What to Support when You're Compressing: The State of Practice, Gaps, and Opportunities for Scientific Data Compression

Franck Cappello*† Argonne National Laboratory Lemont, IL USA

Robert Underwood*‡ Argonne National Laboratory Lemont, IL USA

Yuri Alexeev Argonne National Laboratory Lemont, IL USA

Allison Baker NSF National Center for Atmospheric Research Boulder CO, USA

Ebru Bozdağ Colorado School of Mines Golden, CO USA

Martin Burtscher Texas State University San Marcos TX, USA

Kyle Chard The University of Chicago Chicago, IL USA

Jackie Chen Sandia National Laboratories Albuquerque NM, USA

Sheng Di§ Argonne National Laboratory Lemont, IL USA

Kyle Gerard Felker Argonne National Laboratory Lemont, IL USA

Paul Christopher O'Grady Stanford University Stanford CA, USA

Hanqi Guo Ohio State University Columbus OH, USA

Yafan Huang University of Iowa Iowa City IA, USA

Sian Jin Temple University Philadelphia PA, USA

Peng Jiang University of Iowa Iowa City IA, USA

Petter Johansson KTH Department of Applied Physics Solna Sweden

Shaomeng Li¶ NSF National Center for Atmospheric Research Boulder CO, USA

Xin Liang University of Kentucky Lexington KY, USA

Erik Lindahl Stockholm University Stockholm Sweden

Peter Lindstrom Lawrence Livermore National Laboratory Livermore CA, USA

Zarija Lukić Lawrence Berkeley National Laboratory Alameda CA, USA

Magnus Lundborg KTH Department of Applied Physics Solna Sweden

Danylo Lykov Argonne National Laboratory Lemont, IL USA

Masaru Nagaso Colorado School of Mines Golden, CO USA RIKEN R-CCS Kobe, Japan

Kento Sato Amarjit Singh RIKEN R-CCS Kobe, Japan

Seung Woo Son UMass Lowell Lowell, MA USA

Shihui Song University of Iowa Iowa City IA, USA

William Tang Princeton Plasma Physics Laboratory Princeton NJ, USA

Dingwen Tao Jiannan Tian‖ Indiana University Bloomington Bloomington IN, USA

Kazutomo Yoshii Argonne National Laboratory Lemont, IL USA

Kai Zhao Florida State University Tallahassee FL, USA

*The co-first authors are listed in alphabetical order by last name. The remaining authors are listed in alphabetical order by last name. The corresponding authors are runderwood@anl.gov and cappello@anl.gov

†Also with The University of Chicago.

‡Also with The University of Chicago.

§Also with The University of Chicago.

¶Currently at NVIDIA

‖Currently at Oakland University

SC '25, November 16–21, 2025, St. Louis, MO

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/3712285.3759856

ABSTRACT

Over the last nearly 20 years, lossy compression has become an essential aspect of HPC applications' data pipelines, allowing them to overcome limitations in storage capacity and bandwidth and, in some cases, increase computational throughput and capacity. However, with the adoption of lossy compression comes the requirement to assess and control the impact lossy compression has on scientific outcomes. In this work, we take a major step forward in describing the state of practice and by characterizing workloads. We examine applications' needs and compressors' capabilities across 9 different supercomputing application domains. We present 24 takeaways that provide best practices for applications, operational impacts for facilities achieving compressed data, and gaps in application needs not addressed by production compressors that point towards opportunities for future compression research.

CCS CONCEPTS

• **Software and its engineering → Requirements analysis**; • **Theory of computation → Data compression**; • **Computing methodologies → Massively parallel and high-performance simulations**; • **General and reference** → *Surveys and overviews.*

KEYWORDS

Error-Bounded Lossy Compression, High-Performance Computing, I/O Optimization, Requirements Analysis, State of Practice

ACM Reference Format:

Franck Cappello, Robert Underwood, Yuri Alexeev, Allison Baker, Ebru Bozdağ, Martin Burtscher, Kyle Chard, Jackie Chen, Sheng Di, Kyle Gerard Felker, Paul Christopher O'Grady, Hanqi Guo, Yafan Huang, Sian Jin, Peng Jiang, Petter Johansson, Shaomeng Li, Xin Liang, Erik Lindahl, Peter Lindstrom, Zarija Lukić, Magnus Lundborg, Danylo Lykov, Masaru Nagaso, Kento Sato, Amarjit Singh, Seung Woo Son, Shihui Song, William Tang, Dingwen Tao, Jiannan Tian, Kazutomo Yoshii, and Kai Zhao. 2025. What to Support when You're Compressing: The State of Practice, Gaps, and Opportunities for Scientific Data Compression. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3712285.3759856

1 INTRODUCTION

Scientific simulations, experiments, and observations are producing increasing volumes of data due to the change of supercomputer generation (from petascale to exascale) and the update of large scientific instruments (accelerators, light sources, and telescopes). In many situations, the data produced is too large to be communicated on a network, stored in storage systems, and analyzed with user tools. The scientific community's response to this challenge is data reduction. Reduction can take many forms, such as compression, triggering, sampling/filtering, quantization, and dimensionality reduction. This report focuses on a specific technique: lossy compression. Compared with other data reduction techniques, lossy compression keeps all data points. It leverages the correlations between data points and the reduction of data point accuracy to reduce the scientific data. To preserve the same opportunities for scientific discoveries from lossy compressed data as from noncompressed data, compression techniques need to respect user quality constraints that generally concern the preservation of quantities of interest (QoIs) to a certain accuracy. In addition, to be useful, a lossy compression technique needs to satisfy user requirements in terms of compression ratio (by what factor the data has been reduced compared with the original version) and compression speed (how fast and at what throughput the scientific data can be compressed).

While many papers have been published on lossy compression techniques and reference datasets have been shared by the community [72], there is a lack of detailed specifications of application needs that can guide research and development of lossy compression techniques. This paper fills this gap by reporting on the requirements and constraints of nine scientific applications covering a large spectrum of domains (climate, combustion, cosmology, fusion, light sources, molecular dynamics, quantum circuit simulation, seismology, and system logs). For every application, the report details the motivations for compression, the current uses of compression, the compression requirements, the analysis and quality requirements, the performance requirements, the constraints on sustainability, integration, and installation, the special needs, and the expected changes to compression needs. Table 1 summarizes these needs for the nine applications.

Our contributions are: **First**, we present the most comprehensive and detailed study of 9 application domains' needs for compression ever conducted. **Second**, we significantly extend the state of the art for benchmarking in compression efforts by holistically describing a series of applications with requirements for specific compression ratio, throughput, and quality, along with datasets for each application area, instead of just available datasets lacking these requirements with generic quality metrics unrelated to application needs. This significantly improves the quality of benchmarking results for application scientists by making results more accurately reflect their use cases. **Third**, we consider practical barriers to adoption not extensively studied before for compression, including sustainability, needed integrations (e.g., language and library support), application-specific gaps in compression functionality that significantly affect performance or usability, and expected changes to these requirements. **Finally**, we identify 24 takeaways for Application Scientists, Compression Researchers, and Facilities to guide the adoption and future research of lossy compression.

The remainder of the paper is laid out as follows: In Section 2, we provide some basic terminology for compression. In Section 3, we describe the requirements gathering process used to perform the gap analysis. In Section 4, we survey the data collected for the application requirements and motivate their needs for compression and describe their quantities of interest. In Section 5, we describe the compression technologies investigated. Next, we present a gap analysis for the state of practice for error-bounded lossy compressors for Application Scientists in Section 6, for Compression Researchers in Section 7, and for Facilities in Section 8. We then present related studies and efforts in Section 9 and conclude with high-level takeaways from our efforts in Section 10.

2 BACKGROUND

Here we define a few key terms used in the remainder of the report. **Compression** refers to a process that reduces the footprint of data (i.e., how much space it occupies in memory or on disk). Data that is compressed can then be **decompressed** to restore the data to

its original footprint. If there is no difference between the original input/dataset, the operation is said to be **lossless**; otherwise, it is **lossy**. Lossless compressors tend not to achieve high enough compression ratios for scientific floating-point datasets to address the compression needs of applications, causing scientists to increasingly turn to lossy compression. However, to adopt compression, they need to ensure that the results of their scientific analyses are preserved. The most frequent way that modern lossy compressors accomplish this is with an **error bound**, a user-defined setting that controls or bounds the amount of allowed difference between the input data and the decompressed data. When we evaluate compressors in this report, we consider three key aspects. **Compression Ratio** refers to the size of the data before compression divided by the size of the data after compression. **Compression Throughput** refers to the rate at which a compressor can process an input dataset. **Quantity of Interest (QoI)** refers to the value or distribution of values that an application scientist wishes to preserve, such as a descriptive statistic, distribution, conservation law, or a visualization. It could refer to the raw data (e.g., maximum decompression error), but more often, QoIs are derived from the differences in or compared between the raw and decompressed data.

3 METHODOLOGY

The data for this report was collected over three days during an in-person meeting in February 2025. 38 Experts from 9 application domains and 8 compressor technologies presented their applications and the constraints and requirements regarding lossy compression. The compression experts presented their technologies in detail. The general presentation of the applications and compression technologies was followed by a series of one-to-one interviews between the application and compression experts to refine the specification of the requirements and constraints and to identify needs that were not expressed during the application presentation.

During these meetings, application and compression developers explored a consistent set of questions relating to the motivation for compression, current uses of compression in the field, key quantities of interest to preserve, key performance requirements in terms of compression ratio and bandwidth, data sustainability needs, key integrations needed for adoption (e.g., programming language bindings and I/O library plugins), specialized needs for particular applications, and expected changes in these needs. Likewise, each compressor developer team was surveyed to determine key operating principles, error control features, hardware support, unique features, and the impact of each compressor on applications.

This report is a condensed version highlighting takeaways derived from these interactions in collaboration with the application and compression technology experts. The report reflects the state of the art for application needs in March 2024 and the lossy compression technologies as of April 14, 2025. A more detailed version (38 pages) of this report is available on ArXiv.[1]

4 APPLICATION REQUIREMENTS

In this section, we provide a high-level motivation for each application, its need for compression, and its performance requirements, including bandwidth, compression ratio, and quantities of interest as specified by the application scientist during the interviews. A summary of the properties can be found in Table 1. A visual summary of each quantity of interest can be found in Figure 1. For justification and explanation of these requirements as well as additional application-specific detail, please see our full report.

Climate Understanding the Earth's climate system has long been of interest, particularly as a requirement for better predicting future climate states. Climate simulation models have become increasingly complex over the decades as computing resources have grown in power and sophistication [24, 67]. While these advances are desirable for more accurate and realistic simulations, the associated data storage requirements are often prohibitively large, since supercomputing storage capacities have not increased as rapidly as computational power and financial constraints limit the storage capacity available at many institutions [31]. Computational and storage costs were so high for the initial DYAMOND atmosphere-only model experiments that simulations were limited to 40 days, and 3D variable output was scant—in some cases outputted 12× less often compared with 2D data. The DYAMOND contribution from SCREAM (Simple Cloud-Resolving E3SM Model [58]), run at 3.25 km resolution, was nearly 4.5 TB of data per simulated day [9]. The reality is that climate scientists are often unable to store all of the simulation output that they would like, and this limitation directly impacts climate science research investigations. A key quality metric for Climate is dSSIM [59] (see Figure 1a). For climate data, a compression ratio that is 2-3× higher than what lossless compression can achieve is desired with a bandwidth exceeding that of CPU-based lossless compressors. A distinguishing aspects of climate applications are the number of tools used to interact with climate data and the indefinite longevity of the data.

Combustion While turbulent fluid motion is a common thread through computational fluid dynamics (CFD) applications, the multiphysics coupled with fluid motion spans many different subdisciplines, including chemistry in the gas phase and at surface interfaces, plasma physics critical to energy-efficient chemical manufacturing and fusion energy, aerosol growth and coagulation, and spray atomization and evaporation. CFD at the exascale on DOE leadership-class supercomputers runs on thousands of computational nodes powered by GPUs and generates massive volumes of primary data, requiring large amounts of storage and analysis of quantities of interest (QoIs). It is infeasible to store data at sufficient frequencies to capture highly intermittent phenomena that occur in these transient simulations. Key qualities of interest in combustion data are topological features representing superadiabatic regions (see Figure 1b), which are challenging to preserve. A compression ratio of 2-5× is important but, notably, the runtime of the compressor is not critical.

Cosmology The study of the universe on its largest scales and across its entire history explores some of the most exciting questions in fundamental physics: the nature of dark energy and dark matter, the origin of primordial fluctuations, the origin and evolution of galaxies, and the intergalactic medium. Interpreting the ongoing and future sky surveys involves solving an inverse problem: deducing underlying physics from observational data. Here, the numerical simulations play an essential role as a forward model, since they are the only accurate way to model the nonlinear evolution

[1]The latest version of the full report can be found at https://arxiv.org/abs/2503.20031

Table 1: Summary of Application Requirements (an l means compared to the best lossless compressor; an → indicates an expected change in requirements, a ⋆ or ⋆⋆ represents mature and very mature domains)

Application	Needs	Device	Target CR	Target Bandwidth	Longevity	Mechanism	Installation	Special Needs	Format
Climate⋆⋆	storage	CPU → GPU	$2-3\times l$	$>1\times l$	indefinite	Python, Julia, R, HDF5, pnetcdf	site modules, pip/conda, spack	uncorrelated dims	Dense → unstructured grid
Combustion	storage	GPU	$2\times-5\times$	not urgent	5-10 years	C++, Python, HDF5	manual, pip	high accuracy for feature preservation	Dense
Cosmology⋆	storage +throughput	GPU	$>10\times$	$>1\times l$	to be determined	HDF5, C++, Python	manual	hardware portable	Dense
Fusion	storage +throughput	GPU	$>5\times$	not urgent → $>1\times l$	ephemeral	HDF5 → Python	manual, site modules, pip	streaming, provence	2D dense
Light Sources⋆	throughput +storage	CPU → GPU, FPGA	$>10\times$	real-time 1 TB/s	10 Years	Python, C++	conda → spack	uncorrelated dims, hardware portable	Dense
Seismology	throughput +storage	GPU, CPU	$>20\times$	$>1\times l$	ephemeral	Fortran90, CUDA, Python	manual	asynchronous batching	Dense
Molecular Dynamics⋆	storage	CPU	>3	$>1\times$	10 Years	C, C++, HDF5	spack, manual	random block access	Particles
System Logs	throughput	CPU	$>10\times$	$>10\times l$	ephemeral	Python	pip, conda, spack	queries (à la SQL)	2D tables
Quantum Circuit	memory capacity	GPU, CPU	$2\times-10\times$	real-time 25 GB/s	ephemeral	Python	pip	high dimensional	20-30D dense

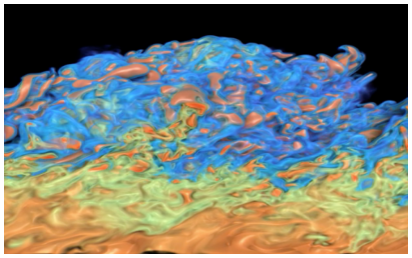
of the universe. Storage capacity requirements drive cosmology's needs for compression. On current exascale machines like OLCF's Frontier, a $16{,}384^3$-element simulation is doable based on reported Nyx [5, 42, 51] code efficiency and scalability. However, storage requirements are holding back the Nyx team from facilitating such a simulation, as a single double-precision array of $16{,}384^3$ elements is 32 TB in size. One checkpoint requires a minimum of 14 arrays, and a science case requires at least 20 time snapshots saved, resulting in a total storage requirement of almost 10 PB. For cosmology, a key quantity of interest is the Halo features in the data (see Figure 1c). A compression ratio greater than 10× at speeds faster than existing lossless compressors is desired. Notably, cosmology requires hardware portable decompression (e.g, compress on GPUs and decompress on CPUs).

Fusion This kind of energy holds the promise of a clean, baseload generation source of electricity in a decarbonized future. Magnetic confinement is one approach for achieving viable fusion energy, and tokamaks are the predominant experimental direction for magnetic confinement fusion today. The ITER tokamak, currently under construction in France, aims to demonstrate the technical feasibility of a burning plasma with a tenfold ($Q \geq 10$) power gain. Experimental tokamaks, although smaller than ITER, generate vast quantities of data through their extensive instrumentation. These datasets are multimodal and can accumulate over years of research campaigns, making data management a significant challenge. One major data source is electron cyclotron emission imaging (ECEi), used in tokamaks such as DIII-D in San Diego, CA. ECEi captures snapshots of electron temperature fluctuations at a high temporal frequency of 1 MHz and a spatial resolution of 20×8 grid points. The sheer volume of data generated by ECEi, especially considering that measurements are continuous and span extended periods, results in large datasets that can be difficult to store and manage without compression. For fusion, the quantity of interest to preserve is the spikes/peaks in disturbances (see Figure 1d). A compression ratio

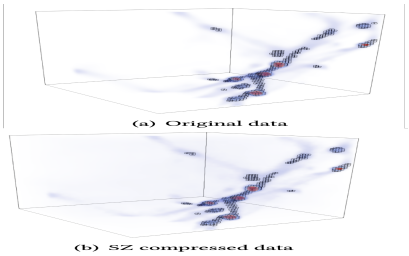
Difference calcs:

	conservative	middle_ground	aggressive
max abs diff	0.000244141	0.0078125	0.125
min abs diff	0	0	0
mean abs diff	0.000115477	0.00370324	0.0592399
mean squared diff	1.89216e-10	3.11831e-12	1.55522e-07
root mean squared diff	0.00013564	0.00433643	0.0692629
normalized root mean squared diff	1.23048e-06	3.93283e-05	0.000628213
normalized max pointwise error	2.06696e-06	7.52962e-05	0.00120917
pearson correlation coefficient	1	1	0.999996
ks p-value	1	1	0.181207
spatial relative error(% > 0.0001)	0	0	73.3037
max spatial relative error	9.53492e-07	3.03179e-05	0.000485199
DSSIM	0.999985	0.99807	0.974567
file size ratio	1.23	1.95	3.16

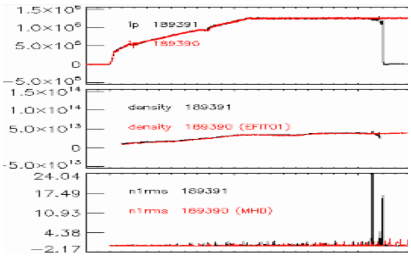
(a) Climate: LDCPY [47] output for the average daily surface temperature (TS) field of CESM [27] including DSSIM [59] on 3 levels of compression



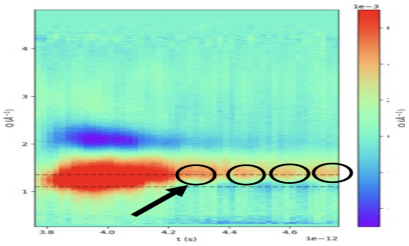
(b) Combustion: Small, fast-moving features representing superadiabatic regions with high combustion rates (orange) in instantaneous volume rendering of hydroxyl radical from simulation



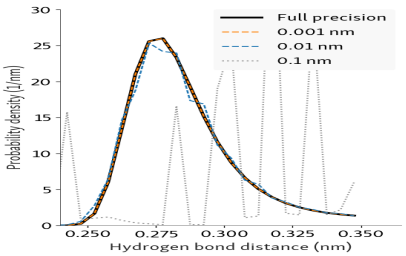
(c) Cosmology: Location, size, and properties of Halos identified by the Halo Finder from simulation



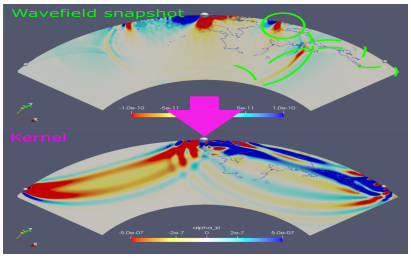
(d) Fusion: Presence, magnitude, and timing of a disruptions during an ECEi data trace from a fusion experiment



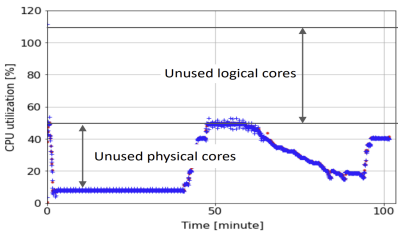
(e) Light sources: Reconstruction of SAX/WAX light source reconstruction data



(f) Molecular dynamics: Water-to-water hydrogen bond distance distribution. A sensitive measure that is disrupted for positional precision worse than 0.001 nm.



(g) Seismology: Wavefield snapshots used for kernel calculation.



(h) System logs: Distributions and trends in resource usage in a cosmology workflow executed with Parsl [7]

$$F(or, de) = |\langle \psi_{or} | \psi_{de} \rangle|$$

(i) Quantum circut: Preserving the fidelity of the circuit

Figure 1: Visual Summary of Example Quantities of Interest for each Application Class

greater than 5× is desired, and while runtimes are currently not critical, that is expected to change and will need to be faster than lossless compressors. Fusion applications need streaming support and strong provenance support to track changes to data.

Light Sources Devices like the Linac Coherent Light Source (LCLS) operated at the SLAC National Accelerator Laboratory (SLAC) and the Advanced Photon Source (APS) operated at Argonne National Laboratory allow scientists to improve their understanding of the materials that make up our universe, as well as improving our understanding and ability to construct advanced electronics, pharmaceuticals, and nanoscale technologies and to study the makeup of living things. Although the two facilities have significant differences, they both produce enormous volumes of data and are expected to produce more as upgrades are completed, which will result in dramatically increased X-ray pulse rates. For LCLS-II, the repetition rate will increase to 1 MHz compared to 120 Hz for LCLS-I. Over the next few years, the LCLS-II project at SLAC and APS-U at Argonne will deploy new area detectors for these high-rate ultrafast X-ray shots. At SLAC, the devices with the highest data volume are 16 megapixel area detectors running at 35 kHz, producing ~1 TB/s in just one experimental hutch. Facilities like the APS have over 70 such hutches. While each beamline differs (in some cases dramatically), compression ratios greater than 10 are desired at real-time bandwidths of up to 1TB/s. An example quantity of interest from a small-angle/wide-angle scattering at LCLS is the reconstruction (see Figure 1e).

Molecular Dynamics MD simulations examine the movement of particles within physical space to uncover the system's dynamic progression based on particle interactions. These simulations have become a crucial research tool across numerous scientific fields,

including physics, biology, and materials science. In biophysics and structural biology, MD simulations are widely used to investigate the behavior of macromolecules such as proteins and nucleic acids, facilitating the interpretation of biophysical experimental data and the modeling of molecular interactions [1, 3]. In materials science, MD simulations enable researchers to model and predict the structural, thermal, and mechanical properties of materials at the atomic level. This capability helps in understanding phenomena such as material deformation, fracture mechanics, and phase transitions, providing insights that are often inaccessible through direct experimental observation [61]. MD simulations have applications in numerous domains, where the generated data typically consists of particle coordinates as a function of time. The size of uncompressed binary coordinate trajectory files depends on the system size and simulation settings but is ordinarily tens to hundreds of gigabytes, emphasizing the need for efficient lossy compression algorithms—system sizes of biomolecular MD simulations commonly range from 100,000 atoms to a few million atoms, while the simulation time scales are often tens of nanoseconds to tens of microseconds, with trajectory frame writing intervals often in the range of 1 frame per 10 picoseconds to 1 frame per nanosecond [43, 54]. MD simulations need to preserve bonds and sequences as well as the radial distribution function (see Figure 1f). They need a compression ratio greater than 3 while not slowing down the application. MD simulations are notable in that they contain particle data.

Seismology This is a technology that creates high-fidelity images of the Earth's subsurface by analyzing the propagation and reflection of seismic waves. In energy industries, companies such as Saudi Aramco utilize seismic imaging to optimize resource (e.g., oil) extraction while minimizing environmental impact [26, 44]. Seismic imaging is also essential in various other domains [16, 19, 35], such as assessing the stability of tunnels and bridges [29], and even in planetary sciences [70], where it helps study the internal structures of celestial bodies such as the moon and Mars. Given its wide-ranging applications, improving the efficiency and accuracy of seismic imaging is critical for both scientific advancements and industrial applications. However, large-scale FWI workflows generate massive data volumes, since wavefield snapshots must be stored and retrieved during adjoint computations. Because of memory constraints, these snapshots are written to disk, creating significant I/O bottlenecks when reading and writing volumetric data, especially when processing multiple seismic events concurrently. Seismology workflows need high compression ratios (>20×) while achieving performance exceeding that of lossless compression. The quantity to preserve is the visualization of the stacking image (see Figure 1g). Seismology is notable in that some important applications still depend on Fortran 90.

System Logs Large-scale parallel and distributed applications generate enormous volumes of logging and monitoring data that must be aggregated and interpreted to understand the progress and performance of applications. One common example is the use of scientific workflows to orchestrate various scientific applications. Parsl [7], Ray [46], TaskVine [52], and Globus Compute [13] are examples of systems that coordinate the execution of many different tasks on parallel and distributed infrastructure. One challenge with task-based parallel and distributed applications, such as workflows and function-as-a-service platforms, is the need to monitor the execution performance of various tasks on parallel and distributed systems. Monitoring information is used both interactively in real time and after execution to investigate how an application performed, detect anomalies, and guide scheduling decisions. Providing rich monitoring information can represent a significant amount of data, as monitoring information is captured at a subsecond granularity from each worker in the system. Workflows running on a supercomputer may therefore have hundreds of thousands of workers concurrently capturing resource use. This presents a significant overhead for performance monitoring. For example, Parsl workflows have incurred up to an order of magnitude throughout degradation when monitoring is enabled [30]. Further, loss of monitoring information or delayed transmission can affect scheduling decisions, leading to reduced workflow performance. System logs are a new application for compression. They are expected to need a compression ratio of at least 10 to adopt compression while maintaining a bandwidth of the workflow manager exceeding 10× what is achievable with lossless compression. What is notable about system logs is that compression needs to preserve the values of queries that may be specified at runtime.

Quantum Circuit This kind of simulation is essential for advancing quantum computing, a rapidly growing field at the intersection of physics and computer science. Quantum physics enables the design of devices that have the potential to solve problems infeasible for classical computers. To develop and verify these technologies, quantum circuit simulators play a crucial role by allowing researchers to test quantum devices and evaluate quantum algorithms without requiring physical quantum hardware. These simulations help researchers optimize existing algorithms and explore new approaches. These simulations typically have one of two goals: finding a value of some observable and finding the probability of some set of states. The fundamental building block of a quantum computer is called a qubit. A quantum system of N qubits requires 2^N numbers to fully specify its state. The simulation of a quantum system then involves applying a sequence of operations or "gates" to the state. State vector simulators face several significant bottlenecks that limit their scalability and efficiency in simulating large quantum circuits. The most prominent bottleneck is the exponential growth of memory needed to store the quantum state vector. This scaling quickly exhausts available resources as the number of qubits increases, typically limiting state vector simulations to about 45 qubits on existing supercomputers. Quantum circuits use compression to increase memory capacity. They need real-time compression at 25 GB/s for current simulations at a compression ratio of between 2 to 10×. While doing so, they need to preserve conservation and distribution laws such as fidelity (see Figure 1i).

5 COMPRESSION CAPABILITIES PRIMER FOR APPLICATION SCIENTISTS

In this section, we provide a brief overview of the compression technologies discussed during the workshop for the benefit of application scientists and facilities staff guiding users on adopting compression. Much more detail regarding compression features, performance, history, and impact is provided in our full report.

SZ (https://szcompressor.org) is a prediction-based error-bounded lossy compressor. It is not only an off-the-shelf general-purpose

lossy compressor but also a composable framework allowing users to customize appropriate/effective compressors for specific applications or use cases. This framework allows users to create diverse compressors by easily implementing/customizing specific methods in five different stages: data preprocessing (e.g., transforming raw data to log domain for pointwise relative-error-bounded compression [36]), prediction (e.g., Lorenzo, linear regression [37] and spline interpolation [71]), quantizer (such as linear-scale quantization [56]), encoder (such as Huffman encoding), and lossless compressor (such as Zstd [15]). It has many variants specialized for particular applications and hardware devices.

ZFP (https://zfp.io) is primarily an in-memory compressed representation for multidimensional floating-point arrays that supports high-speed read and write random access at very fine granularity. ZFP offers an alternative number format for multidimensional arrays that exhibit "smoothness" or autocorrelation, as is the case with most fields representing physical quantities. Compared with IEEE floating point and many recent variants, such as BFloat16, TensorFloats, Posits, and Blaz, ZFP provides much higher accuracy per bit stored [40]. Error bounds can be specified, not just for a single application of ZFP compression [18], but also when ZFP is used in iterative methods [22], where compression errors may propagate and cascade.

MGARD (https://github.com/CODARcode/MGARD) is a lossy compression framework built on finite-element analysis and wavelet theories. A primary way that MGARD distinguishes itself is its robust mathematical notions of error bounds for quantities of interest, compared with other compressors, and support for structured non-Cartesian grids not offered by most other compressors. Another novel feature MGARD offers is data refactoring and progressive retrieval [38]. This mode archives data nearly losslessly using multilevel decomposition and bitplane encoding and allows for on-demand data retrieval with error control in an incremental fashion. This has been further incorporated with erasure encoding to reduce storage and network overhead while maintaining data availability [66].

LC (https://github.com/burtscher/LC-framework/) is a framework for automatically creating customized high-speed lossless and guaranteed-error-bounded lossy data compression algorithms for individual files or groups of files [6, 20, 21, 50]. It supports both exhaustive search for the best algorithm in the search space and a genetic-algorithm-based search for cases where the exhaustive search would take too long. LC can search for the best algorithm based solely on compression ratio or based on both compression ratio and throughput. In the latter case, it outputs the Pareto front, that is, a set of algorithms that represent different compression-ratio versus speed tradeoffs. LC generates bit-for-bit compatible parallelized CPU and GPU compressors.

SPERR (github.com/NCAR/SPERR) is a wavelet-based compressor tailored for 2D and 3D scientific data compression [33]. Compared with other established compressors, SPERR excels in compression efficiency: SPERR most likely uses the least amount of storage to achieve a specific compression quality, often by a comfortable margin [33]. SPERR also features two special decoding modes: *flexible-rate* and *multi-resolution*, which allow decoding using a prefix of the compressed bitstream and producing different resolutions, respectively.

DCTZ (https://github.com/swson/DCTZ) is a transform-based lossy compressor inspired by the discrete cosine transform (DCT), specifically DCT-II, and is designed to work with floating-point (single- or double-precision) values in scientific and Internet-of-Things datasets. DCTZ is a newer compressor introduced in 2019 [14, 45, 68, 69]. What distinguishes DCTZ from other compressors is its specific near-orthogonal transforms to decorrelate data and its quantization design.

TEZip (https://tezip.readthedocs.io/) or Time Evolutionary Zip is developed at RIKEN R-CCS and designed to compress time-evolutionary data by using deep learning for prediction. Inferring subsequent frames progressively from previous inference results could lead to gradual degradation in image accuracy. TEZip is uniquely optimized for time-series-based compression and utilizes the notion of key frames often used in video compression formats. To mitigate and maintain a level of accuracy, it is essential to periodically incorporate inference based on the original image data. This can be achieved through two approaches: Static Window-based Prediction (SWP) and Dynamic Window-based Prediction (DWP). TEZip uses both approaches that stabilize the inference quality by adjusting the prediction window to ensure more reliable continuity in the image sequence.

LibPressio (https://github.com/robertu94/libpressio) is not a compressor itself, but it provides a common, lightweight interface to many compressors, including all the compressors listed above. LibPressio provides several features critical to the adoption of compressors in scientific codes. (1) *Consistent API*: This allows compressors and applications to evolve independently, even as compressors change dramatically. (2) *Common Integration Mechanisms*: Each compressor does not need to develop language bindings or I/O library adapters. This enables applications to more easily adopt compression. Compressors benefit from greatly reduced development and maintenance costs. (3) *Generic Implementations of Compressor Features*: Generic implementations mean that these features do not need to be implemented separately for every compression library. For example (i) embedding of provenance in the compressed stream, (ii) automatic configuration of compressors [63], (iii) automatic CPU parallelization of thread-safe compression libraries, (iv) prediction of compression performance, (v) techniques to convert between bound types, and (vi) standard configuration-file formats.

6 GAP ANALYSIS FOR APPLICATION SCIENTISTS: RECOMMENDATIONS AND GAPS FOR STUDIED APPLICATIONS AND COMPRESSORS

Table 2 summarizes the gaps in device support, language and integration support, special features demanded by the applications, and performance (including compressor Throughput, Compression Ratio, and Quality of Interest) as well as recommendations for each application class. To earn a ◆, the compressor needs to meet all of the stated requirements as demonstrated by a paper documenting the capability. To earn a ⬙, the compressor needs to implement at least 50% of the requirements (e.g., quality and size but not speed). If a compressor meets less than 50% of the requirements, it earns a ◇. If the compressor still needs more evaluation to assign a score, it

	Molecular Dynamics	Climate	Light Sources	Cosmology	Siesmology	Combustion	Fusion	Quantum Circuits	System Logs
SZ	◆ ◆ ◆ ◆	◆ ◈ ◆ ◆	◆ ◆ ◆ ◆	◆ ◆ ◇ ◆	◆ ◆ ◆ ◆	◆ ◆ ◆ ◆★	◆ ◆ ◆ ▽	◆ ◆ ◆ ◆	◆ ◆ ◇ ▽
ZFP	◆ ◆ ◆ ◆	◆ ◈ ◆ ◆	◆ ◆ ◆ ◇	◆ ◆ ◆ ◆	◆ ◆ ◆ ◆	◆ ◆ ◆ ◆★	◆ ◆ ◆ ◆	◆ ◆ ◆ ▽	◆ ◆ ◇ ▽
MGARD	◆ ◈ ◇ ◇	◆ ◈ ◇ ◆	◆ ◈ ◇ ◇	◆ ◈ ◇ ◇	◆ ◆ ◇ ◇	◆ ◈ ◆ ◆★	◆ ◈ ◆ ◆	◆ ◈ ◆ ▽	◆ ◈ ◇ ◇
LC	◆ ◈ ◇ ◇	◆ ◈ ◇ ◇	◆ ◈ ◆ ◆	◆ ◈ ◆ ◆	◆ ◆ ◆ ◇	◆ ◈ ◆ ◆	◆ ◈ ◆ ▽	◆ ◈ ◆ ▽	◆ ◆ ◇ ▽
SPERR	◆ ◆ ◇ ▽	◆ ◈ ◇ ◆	◆ ◈ ◇ ◇	◇ ◈ ◇ ◇	◆ ◆ ◇ ◇	◇ ◈ ◇ ◆	◇ ◈ ◆ ◆	◆ ◈ ◆ ◇	◆ ◈ ◇ ◇
DCTZ	◆ ◈ ◇ ▽	◆ ◈ ◇ ▽	◆ ◈ ◇ ◇	◆ ◈ ◇ ▽	◆ ◇ ◇ ▽	◆ ◈ ◇ ▽	◆ ◈ ◆ ▽	◆ ◈ ◆ ◇	◆ ◆ ◇ ▽
TEZIP	◆ ◈ ◇ ◇	◆ ◈ ◇ ◇	◆ ◈ ◇ ◇	◆ ◈ ◇ ◇	◆ ◇ ◇ ◇	◆ ◈ ◇ ◇	◆ ◈ ◆ ◇	◆ ◈ ◆ ◇	◆ ◆ ◇ ◇
Recommendation	SZ(MSZ)	SZ(CLIZ),ZFP LibPressio	SZ(ROIBIN-SZ),LC LibPressio	SZ(cuSZ),ZFP,LC LibPressio	SZ(cuSZp),ZFP LibPressio	SZ(cuSZ,MSZ),ZFP,MGARD LibPressio	MGARD → ▽ LibPressio	SZ(cuSZx) LibPressio	▽

Table 2: Compression Feature Analysis. Key: devices, languages and mechanisms, special features, performance (CR, QoI, BW), ★native QoI support , ◆ acceptable support, ◈ acceptable using libpressio, ◆ ≥50% support, ◇ <50% support, ▽ needs evaluation for support. Acceptable performance indicates that domain scientists have evaluated and accepted this compressor's speed, size, and quality, but this is empirical and may not generalize. Recommendations are based on the highest scores in each category.

is marked with a ▽– most frequently this happens for performance when there is no automated way to check the preservation of quantities of interest. The few compressors that go beyond ◆ support to natively support a user-definable bound for the quantities of interest are marked with a ★. More work is needed to support this for more applications; see Section 7.2 for additional details on needed work for native QoI bounds. In case of SZ, a specific SZ variant is recommended per application. LibPressio is recommended whenever one or more compressors in the recommendations list need LibPressio to achieve their score. One application, System Logs, is a very new use case for lossy compression and does not yet have extensive evaluation experience is rated as unknown – a few compressors can be ruled out based on their throughput and compression ratio characteristics. A full explanation of the scores can be found in our full report.

It is important to note that a low score does not mean that a compression is not useful. For example, device support, language, and integrations, specialized features, and performance for compressors improve regularly. These results represent a snapshot in time when this paper was written. The authors will endeavor to update the arXiv report on a regular cadence to reflect the new state of the art as it develops. From the capability analysis presented in Table 2, we can identify the following takeaways:

Takeaway 1: *Many applications can already be well served by existing lossy compressors.*

Takeaway 2: *Very few applications have **all** their constraints and requirements satisfied by existing lossy compression, which calls for more collaborations between application scientists and experts in compression.*

7 GAP ANALYSIS FOR COMPRESSION RESEARCHERS

7.1 Use Cases and Hardware for Compression

The majority of applications considering compression need to save storage (7 of 9) or increase throughput (5 of 9), with slightly fewer applications looking to improve throughput. Of the 9 applications, 7 report that a compression ratio of at least 5 is required to adopt compression, and many describe this requirement as an improvement over lossless compression rather than no compression. Likewise, applications that describe throughput as a priority want to see application speedups that exceed what they can achieve with lossless compression; 2 of the 9 applications want compression to help them meet a real-time streaming bandwidth target. **Takeaway 3:** *This highlights that applications need specific throughput, compression, and quality targets, indicating a weakness in prior benchmarking approaches [72] that only consider one or two of these criteria.*

Of the 9 studied workflows, 6 need to perform compression on the CPU, a number that is expected to decrease to 4 out of 9, while 5 of the 9 applications require performing compression on the GPU, a number that is expected to increase to 7 out of 9 soon. This shift largely represents the shift of applications from CPUs to GPUs to leverage the GPUs in leadership-class computing facilities. One application group—light sources—called out the use of FPGAs as also increasingly important for their use cases. FPGAs are already used in these applications, so incorporating them for compression is consistent with other work in the field. One critical pair of applications not included in this report—AI training and inference—uses other forms of specialized hardware besides GPUs, such as TPUs and Cerebras wafer-scale engines. We intend to study use cases of compression in these and other applications in a future version of the long report. **Takeaway 4:** *Most applications use GPUs for compression. A few applications are starting to consider other accelerators (e.g., FPGA or AI accelerators).*

Of the 9 applications, 3 describe the need for interoperable compression and decompression on different hardware platforms; so far, only two compressors fully meet this requirement, and only one implements byte-for-byte interoperability. In some cases, this kind of interoperability can be difficult to efficiently implement (e.g., Huffman tree construction on a GPU [62]), difficult to implement correctly because of platform differences (e.g., LC reimplemented core math function to ensure byte-for-byte compatibility between CPUs and GPUs [21]), or difficult to implement on all platforms because of platform limitations (e.g., lack of global memory in Cerebras requiring an alternative Huffman tree implementation [53]), necessitating innovations in algorithms. More on this topic can be found in Item 3 of Section 8.1. **Takeaway 5:** *Interoperability between hardware is a common underserved need for compressors that requires additional research and effort from compression developers.*

7.2 Quantities of Interest to Preserve

One area where compressors can improve significantly is the preservation of higher-order quantities of interest. Of the 9 applications surveyed, all but 1 indicated that they found it difficult to preserve their quantities of interest with existing production lossy compressors. Three major groups of quantities of interest need additional focus by compression developers: derived quantities of interest, topological features, and distributional features. Additionally, configuration search and iterative error analysis are needed.

Derived quantities of interest are scalar values derived with an explicit formula from the data or its error (e.g., dSSIM and descriptive statistics). At least 4 of the 9 applications have at least one of these QOIs requiring preservation. While in many cases a relationship exists between the error bound and the derived QoI (see [57] for an example), it is nontrivial to explicitly derive this relationship. Some work in this area has been done [8, 41], but these techniques are either difficult to use, still requiring extensive mathematical proofs to establish correctness, or are not fully integrated into production compressors adopted by applications. Moreover, if one can derive the relationships between application-derived QoI, the bound may be very pessimistic, resulting in lower than otherwise required compression ratios [63], and the run performance of the approach may be unacceptably low [63] for applications with large datasets. **Takeaway 6:** *More work is needed on both theory and application to make techniques for preserving derived quantities of interest available to the applications that need them.*

Topological features refer to the minima, maxima, and critical points that exist in data and its integrals or derivatives. At least 3 of the 9 applications cite the need to preserve these kinds of QoIs. While compressors exist for these types of bounds as well, they are largely research prototypes with high overheads and lack integration into appropriate libraries and languages where applications would use them [34], or they overpreserve the data by preserving all derivatives as part of preserving the Sobolev norm of data [65], resulting in lower than desirec performance. The accessibility of the functionality aspect can be improved by the integration of existing or the development of new research prototypes of compressors using frameworks such as LibPressio that export these functions automatically, but resource utilization improvements come from both algorithmic improvements and productionizing the relevant codes. **Takeaway 7:** *The preservation of topological features requires improvements to performance and interface standardization/integration to better serve applications.*

Distributional features refer to the shape of the distribution of values either in some window or globally. At least 2 of the 9 applications need to preserve these kinds of features either in the data itself or in decompression errors. While the distribution of error bounds of compressors has been studied and characterized [39], this work is substantially out of date compared with current compressors and is merely descriptive. **Takeaway 8:** *Applications need prescriptive protection of the distribution of data values and errors, which is not supported by any major and possibly any research-grade compressor.*

Another key aspect of the adoption of compressors is the simplicity with which applications can specify their quantities of interest and identify configurations of compressors that can meet their requirements. Configuration search tools such as OptZConfig [63] included with LibPressio can help with this process, but the overhead of these methods can still be very high [48], and the tools are not scalable to applications with a large number of fields that potentially need to be configured differently. **Takeaway 9:** *Configuration search tools need lower overhead to help applications.*

Lastly, more research is needed to study the effects of iterative lossy compression to better support facilities and users who need to understand the effects of iterative compression to migrate from one compressor implementation to another or as part of more advanced techniques such as lossy checkpoint restart [10]. The only compressor for which a significant study of the effects of iterative lossy compression exists is ZFP [22]. There has also been some work considering PDEs where losses caused by a lossy restart are recovered by additional iterations [10], but this may not apply to all application types. **Takeaway 10:** *More work is needed to understand the cumulative errors of iterative lossy compression for compressors other than ZFP and in applications other than PDE solvers.*

7.3 Packaging and Integrations to Prioritize

Nearly all studied applications (8 of 9) use Python somewhere in their data analysis stack, so integration with Python is critical to the adoption of lossy compressors. Only 3 of the studied families of compressors have their own Python bindings, often with a subset of features, and Python packaging. The availability of Python bindings extends to 100% of the compressors with LibPressio bindings and most but not all features[2]. However, LibPressio does not automatically improve the packaging. **Takeaway 11:** *Compression developers better serve applications by targeting Python. Implementing compressors using LibPressio provides this automatically.*

Moreover, 5 of the 9 application areas utilize a lower-level language as a key component of their software stack. Of these 5 applications, 4 use C++ and 1 uses Fortran90. Fortran 2003 added minimal support for variable-length strings and C-style pointers, making it possible but complicated for compressors to support Fortran. Having individual compressors add support for Fortran 2003 or later via LibPressio is possible but would require substantial effort. For Fortran90, however, lacking this minimal support leaves no practical path to direct integration of compressors without resorting to nonstandard compiler extensions or the adoption of I/O libraries for Fortran that support compression, such as HDF5; and even this approach may not be possible given the subset of features of HDF5 available in Fortran90 [60]. **Takeaway 12:** *There is no practical way to support Fortran90 applications without the* `iso_c_binding` *extension added in Fortran2003.*

5 of 9 application domains report using HDF5 at some point in their data analysis stack. There are numerous documented weaknesses with HDF5's compression filters when using lossy compression: most notably, the lack of proper support for synchronizing with GPU/accelerator codes and the size limitation on compression configurations. This could potentially be addressed through implementing an HDF5 volume adapter to incorporate compression at a substantial increase in implementation complexity or via improvements to the HDF5 filter interface. **Takeaway 13:** *Many applications use HDF5, so supporting it is important for compression developers. LibPressio can provide this support.* **Takeaway 14:**

[2]For example, ZFP's array feature is not exported by LibPressio.

HDF5's filter interface needs substantial improvements to be suitable for more error-bounded lossy compressors.

7.4 Application-Specific Compression Needs and Future Research Directions

The need for greater support for data structures was cited by 3 of the 9 applications. Of these, 2 needed support for uncorrelated dimensions passed as a dense tensor. Without this feature, the compression ratio of compressors is unduly hampered by trying to relate unrelated elements of data stored in a dense tensor. The research compressor CLIz [28] supports this feature by identifying uncorrelated dimensions with autotuning, but this technique has not been adopted by major compressors. **Takeaway 15:** *Support for uncorrelated dimensions in dense tensors is important for many applications, but an uncommon feature in compressors.*

Multiple applications reporte needing support for random access decompression by block. SZ2 (but not SZ3), ZFP, and SPERR include an API for these functions, but they are low-level and not featured by higher-level abstractions in LibPressio that would work between compressors. LibPressio has functionality that can be used to implement a similar function generically with size and runtime overhead compared to native compressors. **Takeaway 16:** *Many compressors need random access by block decompression, but this feature is not widely implemented.*

A third of the applications, 3 of 9, need greater optimizations to meet bandwidth requirements during streaming. Light Sources need careful co-design between the compressor and the data reduction pipeline infrastructure to meet bandwidth requirements with available hardware, including optimizations to streaming, GPU kernel launches, and compression algorithms to meet hardware requirements. Fusion and system log applications also report the need to support streaming of data to alleviate bandwidth requirements. Streaming differs from traditional compression tasks in that the entire data is not available at once. **Takeaway 17:** *For streaming data, decisions need to be made to balance throughput and compression ratios, an area requiring further study.*

Of the 9 applications, 2 cited the need to support additional operations on compressed data. For example, System Logs require the ability to perform queries (à la SQL) on compressed data; no current compressor supports this operation. **Takeaway 18:** *Homomorphic compression [4] is an open and active research area in lossy compression; but since this is a newly identified need for applications, it requires further study.*

Some specific features are needed by individual applications, such as support for unstructured grid data. Some research-grade compressors do support this feature [49] but are not widely adopted nor supported by higher-level abstractions such as LibPressio. Without this support, compressors must treat this data as one-dimensional, which may limit the correlations that compressors can leverage to preserve quality and increase compression ratios. **Takeaway 19:** *A few applications reported needs for compressing unstructured grids, which is an area that needs more research and integration.* While some data-processing frameworks support compression of heterogeneous columns of data streamed over a network (i.e., streaming dataframes) [32], these frameworks do not include support for modern lossy compression and need further study. **Takeaway 20:** *Streaming and compressed dataframes are an open area of research in compression.*

8 GAP ANALYSIS FOR FACILITIES

8.1 Retention Policies for Compressed Data

While 44% of the studied applications cite a need for only ephemeral compression—that is, used as part of the workflow but discarded afterward (for example to accelerate MPI All Reduce [25])—the remaining 55% of the applications need long-term stability and support of the format of their compressed data to facilitate adoption. The most common duration cited was at least 5-10 years.

