simulation.
Kazutomo Yoshii, contributed to the section on compression for instruments.
Boyuan Zhang, contributed to the section on compression for AI and GPU for compression.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: