Revolutionizing I/O Performance: Lossy Compression Meets HDF5

Dingwen Tao, Associate Professor
Indiana University Bloomington
### Storage and I/O Issues in HPC Systems

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

<table>
<thead>
<tr>
<th>Supercomputer</th>
<th>Year</th>
<th>Class</th>
<th>Peak FLOPS (PF)</th>
<th>Memory Size (MS)</th>
<th>Storage Band (SB)</th>
<th>MS/SB</th>
<th>PF/SB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray Jaguar</td>
<td>2008</td>
<td>1 PFLOPS</td>
<td>1.75 PFLOPS</td>
<td>360 TB</td>
<td>240 GB/s</td>
<td>1.5k</td>
<td>7.3k</td>
</tr>
<tr>
<td>Cray Blue Waters</td>
<td>2012</td>
<td>10 PFLOPS</td>
<td>13.3 PFLOPS</td>
<td>1.5 PB</td>
<td>1.1 TB/s</td>
<td>1.3k</td>
<td>13.3k</td>
</tr>
<tr>
<td>Cray Cori</td>
<td>2017</td>
<td>10 PFLOPS</td>
<td>30 PFLOPS</td>
<td>1.4 PB</td>
<td>1.7 TB/s</td>
<td>0.8k</td>
<td>17k</td>
</tr>
<tr>
<td>IBM Summit</td>
<td>2018</td>
<td>100 PFLOPS</td>
<td>200 PFLOPS</td>
<td>&gt; 10 PB</td>
<td>2.5 TB/s</td>
<td>&gt; 4k</td>
<td>80k</td>
</tr>
</tbody>
</table>

(*) when using burst buffer
(**) counting only DDR4

---

<table>
<thead>
<tr>
<th>Supercomputer</th>
<th>Year</th>
<th>Class</th>
<th>Peak FLOPS (PF)</th>
<th>Memory Size (MS)</th>
<th>Storage Band (SB)</th>
<th>MS/SB</th>
<th>PF/SB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fujitsu Fugaku</td>
<td>2020</td>
<td>“Exascale”</td>
<td>537 PFLOPS</td>
<td>4.85 PB</td>
<td>&gt; 1.5 TB/s</td>
<td>&gt; 3.23k</td>
<td>358k</td>
</tr>
<tr>
<td>AMD Frontier</td>
<td>2021</td>
<td>Exascale</td>
<td>1.6 EFLOPS</td>
<td>9.2 PB</td>
<td>&gt; 10 TB/s</td>
<td>&gt; 0.92k</td>
<td>160k</td>
</tr>
<tr>
<td>Intel Aurora (#)</td>
<td>future</td>
<td>Exascale</td>
<td>&gt; 2 EFLOPS</td>
<td>&gt; 10 PB</td>
<td>&gt;= 25 TB/s</td>
<td>&gt; 0.40k</td>
<td>80k</td>
</tr>
</tbody>
</table>

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

(**) DDN Newsroom

---

*(OLC: IBM/NVIDIA Summit, OLCF: AMD Frontier, ALCF: Intel Aurora)*
# Data Management Issues for Scientific Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Data Scale</th>
<th>Bottleneck</th>
<th>Reduce by</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HACC</strong></td>
<td>20 PB</td>
<td>use up filesystem (26 PB in total)</td>
<td>10× in need</td>
</tr>
<tr>
<td>cosmology simulation</td>
<td>one-trillion-particle</td>
<td>Mira@ANL</td>
<td></td>
</tr>
<tr>
<td><strong>CESM</strong></td>
<td>50% vs 20%</td>
<td>5h30m to store</td>
<td>10× in need</td>
</tr>
<tr>
<td>climate simulation</td>
<td>storage in hardware</td>
<td>NSF Blue Waters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>budget, 2017 vs 2013</td>
<td>1-TBps I/O</td>
<td></td>
</tr>
<tr>
<td><strong>APS-U</strong></td>
<td>$10^2$ PB</td>
<td>saturate connection</td>
<td>100× in need</td>
</tr>
<tr>
<td>High-Energy X-Ray Experiments</td>
<td>Brain Initiatives</td>
<td>100 GBps bandwidth</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The images illustrate the scale and complexity of data management in scientific applications.
Our Solution – Error-Bounded Lossy Compression

