Luddy School of Informatics, Computing, and Engineering

Revolutionizing I/O Performance: Lossy Compression Meets HDF5

Dingwen Tao, Associate Professor

Indiana University Bloomington

Storage and I/O Issues in HPC Systems

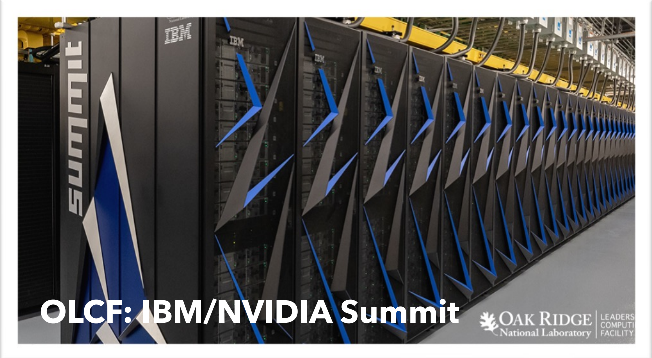
SUPERCOMPUTER SYSTEM	YEAR	CLASS	PEAK FLOPS (PF)	MEMORY SIZE (MS)	STORAGE BAND-WIDTH (SB)	MS/SB	PF/SB
Cray Jaguar	2008	1 PFLOPS	1.75 PFLOPS	360 TB	240 GB/s	1.5k	7.3k
Cray Blue Waters	2012	10 PFLOPS	13.3 PFLOPS	1.5 PB	1.1 TB/s	1.3k	13.3k
Cray CORI	2017	10 PFLOPS	30 PFLOPS	1.4 PB	1.7 TB/s (**)	0.8k	17k
IBM Summit	2018	100 PFLOPS	200 PFLOPS	> 10 PB (*)	2.5 TB/s	> 4k	80k

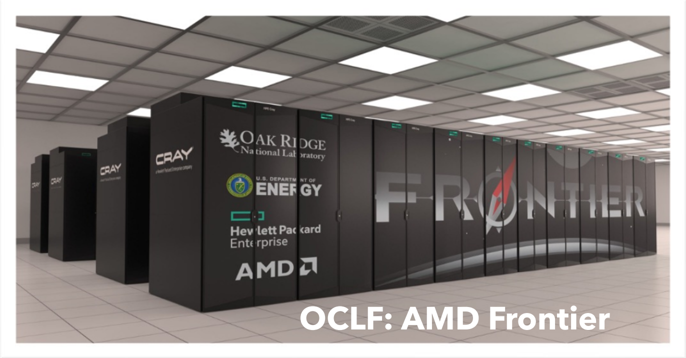
(*) when using burst buffer (**) counting only DDR4 source: F. Cappello (ANL)

SUPERCOMPUTER SYSTEM	YEAR	CLASS	PEAK FLOPS (PF)	MEMORY SIZE (MS)	STORAGE BAND-WIDTH (SB)	MS/SB	PF/SB
Fujitsu Fugaku	2020	"ExaScale"	537 PFLOPS (*)	4.85 PB	> 1.5 TB/s (**)	> 3.23k	358k
AMD Frontier	2021	ExaScale	1.6 EFLOPS	9.2 PB (a)	10 TB/s	> 0.92k	160k
Intel Aurora (#)	future	ExaScale	> 2 EFLOPS	> 10 PB (a)	>= 25 TB/s	> 0.40k	80k

(*) Rpeak, Top-500 as of November 2020 (**) DDN Newsroom
(a) aggregated memory (CPU DDR + GPU HBM)

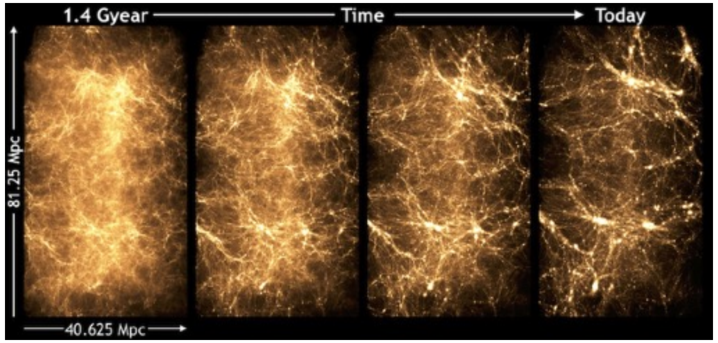
The compute capability is ever-growing, but storage capacity and bandwidth are developing much **more slowly**

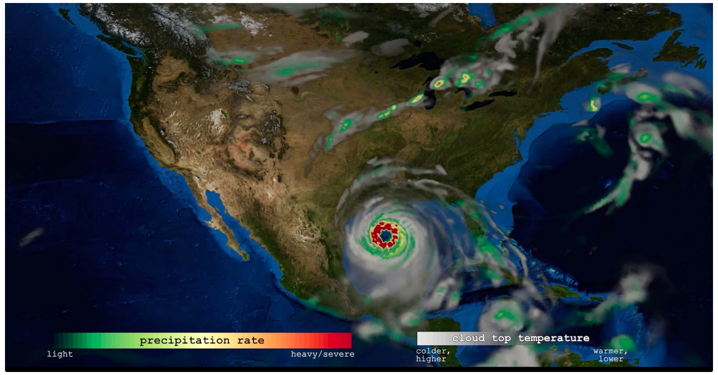




Data Management Issues for Scientific Applications

application	data scale	bottleneck	reduce by
HACC cosmology simulation	**20 PB** one-trillion-particle	**use up filesystem (26 PB in total)** Mira@ANL	**10×** in need
CESM climate simulation	**50%** vs 20% storage in hardware budget, **2017** vs 2013	**5h30m to store** NSF Blue Waters 1-TBps I/O	**10×** in need
APS-U High-Energy X-Ray Beams Experiments	**10^2 PB** Brain Initiatives	**saturate connection** 100 GBps bandwidth	**100×** in need





Our Solution – Error-Bounded Lossy Compression

2:1 **(FP-type)**

lossless on scientific datasets

10:1 **or higher**

reduction ratio in need

industry
lossy compressor (JPEG)

high in reduction rate,
but **not** suitable for **HPC**

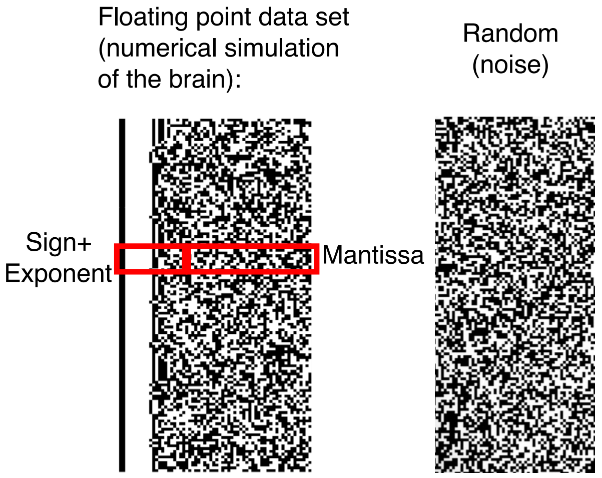
need **diverse** compression modes

1) absolute error bound (infinity-norm)
2) pointwise relative error bound
3) RMSE error bound (2-norm)
4) fixed bitrate
5) satisfying post-analysis requirements

SZ

Di and Cappello 2016, Tao *et al.* 2017, Xin *et al.* 2018, Tian *et al.* 2020

> prediction-based lossy compressor framework for scientific data
> strictly control the global upper bound of compression error
> implemented on CPU, GPU, FPGA
> integrated in I/O libraries (HDF5, ADIOS, PnetCDF)



Source: Leonardo Bautista Gomez (BSC)

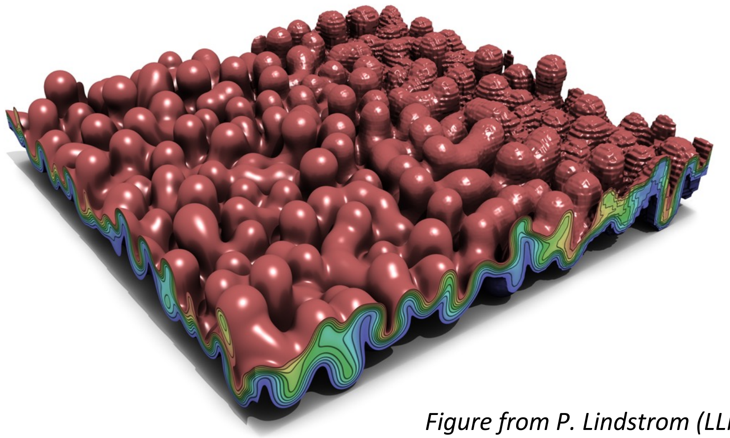
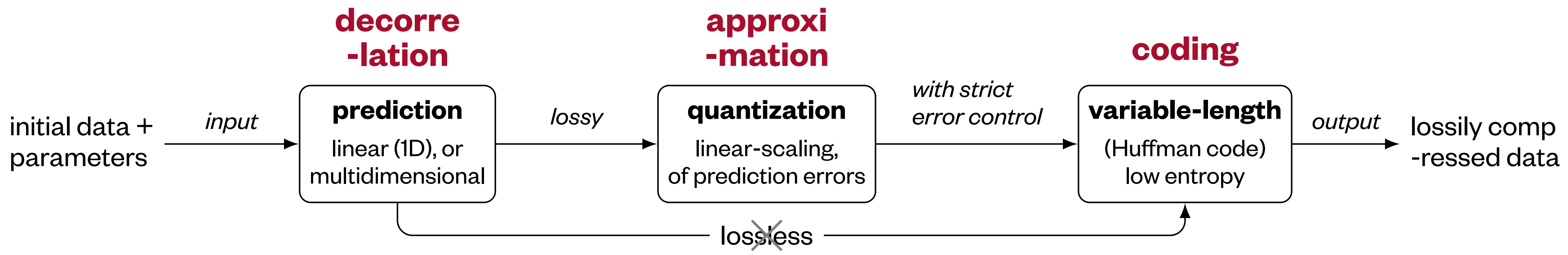
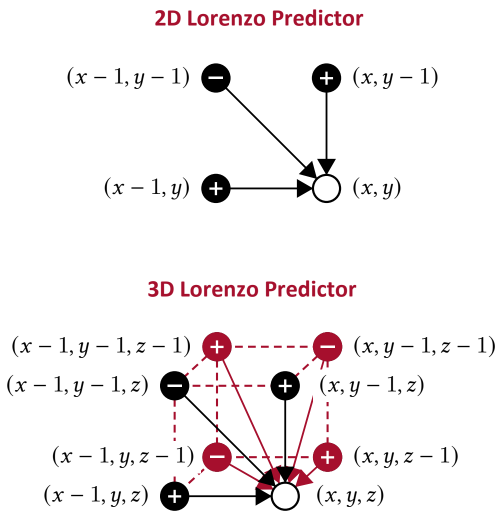


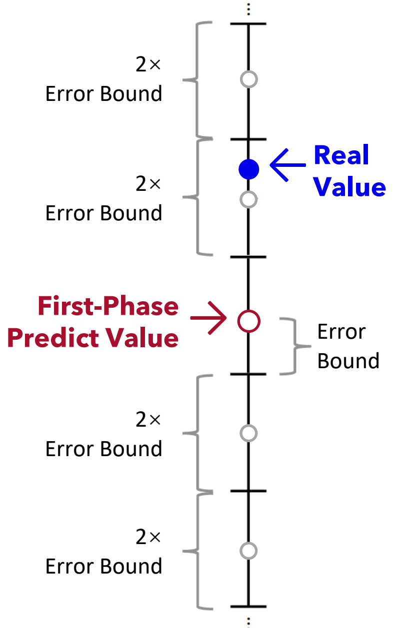
Figure from P. Lindstrom (LLNL)

Lossy compression for scientific data at varying reduction ratio (10:1 to 250:1, left to right)

SZ Compression Pipeline

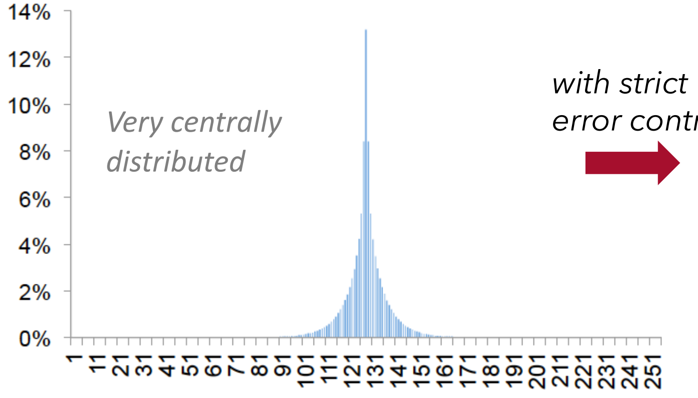






lossy

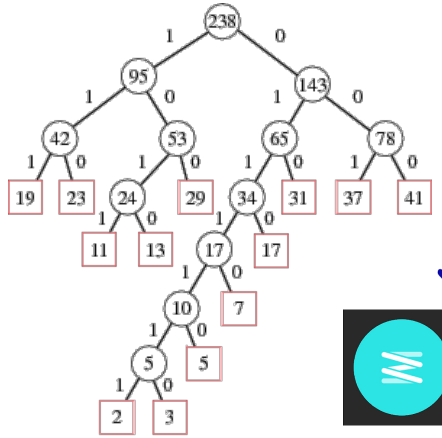
We may need much less than 256 intervals (that 8 bits can represent).



Error-Bounded Uniform Quantization Code

with strict error control

Huffman Coding



gzip

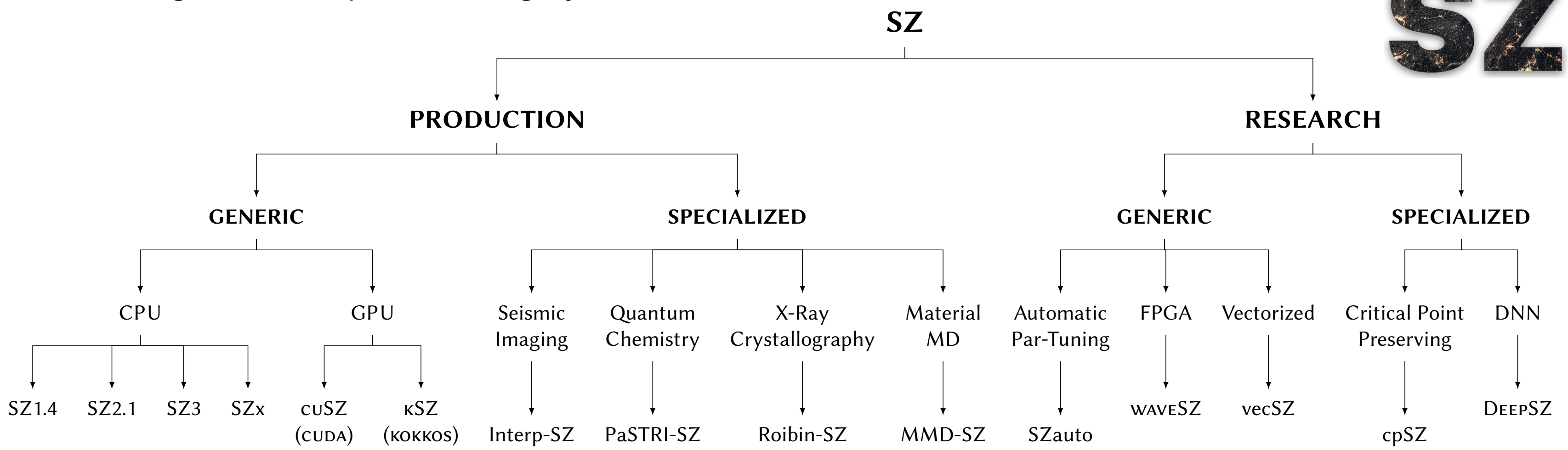
Zstandard

prediction **quantization** **coding**

$6M from DOE
$4M from NSF
$1M from Aramco

SZ: A Lossy Compression Framework for Scientific Data

Established in 1963, the R&D 100 Awards is the only S&T (science and technology) awards competition that recognizes new commercial products, technologies and materials for their technological significance that are available for sale or license. The R&D 100 Awards have long been a benchmark of excellence for industry sectors as diverse as telecommunications, high-energy physics, software, manufacturing, and biotechnology. This 2021 R&D 100 winner is listed below, along with its respective category.



SZ compression framework family tree.

Core R&D Team

Argonne National Laboratory
Dr. Franck Cappello, Dr. Sheng Di
Dr. Robert Underwood

Indiana University
Dr. Dingwen Tao, Jiannan Tian, Sian Jin
Chengming Zhang, Boyuan Zhang

Clemson University
Dr. Jon Calhoun, Griffin Dube

Others
Dr. Xin Liang (UKY), Dr. Kai Zhao (UAB)
Jinyang Liu (UCR), Cody Rivera (UIUC)

https://github.com/szcompressor

HPC use-cases:

- Reducing storage footprint
- Accelerating I/O & communication
- Accelerating visualization
- Reducing streaming intensity
- Running larger problems
- Checkpoint/restart

AI use-cases:

- DNN model compression
- DNN training data compression
- Reducing DNN memory consumption
- Accelerating distributed training
- …

H5-SZ Compression Filters

SZ

class **hdf5plugin.SZ**(*absolute=None, relative=None, pointwise_relative=None*)

`h5py.Group.create_dataset` 's compression arguments for using SZ filter.

It can be passed as keyword arguments:

```
f = h5py.File('test.h5', 'w')
f.create_dataset(
    'sz',
    data=numpy.random.random(100),
    **hdf5plugin.SZ())
f.close()
```

This filter provides different modes:

- **Absolute mode:** To use, set the `absolute` argument. It ensures that the resulting values will be within the provided absolute tolerance.

```
f.create_dataset(
    'sz_absolute',
    data=numpy.random.random(100),
    **hdf5plugin.SZ(absolute=0.1))
```

- **Relative mode:** To use, set the `relative` argument. It ensures that the resulting values will be within the provided relative tolerance. The tolerance will be computed by multiplying the provided argument by the range of the data values.

```
f.create_dataset(
    'sz_relative',
    data=numpy.random.random(100),
    **hdf5plugin.SZ(relative=0.01))
```

- **Point-wise relative mode:** To use, set the `pointwise_relative` argument. It ensures that each grid point of the resulting values will be within the provided relative tolerance.

```
f.create_dataset(
    'sz_pointwise_relative',
    data=numpy.random.random(100),
    **hdf5plugin.SZ(pointwise_relative=0.01))
```

For more details about the compressor SZ.

filter_id= *32017*

filter_name= *'sz'*

SZ3 is more modularized and composable, providing greater flexibility in configuring compression pipelines.

SZ3

class **hdf5plugin.SZ3**(*absolute=None, relative=None, norm2=None, peak_signal_to_noise_ratio=None*)

`h5py.Group.create_dataset` 's compression arguments for using SZ3 filter.

For more details about the compressor, see SZ3.

filter_id= *32024*

filter_name= *'sz3'*

Undergoing Projects

- **CSSI: Frameworks: FZ: A Fine-tunable Cyberinfrastructure Framework to Streamline Specialized Lossy Compression Development**

NEW!

  - **Goal**: To create a framework, called FZ, that revolutionizes the development of specialized lossy compressors by providing a comprehensive ecosystem to enable scientific users to intuitively research, compose, implement, and test specialized lossy compressors from a library of pre-developed, high-performance data reduction modules optimized for heterogeneous platforms.
  - **HDF5 Role**: The constructed compressor is instantiated as a dynamic library, which can be loaded by I/O libraries such as HDF5 through various languages, including C++ and Python.
- **CAREER: A Highly Effective, Usable, Performant, Scalable Data Reduction Framework for HPC Systems and Applications**
  - **Goal**: To research and develop novel algorithms and software to improve the efficacy, usability, performance, and scalability of data reduction for HPC systems and applications.
  - **HDF5 Role**: We offer a series of optimizations for compression coupled with parallel writing in HDF5 library for HPC applications.
- **CSSI: Elements: ROCCI: Cyberinfrastructure for In Situ Lossy Compression Optimization Based on Post Analysis Requirements**
  - **Goal**: To develop a requirement-oriented compression cyberinfrastructure (ROCCI) for data-intensive domains, which can select and run the best fit lossy compressor automatically at runtime, in terms of user's requirement on their post hoc analysis.
  - **HDF5 Role**: ROCCI provides a series of functions that transparently configure compression parameters in the HDF5 environment.

CSSI: Frameworks: FZ: A Fine-tunable Cyberinfrastructure Framework to Streamline Specialized Lossy Compression Development

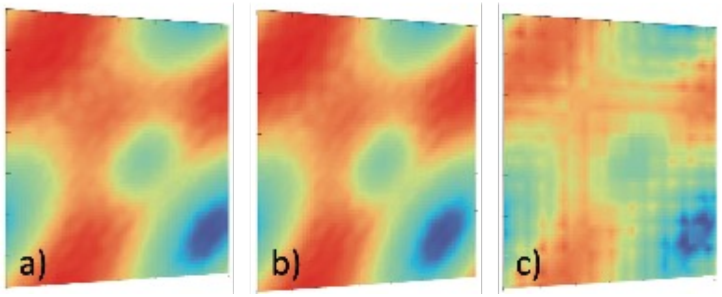
Franck Cappello, Sheng Di, University of Chicago [Award #2311875]
Dingwen Tao, Indiana University [Award #2311876]
Hanqi Guo, Ohio State University [Award #2311877]
Kai Zhao, Florida State University [Award #2311878]

Summary:
This project aims to create a framework, called FZ, that revolutionizes the development of specialized lossy compressors by providing a comprehensive ecosystem to enable scientific users to intuitively research, compose, implement, and test specialized lossy compressors from a library of pre-developed, high-performance data reduction modules optimized for heterogeneous platforms.

Approach:
This project builds FZ by adapting, combining, and extending multiple existing capabilities from SZ lossy compressor, LibPressio unifying compression interface, OptZConfig optimizer of compressor configurations, Z-checker and QCAT compression quality analysis tools, and Paraview and VTK visualization tools. Specifically, it builds three main components:

- **Programming interfaces and compressor generator**: create new compressors from high-level languages such as Python and optimize their execution.
- **New compression modules**: Refactor SZ lossy compressors to enable fine-grained composability of a large diversity of data transformation modules and integrate non-uniform compression capabilities, new preprocessing, decorrelation, approximation, and entropy coding modules.
- **Interactive visualization, quality assessment, and GUI tools**: adapt and extend existing capabilities to automatically search optimized lossy compression module compositions and to identify relevant compression ratio, speed, and quality trade-offs for their use cases.

a) Original data from QMCPACK. b) Specialized compressor (CR=54). c) Generic compressor (CR=27).