First, left unchecked, long retention periods combined with the increasing rates of data production and comparatively slower decreases in storage costs will eventually overwhelm storage systems. **Takeaway 21:** *Facilities may help alleviate storage demands by encouraging the adoption of compression and appropriate stewardship of storage resources through asking applicants how they plan to incorporate data reduction techniques, including data compression, as part of large-scale allocations.*

Second, the 5-10 year retention policy demanded by most applications means that many datasets need to live beyond the life span of a typical computing system. For example, considering systems introduced since 2008, the Argonne Leadership Computing Facility deprecates machines about every 6.4 years after they have been delivered. This means that plans to port compression libraries as part of the I/O stack from system to system are important as they enable continuity of data between systems. Current production compressors such as SZ, ZFP, and MGARD have each been successfully ported from CPU-based systems to systems such as Theta based on the Knights Landing architecture to GPU systems such as Aurora and Frontier, suggesting these kinds of ports are possible. **Takeaway 22:** *Facilities can help by asking applications with long retention requirements to plan for potential data format migrations from system to system as part of their data management plans.*

Third, bit-for-bit reproducibility of error-bounded lossy compressors with the introduction of changing hardware is a daunting task. Only one of the production compressors – LC – fully implements bit-for-bit reproducable compression across CPUs and GPUs. This required reimplimenting mathematical primitives to ensure consistent results between hardware platforms [21]. Performance portability libraries such as Kokkos, Raja, and Sycl are insufficient as they stand today on their own: if they provide the full set of primitives, they delegate to the standard library for implementations of functions like `std::log2`. They need to be compiled against a portable libc implementation such as llvm-libc and be compiled with `-mno-fma` or similar to obtain bit-for-bit reproducibility [21]. Other compressors implement error-bound reproducibility – i.e., the error bound is still correctly respected, but the decompressed values may differ within the error bound – but still benefit from performance portability frameworks that provide parallelism primitives to ease porting to new platforms. More research and development effort is needed for performance portability layers and techniques to support applications that need this kind of bit-for-bit reproducibility. **Takeaway 23:** *Facilities can help by ensuring the availability of performance, portability, and correctness of libraries to ease porting to*

new platforms, and with options for high-performance bit-for-bit consistent mathematical primitives (e.g., IEEE 754, `std::log2`*, disabling FMA, etc.) made available on new systems as differences in these lower-level primitives can yield incorrect results.*

8.2 Installing Compressors

Understanding how scientists install dependencies can inform how to best support them. Surprisingly, only one application domain reported using site modules as a significant way to deliver their application to supercomputers. Likewise, a large number of applications (55%) still compile all their dependencies from source as part of a manual installation process, further limiting the adoption of any dependencies, including compressors. 7 of 9 applications report the use an external package manager such pip, Anaconda, or Spack. The LibPressio maintainers make an effort to ensure that all supported compressors are installable via Spack [23], but this represents only 44% of the applications. Support in the Python packaging ecosystem for native libraries, especially around large complex C++ dependencies and GPU libraries, is lacking [2], making it difficult to support a large, complex, native ecosystem such as compression libraries, but some of the production compressors have Python pip and/or conda packages. **Takeaway 24:** *Given that applications are increasingly prioritizing non-site provided package sources, facilities should consider deploying site-wide caches and default configurations (e.g.,* `/etc/spack/packages.yaml` *or* `/etc/pip.conf`*) that are specialized for their particular site to ensure that users get optimal performance and fast package installations.*

9 RELATED WORK

The first signifigant paper considering the state of the practice for error-bounded lossy compression in HPC applications is a 2019 paper [12] that describes 7 use cases for lossy compression: visualization, reducing streaming intensity, reducing storage footprint, reducing I/O time, accelerating checkpoint restart, reducing memory footprint, and accelerating computation. The paper highlights one compressor per use case.

In 2020, research on lossy compression became more standardized with the introduction of the SDRBench Benchmark [73]. SDRBench includes datasets from 14 applications in 10 domains. This moved the state of the practice forward by presenting a consistent set of data for compression researchers to target while trying to achieve the best rate-distortion curve (least distortion per bit rate), typically using the PSNR to measure distortion. This was further improved in 2021, when the library LibPressio was introduced [64], which provides a consistent interface over the existing compressors, reducing sources of errors when comparing compressors (e.g., misunderstanding the differing definitions of error bounds or how dimension or data-type information is specified).

Since then, scientific lossy compressors have flourished. A recent survey of compression technologies by Di et al. [17] focuses on how the compressors function. It describes 10 compression principles: pointwise data prediction, quantization, wavelets, domain transforms, bit-plane codings, Tucker and singular value decompositions, sampling, filtering, lossless encoding, and deep neural networks. It discusses 46 distinct general compressors at a high level and 7 in detail. It concludes with a brief survey of the compression principles for specialized compressors covering 7 application areas (molecular dynamics, quantum chemistry, quantum circuit, climate, cosmology, seismology, light sources, and federated learning), and four additional specialized compressor features: topological feature preservation (e.g., minima, maxima, and critical points), multi-resolution data, communication optimization, and gradient sparsification. It calls out the need for an additional survey on compression on accelerators. However, this paper does not emphasize the needs of applications or gaps in how compressors address them.

A 2025 paper [11] considers 4 application domains: climate, instrument data (i.e., light sources), numerical methods, and AI with 2 categories of compression techniques: compression on the GPU and scheduling compressed I/O focusing only on technologies developed within the joint laboratory for extreme scale computing (JLESC). It identified the following four open problems: the upper bound of compressibility, compressed communication optimization, compression for AI in science as applied to AI model pre-taining, and combining lossy compression and encryption for increased performance in distributed integrated research infrastructure.

10 CONCLUSIONS

In this report, we present 24 important takeaways that inform Application Scientists, Compression Researchers, and Facilities to drive the use of lossy compression in scientific applications drawn from the most comprehensive study of application needs, compressor capabilities, and gap analysis ever conducted to assess the state of the practice in lossy compression.

The authors of this report are well aware that there are many more applications, including within the domains studied here, that are not adequately represented in this effort. For example, AI pretraining, fine-tuning, inference, and reasoning could be included in a future version of the full report. Application scientists, compression researchers, and facilities should contact the corresponding authors to incorporate their needs into this study and to improve the understanding of use cases of lossy compression for scientific data.

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, OAC-2034169, OAC-1751143, OAC-2330367, OAC-2313122, OAC-2311756, OIA-2327266 and OAC-2103621. 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's 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). We thank the Texas Advanced Computing Center (TACC) at the University of Texas at Austin for providing computational resources on the 'Frontera' system [55]. Use of the Linac Coherent Light Source (LCLS), SLAC National Accelerator Laboratory, is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract No. DEAC02-76SF00515. This work has been supported in part by the Department of Energy, Office of Science, under Award Number DE-SC0022223, as well as by equipment donations from NVIDIA Corporation. This work has been co-funded by the European Union through 'MDDB: Molecular Dynamics Data Bank. The European Repository for Biosimulation Data [101094651], and The Swedish e-Science Research Center.

REFERENCES

[1] [n. d.]. MDDB – Molecular Dynamics Data Bank. https://mddbr.eu/ URL: https://mddbr.eu/. Accessed 2024-06-14.

[2] 2022. Pypackaging-Native. https://pypackaging-native.github.io/.

[3] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C. Smith, Berk Hess, and Erik Lindahl. 2015. GROMACS: High performance Molecular Simulations Through Multi-Level Parallelism from Laptops to Supercomputers. *SoftwareX* 1–2 (Sept. 2015), 19–25. https://doi.org/10.1016/j.softx.2015.06.001

[4] Tripti Agarwal, Sheng Di, Jiajun Huang, Yafan Huang, Ganesh Gopalakrishnan, Robert Underwood, Kai Zhao, Xin Liang, Guanpeng Li, and Franck Cappello. 2024. HoSZp: An Efficient Homomorphic Error-bounded Lossy Compressor for Scientific Data. https://doi.org/10.48550/arXiv.2408.11971

[5] Ann S. Almgren, John B. Bell, Mike J. Lijewski, Zarija Lukić, and Ethan Van Andel. 2013. Nyx: A Massively Parallel AMR Code for Computational Cosmology. *The Astrophysical Journal* 765, 1, Article 39 (March 2013), 39 pages. https://doi.org/10.1088/0004-637X/765/1/39 arXiv:1301.4498 [astro-ph.IM]

[6] Noushin Azami, Alex Fallin, and Martin Burtscher. 2025. Efficient Lossless Compression of Scientific Floating-Point Data on CPUs and GPUs. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1* (Rotterdam, Netherlands) *(ASPLOS '25)*. Association for Computing Machinery, New York, NY, USA, 395–409. https://doi.org/10.1145/3669940.3707280

[7] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S. Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M. Wozniak, Ian Foster, Michael Wilde, and Kyle Chard. 2019. Parsl: Pervasive Parallel Programming in Python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing* (Phoenix, AZ, USA) *(HPDC '19)*. Association for Computing Machinery, New York, NY, USA, 25–36. https://doi.org/10.1145/3307681.3325400

[8] Tania Banerjee, Jong Choi, Jaemoon Lee, Qian Gong, Jieyang Chen, Scott Klasky, Anand Rangarajan, and Sanjay Ranka. 2022. Scalable hybrid learning techniques for scientific data compression. *arXiv preprint arXiv:2212.10733* (2022).

[9] P. M. Caldwell, C. R. Terai, B. Hillman, N. D. Keen, P. Bogenschutz, W. Lin, H. Beydoun, M. Taylor, L. Bertagna, A. M. Bradley, T. C. Clevenger, A. S. Donahue, C. Eldred, J. Foucar, J.-C. Golaz, O. Guba, R. Jacob, J. Johnson, J. Krishna, W. Liu, K. Pressel, A. G. Salinger, B. Singh, A. Steyer, P. Ullrich, D. Wu, X. Yuan, J. Shpund, H.-Y. Ma, and C. S. Zender. 2021. Convection-permitting simulations with the E3SM global atmosphere model. *Journal of Advances in Modeling Earth Systems* 13, 11 (2021). https://doi.org/10.1029/2021MS002544

[10] Jon Calhoun, Franck Cappello, Luke N Olson, Marc Snir, and William D Gropp. 2019. Exploring the feasibility of lossy compression for PDE simulations. *The International Journal of High Performance Computing Applications* 33, 2 (March 2019), 397–410. https://doi.org/10.1177/1094342018762036 Publisher: SAGE Publications Ltd STM.