**2:1** (FP-type)  **10:1** or higher

<table>
<thead>
<tr>
<th>Lossless on scientific datasets</th>
<th>Reduction ratio in need</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>industry</strong></td>
<td>High in reduction rate,</td>
</tr>
<tr>
<td><strong>lossy compressor (JPEG)</strong></td>
<td>but <strong>not</strong> suitable for <strong>HPC</strong></td>
</tr>
<tr>
<td>Need diverse compression modes</td>
<td>1) absolute error bound (infinity-norm)</td>
</tr>
<tr>
<td></td>
<td>2) pointwise relative error bound</td>
</tr>
<tr>
<td></td>
<td>3) RMSE error bound (2-norm)</td>
</tr>
<tr>
<td></td>
<td>4) fixed bitrate</td>
</tr>
<tr>
<td></td>
<td>5) satisfying post-analysis requirements</td>
</tr>
</tbody>
</table>

**SZ**


 Floating point data set (numerical simulation of the brain):

- Mantissa
- Sign
- Exponent

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

Source: Leonardo Bautista Gomez (BSC)

> 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)
SZ Compression Pipeline

- **prediction**: initial data + parameters → prediction (linear (1D), or multidimensional)
- **quantization**: linear-scaling, of prediction errors
- **coding**: variable-length (Huffman code) low entropy

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

Huffman Coding

Error-Bounded Uniform Quantization Code

very centrally distributed with strict error control

---

INDIANA UNIVERSITY BLOOMINGTON
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.

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)
```

It can be passed as keyword arguments:

```
f = hSpy.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))
```

**SZ3**

```
class hdf5plugin_SZ3(absolute=None, relative=None, norm2=None, peak_signal_to_noise_ratio=None)
```

For more details about the compressor **SZ**.

- **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, see **SZ3**.

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

For more details about the compressor, see **SZ3**.
Undergoing Projects

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

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.
Compression Benchmark

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.

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

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 necessary 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 MapReduce paradigms 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 are emerging tools like See-Dash [21] that use Mochi’s Key-Value storage [53]...
Efficient Data Reduction Techniques for HPC Applications

- [TPDS'19] Optimizing Lossy Compression Rate-Distortion for Automatic Online Selection between SZ and ZFP. D. Tao, et al.
- [CLUSTER'18] Error-Bounded Lossy Compression for Two-Electron Integrals in Quantum Chemistry. (Best Paper Award) A. Gok, D. Tao, et al.

Compression-accelerated Communication and I/O in HPC Systems

- [PACT'22] HBMax: Optimizing Memory Efficiency for Parallel Influence Maximization on Multicore Architectures. X. Chen, et al.

High-Performance Deep Learning Training and Inference Systems


Selected Publications (with my students underlined)

Generic

- [CLUSTER'18] Error-Bounded Lossy Compression for Two-Electron Integrals in Quantum Chemistry. (Best Paper Award) A. Gok, D. Tao, et al.

Domain Specific


GPU/FPGA Acceleration

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.

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 re-order compression tasks to achieve higher performance.

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

Performance comparison ↓

<table>
<thead>
<tr>
<th>Reordering</th>
<th>Overlap</th>
<th>Previous</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>120</td>
<td>150</td>
<td>200</td>
</tr>
<tr>
<td>200</td>
<td>250</td>
<td>250</td>
<td>300</td>
</tr>
<tr>
<td>300</td>
<td>350</td>
<td>350</td>
<td>400</td>
</tr>
</tbody>
</table>

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

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 $10.5 \times$ I/O performance improvement over the non-compression solution.
- Up to $39 \times$ I/O and $81 \times$ 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**

**Access Pattern Observations**

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

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Time</th>
<th>Norm’ed</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Access</td>
<td>645.864</td>
<td>203.42×</td>
<td>1.00× ← distributed training access pattern;</td>
</tr>
<tr>
<td>Sequential-stride Access</td>
<td>84.421</td>
<td>26.59×</td>
<td>7.65× ← inconsecutive access pattern;</td>
</tr>
<tr>
<td>Block Sequential Access</td>
<td>30.537</td>
<td>9.62×</td>
<td>21.15× ← consecutive access pattern;</td>
</tr>
<tr>
<td>Chunk access</td>
<td>3.175</td>
<td>1.00×</td>
<td>203.42× ← chunk reading access pattern;</td>
</tr>
</tbody>
</table>

**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.

**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.

**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.

**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.