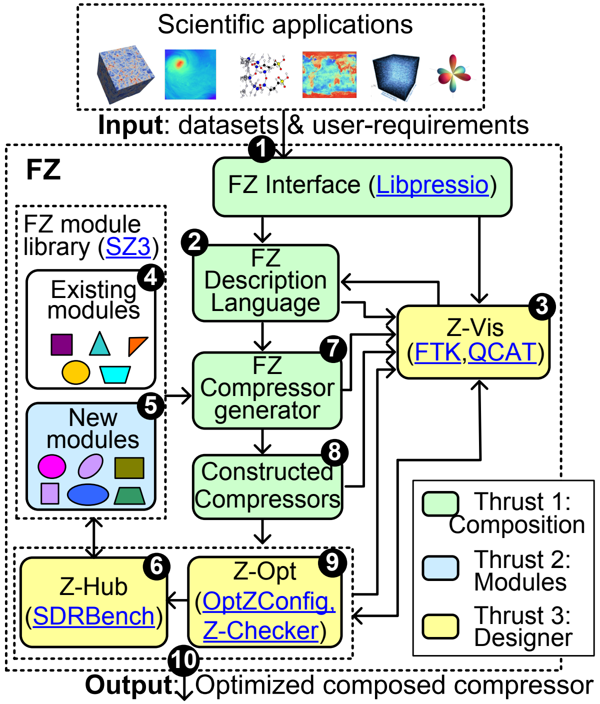


Fig. FZ design overview.

Compression Benchmark

FCBench: 8 CPU-based and 5 GPU-based compression methods on 33 real-world datasets assembled in the

FCBench: Cross-Domain Benchmarking of Lossless Compression for Floating-point Data: Uniting HPC and Database Communities

ABSTRACT

While both the database and high-performance computing (HPC) communities utilize lossless compression methods to minimize floating-point data size, a disconnect persists between them. Each community designs and assesses methods in a domain-specific manner, making it unclear if HPC compression techniques can benefit database applications, or vice versa. With the HPC community increasingly leaning towards in-situ analysis and visualization, more floating-point data from scientific simulations are being stored in databases like Key-Value Stores [73], and queried using in-memory retrieval paradigms. This trend underscores the urgent need for a collective study of these compression methods' strengths and limitations, based on a broad array of data from various domains. In our study, we extensively evaluate the general and database performance of eight CPU-based and five GPU-based compression methods developed by both communities, using 33 real-world datasets assembled in the Floating-point Compressor Benchmark (FCBench). Our goal is to offer insights on these compression methods that could assist researchers in selecting existing methods or developing new ones for integrated database and HPC applications.

1 INTRODUCTION

Floating-point data is widely used in various domains, such as scientific simulations, geospatial analysis, and medical imaging [16, 23, 58]. As the scale of these applications increases, compressing floating-point data can help reduce data storage and communication overhead, thereby improving performance [56].

Why lossless compression? Using a fixed number of bits (e.g., 32 bits for single-precision data) to represent real numbers often results in rounding errors in floating-point calculations [18]. Consequently, system designers favor using the highest available precision to minimize the problems caused by rounding errors [57]. Similarly, due to concerns about data precision, lossless compression is preferred over lossy compression, even with lower compression ratios, when information loss is not tolerable.

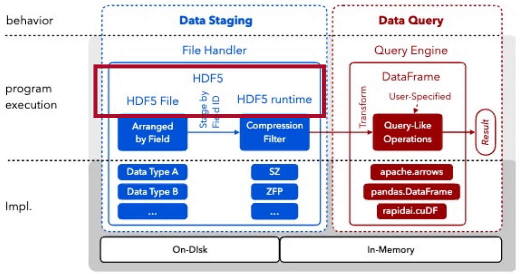


Figure 1: Integrating HPC and database with HDF5 and Dataframes.

1.1 Study Motivation

Both the HPC and database communities have developed lossless compression methods for floating-point data. However, there are fundamental differences between the floating-point data of these two domains. Typically, numeric values stored in database systems are not necessarily in order. To the best of our knowledge, the compression methods for floating-point data developed by the database community are specialized for time-series data. On the other hand, HPC systems deal with structured high-dimensional floating-point data from scientific simulations or observation devices, such as satellites and telescopes. In other words, both communities have developed floating-point data compression methods, but for different data from their respective domains. Therefore, an interesting question is *whether the compression methods developed in one community can work on the data from the other community or vice versa.*

Answering this question becomes urgent due to the trend of increasingly integrated HPC and database systems. For example, the Key-Value format and Map/Reduce paradigm have established many successful Big Data applications [49]. Due to the fine-grained data access ability of Key-Value Stores (KVS), HPC systems have embraced them not only for resource management [73] but also for in-situ analysis and visualization [8, 13, 20, 35]. We see emerging tools like Seer-Dash [21] that use Mochi's Key-Value storage [55]

FCBench Public

Watch 1

main · 1 branch · 0 tags

Go to file · Add file · Code

papersub2023 Update README for evaluate and compare — 6729e94 3 weeks ago — 14 commits

code	integrated evaluation and comparison	3 weeks ago
scripts	integrated evaluation and comparison	3 weeks ago
README.md	evaluate/compare README	last month

README.md

FCbench: Cross-Domain Benchmarking of Lossless Compression for Floating-point Data: Uniting HPC and Database Communities

We benchmarked eight CPU-based and five GPU-based lossless compression methods on 33 datasets from scientific simulation, time series, observation and database transactions domains.