[11] Franck Cappello, Mario Acosta, Emmanuel Agullo, Hartwig Anzt, Jon Calhoun, Sheng Di, Luc Giraud, Thomas Grützmacher, Sian Jin, Kentaro Sano, Kento Sato, Amarjit Singh, Dingwen Tao, Jiannan Tian, Tomohiro Ueno, Robert Underwood, Frédéric Vivien, Xavier Yepes, Yoshii Kazutomo, and Boyuan Zhang. 2025. Multifacets of lossy compression for scientific data in the Joint-Laboratory of Extreme Scale Computing. *Future Generation Computer Systems* 163 (Feb. 2025), 107323. https://doi.org/10.1016/j.future.2024.05.022

[12] Franck Cappello, Sheng Di, Sihuan Li, Xin Liang, Ali Murat Gok, Dingwen Tao, Chun Hong Yoon, Xin-Chuan Wu, Yuri Alexeev, and Frederic T Chong. 2019. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications* 33, 6 (2019), 1201–1220.

[13] Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ian Foster, and Kyle Chard. 2020. funcX: A Federated Function Serving Fabric for Science. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing* (Stockholm, Sweden) *(HPDC '20)*. Association for Computing Machinery, New York, NY, USA, 65—-76. https://doi.org/10.1145/3369583.3392683

[14] Jiaxi Chen, Aekyeung Moon, and Seung Woo Son. 2022. Towards Guaranteeing Error Bound in DCT-based Lossy Compression. In *2022 IEEE International Conference on Big Data (Big Data)*. 3139–3145. https://doi.org/10.1109/BigData55660.2022.10020345

[15] Yann Collet. 2015. Zstandard – Real-time data compression algorithm. *http://facebook.github.io/zstd/* (2015).

[16] Gideon Oluseyi Daramola, Boma Sonimiteim Jacks, Olakunle Abayomi Ajala, and Abiodun Emmanuel Akinoso. 2024. Enhancing oil and gas exploration efficiency through ai-driven seismic imaging and data analysis. *Engineering Science & Technology Journal* 5, 4 (2024), 1473–1486.

[17] Sheng Di, Jinyang Liu, Kai Zhao, Xin Liang, Robert Underwood, Zhaorui Zhang, Milan Shah, Yafan Huang, Jiajun Huang, Xiaodong Yu, Congrong Ren, Hanqi Guo, Grant Wilkins, Dingwen Tao, Jiannan Tian, Sian Jin, Zizhe Jian, Daoce Wang, MD Hasanur Rahman, Boyuan Zhang, Jon C. Calhoun, Guanpeng Li, Kazutomo Yoshii, Khalid Ayed Alharthi, and Franck Cappello. 2024. A Survey on Error-Bounded Lossy Compression for Scientific Datasets. arXiv:2404.02840 (April 2024). http://arxiv.org/abs/2404.02840 arXiv:2404.02840 [cs].

[18] James D. Diffenderfer, Alyson L. Fox, Jeffrey A. F. Hittinger, Geoffrey D. Sanders, and Peter G. Lindstrom. 2019. Error Analysis of ZFP Compression for Floating-Point Data. *SIAM Journal on Scientific Computing* 41, 3 (6 2019), A1867–A1898. https://doi.org/10.1137/18M1168832

[19] Kai-Uwe Doerr, Falko Kuester, Derek Nastase, and Tara C Hutchinson. 2008. Development and evaluation of a seismic monitoring system for building interiors—Part II: Image data analysis and results. *IEEE Transactions on Instrumentation and Measurement* 57, 2 (2008), 345–354.

[20] Alex Fallin, Noushin Azami, Sheng Di, Franck Cappello, and Martin Burtscher. 2025. Fast and Effective Lossy Compression on GPUs and CPUs with Guaranteed Error Bounds. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 874–887. https://doi.org/10.1109/IPDPS64566.2025.00083

[21] Alex Fallin and Martin Burtscher. 2024. Lessons Learned on the Path to Guaranteeing the Error Bound in Lossy Quantizers. In *Workshop on Correct Data Compression*. Montreal, Canada. https://doi.org/10.48550/arXiv.2407.15037 arXiv:2407.15037 [cs.DC]

[22] Alyson L. Fox, James D. Diffenderfer, Jeffrey A. F. Hittinger, Geoffrey D. Sanders, and Peter G. Lindstrom. 2020. Stability Analysis of Inline ZFP Compression for Floating-Point Data in Iterative Methods. *SIAM Journal on Scientific Computing* 42, 5 (2020), A2701–A2730. https://doi.org/10.1137/19M126904X

[23] Todd Gamblin, Matthew LeGendre, Michael R. Collette, Gregory L. Lee, Adam Moody, Bronis R. de Supinski, and Scott Futral. 2015. The Spack package manager: bringing order to HPC software chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, Austin Texas, 1–12. https://doi.org/10.1145/2807591.2807623

[24] Andrew Gettelman and Richard B. Rood. 2016. *Demystifying Climate Models: A Users Guide to Earth System Models*. Springer.

[25] Jiajun Huang, Sheng Di, Xiaodong Yu, Yujia Zhai, Jinyang Liu, Zizhe Jian, Xin Liang, Kai Zhao, Xiaoyi Lu, Zizhong Chen, Franck Cappello, Yanfei Guo, and Rajeev Thakur. 2024. hZCCL: Accelerating Collective Communication with Co-Designed Homomorphic Compression. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15. https://doi.org/10.1109/SC41406.2024.00110

[26] Yafan Huang, Kai Zhao, Sheng Di, Guanpeng Li, Maxim Dmitriev, Thierry-Laurent D Tonellot, and Franck Cappello. 2023. Towards improving reverse time migration performance by high-speed lossy compression. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 651–661.

[27] J. Hurrell, M. Holland, P. Gent, S. Ghan, J. Kay, P. Kushner, J.-F. Lamarque, W. Large, D. Lawrence, K. Lindsay, W. Lipscomb, M. Long, N. Mahowald, D. Marsh, R. Neale, P. Rasch, S. Vavrus, M. Vertenstein, D. Bader, W. Collins, J. Hack, J. Kiehl, and S. Marshall. 2013. The Community Earth System Model: A Framework for Collaborative Research. *Bulletin of the American Meteorological Society* 94 (2013), 1339–1360. https://doi.org/10.1175/BAMS-D-12-00121.1

[28] Zizhe Jian, Sheng Di, Jinyang Liu, Kai Zhao, Xin Liang, Haiying Xu, Robert Underwood, Shixun Wu, Zizhong Chen, and Franck Cappello. 2024. CliZ: Optimizing Lossy Compression for Climate Datasets with Adaptive Fine-tuned Data Prediction. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE.

[29] EJ Kase and TA Ross. 2004. Seismic Imaging to Accurately Characterize Subsurface Ground Conditions. In *17th EEGS Symposium on the Application of Geophysics*

to Engineering and Environmental Problems. European Association of Geoscientists & Engineers, cp–186.

[30] Jamison Kerney, Ioan Raicu, John Raicu, and Kyle Chard. 2024. Towards Fine-Grained Parallelism in Parallel and Distributed Python Libraries. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 706–715. https://doi.org/10.1109/IPDPSW63119.2024.00133

[31] Julian Martin Kunkel, Michael Kuhn, and Thomas Ludwig. 2014. Exascale Storage Systems–An Analytical Study of Expenses. *Supercomputing Frontiers and Innovations* 1, 1 (2014). http://superfri.org/superfri/article/view/20

[32] Geoffrey Lentner. 2019. Shared Memory High Throughput Computing with Apache Arrow™. In *Practice and Experience in Advanced Research Computing 2019: Rise of the Machines (learning) (PEARC '19)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/3332186.3335197

[33] Shaomeng Li, Peter Lindstrom, and John Clyne. 2023. Lossy Scientific Data Compression with SPERR. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 1007–1017. https://doi.org/10.1109/IPDPS54959.2023.00104

[34] Yuxiao Li, Xin Liang, Bei Wang, Yongfeng Qiu, Lin Yan, and Hanqi Guo. 2025. MSz: An Efficient Parallel Algorithm for Correcting Morse–Smale Segmentations in Error-Bounded Lossy Compressors. *IEEE Trans. Vis. Comput. Graph.* 31, 1 (2025), 130–140. https://doi.org/10.1109/TVCG.2024.3456337

[35] Zhen-Chun Li and Ying-Ming Qu. 2022. Research progress on seismic imaging technology. *Petroleum Science* 19, 1 (2022), 128–146.

[36] Xin Liang, Sheng Di, Dingwen Tao, Zizhong Chen, and Franck Cappello. 2018. An efficient transformation scheme for lossy data compression with point-wise relative error bound. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 179–189.

[37] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2018. Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets. In *2018 IEEE International Conference on Big Data*. IEEE.

[38] Xin Liang, Qian Gong, Jieyang Chen, Ben Whitney, Lipeng Wan, Qing Liu, David R Pugmire, Rick Archibald, Norbert Podhorszki, and Scott Klasy. 2021. Error-controlled, Progressive, and Adaptable Retrieval of Scientific Data with Multilevel Decomposition. In *SC'21: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 1–10.

[39] Peter Lindstrom. 2017. Error Distributions of Lossy Floating-Point Compressors. In *Joint Statistical Meetings*. U.S. Department of Energy Office of Scientific and Technical Information, Baltimore, Maryland, United States, 1–18. https://www.osti.gov/biblio/1526183

[40] Peter Lindstrom, Jeffrey Hittinger, James Diffenderfer, Alyson Fox, Daniel Osei-Kuffuor, and Jeffrey Banks. 2025. ZFP: A compressed array representation for numerical computations. *International Journal of High Performance Computing Applications* 39, 1 (2025), 104–122. https://doi.org/10.1177/10943420241284023

[41] Jinyang Liu, Pu Jiao, Kai Zhao, Xin Liang, Sheng Di, and Franck Cappello. 2024. QPET: A Versatile and Portable Quantity-of-Interest-preservation Framework for Error-Bounded Lossy Compression. https://doi.org/10.48550/arXiv.2412.02799 arXiv:2412.02799 [cs].