Test system

The experiments are carried out on a Chameleon Cloud compute node with 2 Intel(R) Xeon(R) Gold 6126 CPUs, 2.60GHz, 187 GB RAM. The compute node also has 1 Nvidia Quadro RTX 6000GPU with 24 GB GPU Memory. The node compilers are GCC/G++9.4, CUDA 11.3, CMAKE 3.25.0 and python 3.8

- setup directories

```
mkdir code data experiments output software
```

Selected Publications *(with my students underlined)*

Efficient Data Reduction Techniques for HPC Applications

Generic

- [**IPDPS'17**] *Significantly Improving Lossy Comp. for Scientific Data Based on Multidimensional Prediction and Error-Controlled Quantization*. **D. Tao**, et al.
- [**TPDS'19**] *Optimizing Lossy Compression Rate-Distortion for Automatic Online Selection between SZ and ZFP.* **D. Tao**, *et al.*
- [**ICDE'22**] *Significantly Improving Prediction-Based Lossy Compression Via Ratio-Quality Modeling*. <u>S. Jin</u>, *et al.*

Domain Specific

- [**CLUSTER'18**] *Error-Bounded Lossy Compression for Two-Electron Integrals in Quantum Chemistry*. (**Best Paper Award**) A. Gok, **D. Tao**, *et al.*
- [**IPDPS'20**] *Understanding GPU-Based Lossy Compression for Extreme-Scale Cosmological Simulations*. <u>S. Jin</u>, *et al.*
- [**HPDC'21**] *Adaptive Configuration of Lossy Compression for Cosmology Simulations via Fine-Grained Rate-Quality Modeling*. <u>S. Jin</u>, *et al.*
- [**HPDC'22**] *TAC: Optimizing Error-Bounded Lossy Compression for Three-Dimensional Adaptive Mesh Refinement Simulations*. <u>D. Wang</u>, *et al.*

GPU/FPGA Acceleration

- [**PPoPP'20**] *waveSZ: A Hardware-Algorithm Co-Design of Efficient Lossy Compression for Scientific Data.* <u>J. Tian</u>, *et al.*
- [**PACT'20**] *cuSZ: An Efficient GPU Based Error-Bounded Lossy Compression Framework for Scientific Data*. <u>J. Tian</u>, *et al.*
- [**IPDPS'21**] *Revisiting Huffman Coding: Toward Extreme Performance on Modern GPU Architectures*. <u>J. Tian</u>, *et al.*
- [**IPDPS'22**] *Optimizing Huffman Decoding for Error-Bounded Lossy Compression on GPUs.* <u>C. Rivera</u>, *et al.*
- [**ICS'23**] *GPULZ: Optimizing LZSS Lossless Compression for Multi-byte Data on Modern GPUs.* <u>B. Zhang</u>, *et al.*
- [**HPDC'23**] *FZ-GPU: A Fast and High-Ratio Lossy Compressor for Scientific Computing Applications on GPUs.* <u>B. Zhang</u>, *et al.*

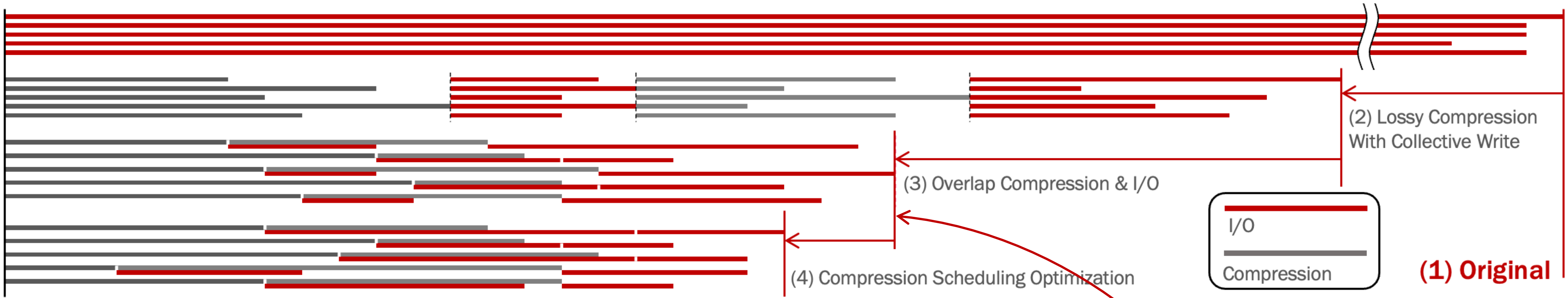
Compression-accelerated Communication and I/O in HPC Systems

- [**ICS'22**] *CEAZ: Accelerating Parallel I/O via Hardware-Algorithm Co-Designed Adaptive Lossy Compression.* <u>C. Zhang</u>, *et al.*
- [**PACT'22**] *HBMax: Optimizing Memory Efficiency for Parallel Influence Maximization on Multicore Architectures*. <u>X. Chen</u>, *et al.*
- [**SC'22**] *Accelerating Parallel Write via Deeply Integrating Predictive Lossy Compression with HDF5.* <u>S. Jin</u>, **D. Tao**, *et al.*
- [**SC'23**] *AMRIC: A Novel In Situ Lossy Compression Framework for Efficient I/O in Adaptive Mesh Refinement Applications*. <u>D. Wang</u>, et al.