[42] Zarija Lukić, Casey W. Stark, Peter Nugent, Martin White, Avery A. Meiksin, and Ann Almgren. 2015. The Lyman α forest in optically thin hydrodynamical simulations. *Monthly Notices of the Royal Astronomical Society* 446, 4 (Feb. 2015), 3697–3724. https://doi.org/10.1093/mnras/stu2377 arXiv:1406.6361 [astro-ph.CO]

[43] Magnus Lundborg, Rossen Apostolov, Daniel Spångberg, Anders Gärdenäs, David van der Spoel, and Erik Lindahl. 2014. An efficient and extensible format, library, and API for binary trajectory data from molecular simulations. *J. Comput. Chem.* 35, 3 (2014), 260–269. https://doi.org/10.1002/jcc.23495

[44] Mikhail Malovichko, Denis Sabitov, Maxim Dmitriev, and Timur Zharnikov. 2025. Towards the shear-wave sonic reverse time migration with the spectral element method. *Journal of Applied Geophysics* 233 (2025), 105573.

[45] Aekyeung Moon, Jiaxi Chen, Seung Woo Son, and Minjun Kim. 2022. Characterization of Transform-Based Lossy Compression for HPC Datasets. In *2022 IEEE/ACM 8th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD)*. 56–62. https://doi.org/10.1109/DRBSD56682.2022.00013

[46] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. 2018. Ray: a distributed framework for emerging AI applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation* (Carlsbad, CA, USA) *(OSDI'18)*. USENIX Association, USA, 561–577.

[47] Alexander Pinard, Dorit M. Hammerling, and Allison H. Baker. 2020. Assessing Differences in Large Spatio-temporal Climate Datasets with a New Python package. In *2020 IEEE International Conference on Big Data (Big Data)*. 2699–2707. https://doi.org/10.1109/BigData50022.2020.9378100

[48] Md Hasanur Rahman, Sheng Di, Kai Zhao, Robert Underwood, Guanpeng Li, and Franck Cappello. 2023. A Feature-Driven Fixed-Ratio Lossy Compression Framework for Real-World Scientific Datasets. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1461–1474.

[49] Congrong Ren, Xin Liang, and Hanqi Guo. 2024. A Prediction-Traversal Approach for Compressing Scientific Data on Unstructured Meshes with Bounded Error. *Computer Graphics Forum* 43, 3 (2024), e15097. https://doi.org/10.1111/cgf.15097

[50] Andrew Rodriguez, Noushin Azami, and Martin Burtscher. 2024. Adaptive Per-File Lossless Compression of Floating-Point Data. In *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 423–430. https://doi.org/10.1109/IPDPSW63119.2024.00092

[51] Jean Sexton, Zarija Lukic, Ann Almgren, Chris Daley, Brian Friesen, Andrew Myers, and Weiqun Zhang. 2021. Nyx: A Massively Parallel AMR Code for Computational Cosmology. *Journal of Open Source Software* 6, 63 (2021), 3068. https://doi.org/10.21105/joss.03068

[52] Barry Sly-Delgado, Thanh Son Phung, Colin Thomas, David Simonetti, Andrew Hennessee, Ben Tovar, and Douglas Thain. 2023. TaskVine: Managing In-Cluster Storage for High-Throughput Data Intensive Workflows. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (Denver, CO, USA) *(SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1978–1988. https://doi.org/10.1145/3624062.3624277

[53] Shihui Song, Yafan Huang, Peng Jiang, Xiaodong Yu, Weijian Zheng, Sheng Di, Qinglei Cao, Yunhe Feng, Zhen Xie, and Franck Cappello. 2024. CereSZ: Enabling and Scaling Error-bounded Lossy Compression on Cerebras CS-2. In *The 33rd International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*.

[54] Daniel Spångberg, Daniel S. D. Larsson, and David van der Spoel. 2011. Trajectory NG: portable, compressed, general molecular dynamics trajectories. *J. Mol. Model.* 17, 10 (Oct. 2011), 2669–2685. https://doi.org/10.1007/s00894-010-0948-5

[55] Dan Stanzione, John West, R. Todd Evans, Tommy Minyard, Omar Ghattas, and Dhabaleswar K. Panda. 2020. Frontera: The Evolution of Leadership Computing at the National Science Foundation. In *Practice and Experience in Advanced Research Computing 2020: Catch the Wave* (Portland, OR, USA) *(PEARC '20)*. Association for Computing Machinery, New York, NY, USA, 106–111. https://doi.org/10.1145/3311790.3396656

[56] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. 2017. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 1129–1139.

[57] Dingwen Tao, Sheng Di, X. Liang, Z. Chen, and Franck Cappello. 2018. Fixed-PSNR Lossy Compression for Scientific Data. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. 314–318. https://doi.org/10.1109/CLUSTER.2018.00048

[58] Mark Taylor, Peter Caldwell, Luca Bertagna, Aaron Donahue, James Foucar, Oksana Guba, Benjamin Hillman, Noel Keen, Jayesh Krishna, Matthes Norman, Sarat Sreepathi, Christopher Terai, James White, Danqing Wu, Andrew Salinger, Renata McCoy, L. Ruby Leung, and David Bader. 2023. The simple cloud-resolving E3SM atmosphere model running on the Frontier Exascale System. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'23)*. https://doi.org/10.1145/3581784.3627044

[59] **Baker, Allison H.**, Alexander Pinard, and Dorit M. Hammerling. 2023. On a Structural Similarity Index Approach for Floating-Point Data. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–13. https://doi.org/10.1109/TVCG.2023.3332843

[60] The HDF Group. 2011. *New Features in the HDF5 Fortran Library: Adding Support for the Fortran 2003 Standard*. Technical Report. 1–19 pages.

[61] Aidan P. Thompson, H. Metin Aktulga, Richard Berger, Dan S. Bolintineanu, W. Michael Brown, Paul S. Crozier, Pieter J. In 'T Veld, Axel Kohlmeyer, Stan G. Moore, Trung Dac Nguyen, Ray Shan, Mark J. Stevens, Julien Tranchida, Christian Trott, and Steven J. Plimpton. 2022. LAMMPS – a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.* 271 (Feb. 2022), 108171. https://doi.org/10.1016/j.cpc.2021.108171

[62] Jiannan Tian, Cody Rivera, Sheng Di, Jieyang Chen, Xin Liang, Dingwen Tao, and Franck Cappello. 2021. Revisiting Huffman Coding: Toward Extreme Performance on Modern GPU Architectures. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 881–891. https://doi.org/10.1109/IPDPS49936.2021.00097 ISSN: 1530-2075.

[63] Robert Underwood, Jon C Calhoun, Sheng Di, Amy Apon, and Franck Cappello. 2022. OptZConfig: Efficient Parallel Optimization of Lossy Compression Configuration. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (Dec. 2022), 3505–3519. https://doi.org/10.1109/TPDS.2022.3154096

[64] Robert Underwood, Victoriana Malvoso, Jon C. Calhoun, Sheng Di, and Franck Cappello. 2021. 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)*. IEEE, St. Louis, Missouri, 1–10. https://doi.org/10.1109/DRBSD754563.2021.00005

[65] Brown Univeristy and Oak Ridge National Laboratory. [n. d.]. MGARD: MultiGrid Adaptive Reduction of Data. https://github.com/CODARcode/MGARD. Online.

[66] Lipeng Wan, Jieyang Chen, Xin Liang, Ana Gainaru, Qian Gong, Qing Liu, Ben Whitney, Joy Arulraj, Zhengchun Liu, Ian Foster, et al. 2023. RAPIDS: Reconciling Availability, Accuracy, and Performance in Managing Geo-Distributed Scientific

Data. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*. 87–100.

[67] Warren M Washington and Claire Parkinson. 2005. *Introduction to three-dimensional climate modeling*. University science books.

[68] Jialing Zhang, Jiaxi Chen, Aekyeung Moon, Xiaoyan Zhuo, and Seung Woo Son. 2020. Bit-Error Aware Quantization for DCT-based Lossy Compression. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–7. https://doi.org/10.1109/HPEC43674.2020.9286177

[69] Jialing Zhang, Xiaoyan Zhuo, Aekyeung Moon, Hang Liu, and Seung Woo Son. 2019. Efficient Encoding and Reconstruction of HPC Datasets for Checkpoint/Restart. In *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*. 79–91. https://doi.org/10.1109/MSST.2019.00-14

[70] DaPeng Zhao, JianShe Lei, and Lucy Liu. 2008. Seismic tomography of the Moon. *Chinese Science Bulletin* 53, 24 (2008), 3897–3907.

[71] Kai Zhao, Sheng Di, Maxim Dmitriev, Thierry-Laurent D. Tonellot, Zizhong Chen, and Franck Cappello. 2021. Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 1643–1654. https://doi.org/10.1109/ICDE51399.2021.00145

[72] Kai Zhao, Sheng Di, Xin Lian, Sihuan Li, Dingwen Tao, Julie Bessac, Zizhong Chen, and Franck Cappello. 2020. SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors. In *2020 IEEE International Conference on Big Data (Big Data)*. 2716–2724.

[73] Kai Zhao, Sheng Di, Xin Lian, Sihuan Li, Dingwen Tao, Julie Bessac, Zizhong Chen, and Franck Cappello. 2020. SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors. In *2020 IEEE International Conference on Big Data (Big Data)*. 2716–2724. https://doi.org/10.1109/BigData50022.2020.9378449