High-Performance Deep Learning Training and Inference Systems

- [**HPDC'19**] *DeepSZ: A Novel Framework to Compress Deep Neural Networks by Using Error-Bounded Lossy Compression*. <u>S. Jin</u>, *et al.*
- [**DAC'20**] *RTMobile: Beyond Real-Time Mobile Acceleration of RNN for Speech Recognition.* P. Dong, *et al.*
- [**FPL'22**] *H-GCN: A Graph Convolutional Network Accelerator on Versal ACAP Architecture.* <u>C. Zhang</u>, *et al.*
- [**ICS'21**] *ClickTrain: Efficient and Accurate End-to-End Deep Learning Training via Fine-Grained Architecture-Preserving Pruning*. <u>C. Zhang</u>, *et al.*
- [**VLDB'22**] *COMET: A Novel Memory-Efficient Deep Learning Training Framework by Using Error-Bounded Lossy Compression*. <u>S. Jin</u>, *et al.*
- [**ICS'23**] *HEAT: A Highly Efficient and Affordable Training System for Collaborative Filtering Based Recommendation on CPUs.* <u>C. Zhang</u>, *et al.*
- [**PPoPP'24**] *SOLAR: A High-Performance Data Loading Framework for Distributed DNN Training with Large Datasets*. <u>B. Sun</u>, *et al.*

SC'22: Accelerating Parallel Write via Lossy Compression with HDF5



Scientific Achievement:

A **parallel write solution** that integrates predictive lossy compression with the asynchronous I/O feature in HDF5, which **overlaps I/O latency with compression**.

This requires a compression ratio prediction!
- Estimate the offset of write operation

Write operation can only be initiated after the offset been assigned for all data blocks

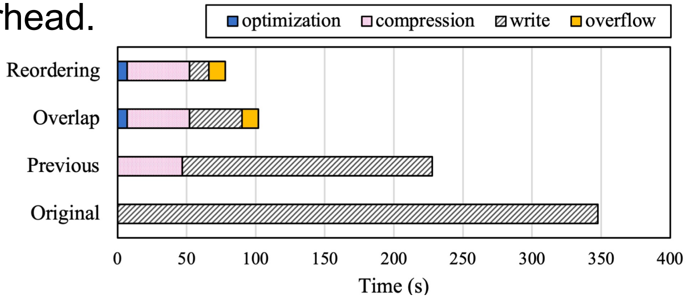
Significance and Impact:

- Evaluate on real-world scientific applications, Nyx and VPIC, with up to 4096 cores on Summit.
- Improve the parallel-write performance by up to **4.5×** and **2.9×** compared to the HDF5 write without compression and with the SZ lossy compression filter, respectively, with only **1.5%** storage overhead.

Research Details:

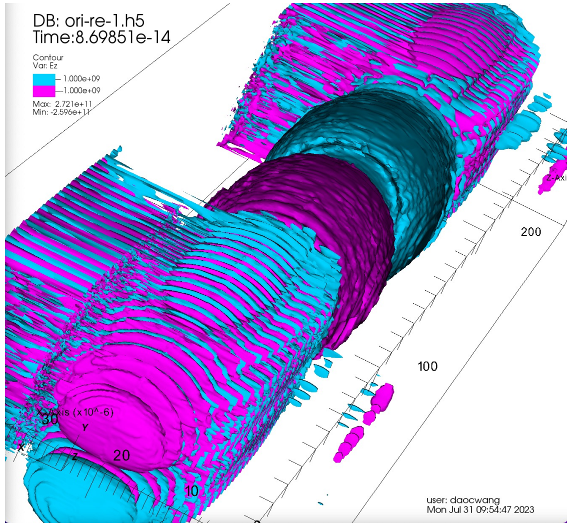
- Extend the prediction model to **estimate the offset** and performance of parallel I/O.
- **Overlap** I/O with compression.
- Optimization for **reorder compression tasks** to achieve higher performance.

Performance comparison ↓

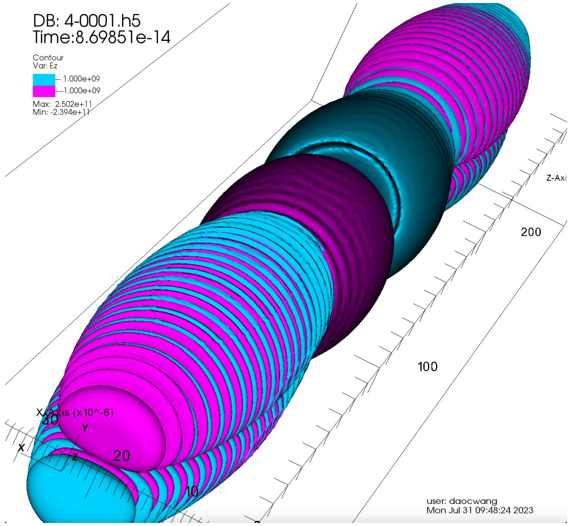


SC’23: In Situ Lossy Compression for Fast I/O in AMR Applications

Original AMReX
Comp ratio = **19.0**



Our AMRIC
Comp ratio = **23.9**

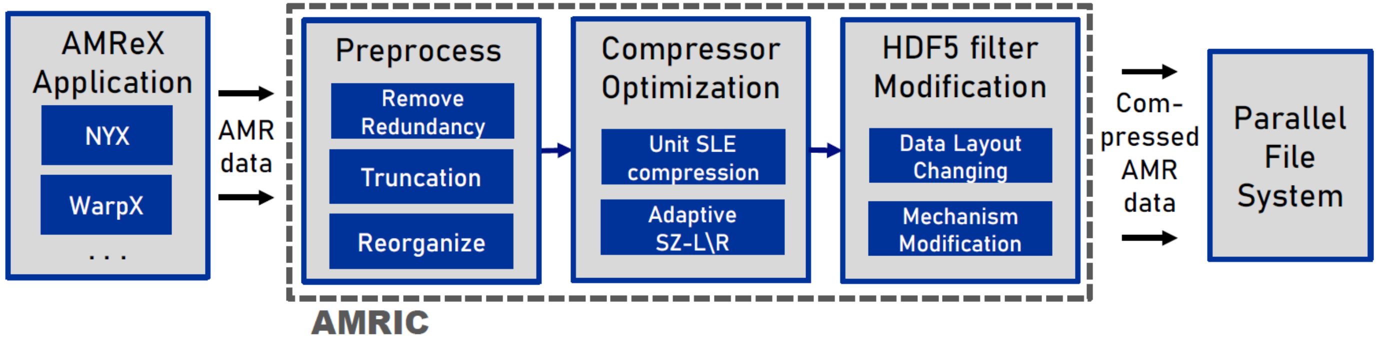


Scientific Achievement:

AMRIC is the first in situ AMR data compression framework that improve both **I/O costs** and boost **compression quality** for AMR applications.

Significance and Impact:

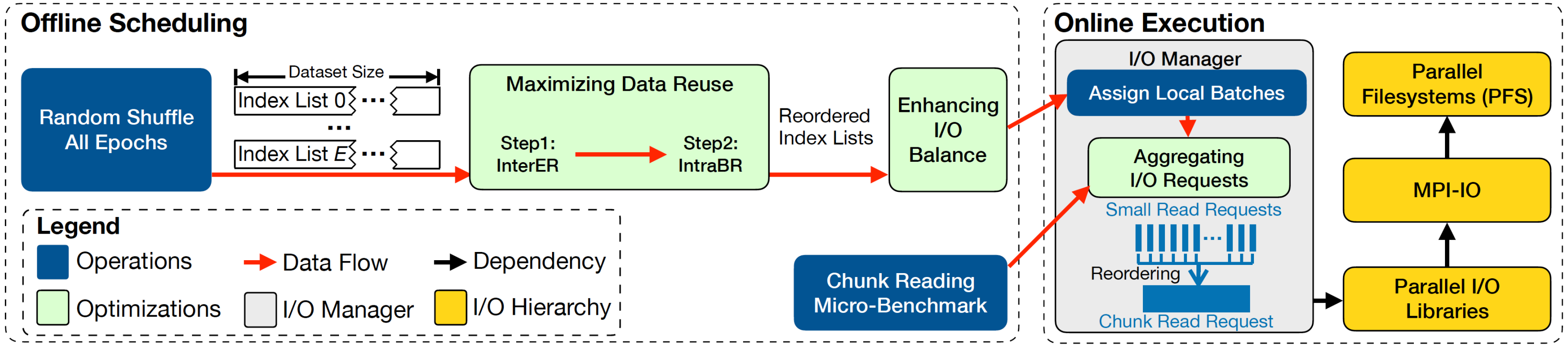
- Evaluate AMRIC on two real-world AMReX applications, WarpX and Nyx, with 4096 cores from Summit.
- AMRIC achieves up to **10.5×** I/O performance improvement over the non-compression solution.
- Up to **39×** I/O and **81×** CR improvement with better data quality over AMReX’s original solution.



Research Details:

- Propose a compression-oriented in situ **pre-processing** workflow for AMR data
- Optimize the state-of-the-art SZ lossy **compressor’s efficiency** on AMR data
- Overcome the gap between the HDF5 and AMR applications by **modifying HDF5 filter**
  - Allowing the use of larger HDF5 chuck size which **benefit compression** ratio and throughput

PPoPP'24: Enhance I/O Performance in Distributed DNN Training



With Parallel HDF5

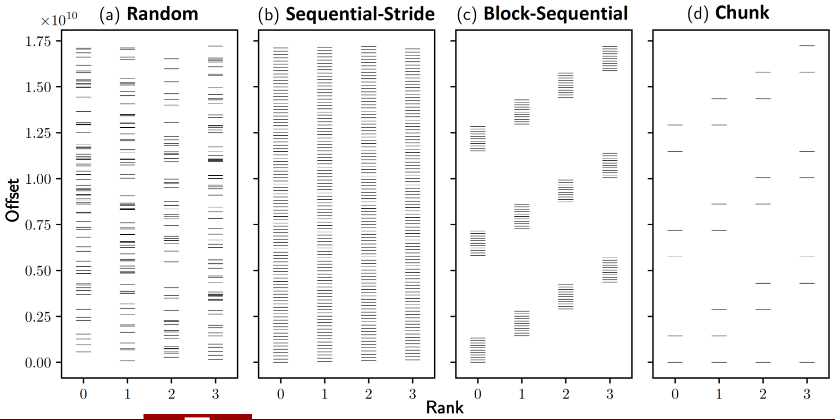
- SOLAR determines read chunk size, which two adjacent data samples in shuffled index list worth loading in a chunk.
- SOLAR aggregates small read requests into a larger chunk read request.

Research Details

- Propose SOLAR, a framework that enhances I/O performance for distributed training.
- Design three optimizations to **maximize data reuse** (reorder epoch order & samples in global batch), achieve **I/O workload balance**, and optimize data access pattern with **parallel HDF5**.
- Analyze that SOLAR has small impact on test accuracy.

Significance and Impact

- Up to a **10.9×** increase in I/O speed and a **5.9×** improvement in overall training performance over a production-level framework.
- Outperforms state-of-the-art approaches with up to a **3.5×** speedup in I/O.



Access Pattern Observations

Unlike HPC applications, distributed training has **random, non-consecutive** access pattern.

Pattern	Time	Norm'ed	Speedup	
Random Access	645.864	203.42×	1.00×	⟵distributed training access pattern;
Sequential-stride Access	84.421	26.59×	7.65×	⟵inconsecutive access pattern;
Block Sequential Access	30.537	9.62×	21.15×	⟵consecutive access pattern;
Chunk access	3.175	1.00×	203.42×	⟵chunk reading access pattern;

w/o random shuffle!