H5Zsci.io

A Proposal for HDF5 Scientific Data Compression Community Benchmarking

Mark C. Miller

Outline: What is H5Zbm.io

- Benchmark data sources
- An HDF5 command-line (e.g. raw data in/hdf5 file out) compression plugin tool
- Standard (command-line) compression tool sources (zip, gzip, xz, etc.)
- Benchmarking scripts comparing (time/space) of HDF5 plugins to standard tools
  - Including variations in relevant HDF5 features (e.g. chunk cache, partial I/O, etc.)
  - Space performance includes both memory space and final file size
- Documentation and examples of HDF5 compression usage
- Web site with published benchmarking tables
- Yearly “releases” coinciding with HDF5 releases

Goals of H5Zbm.io

- Keep community informed of compression capabilities
  - Performance
  - Best practices
  - Documentation
  - Examples
- Ensure performance of HDF5 library and compression features

Benchmark Data Sources: Idea 1

Scientific Data Reduction Benchmarks

This site has been established as part of the ECP CODAR project.

This site provides reference scientific datasets, data reduction techniques, error metrics, error controls and error assessment tools for users and developers of scientific data reduction techniques.

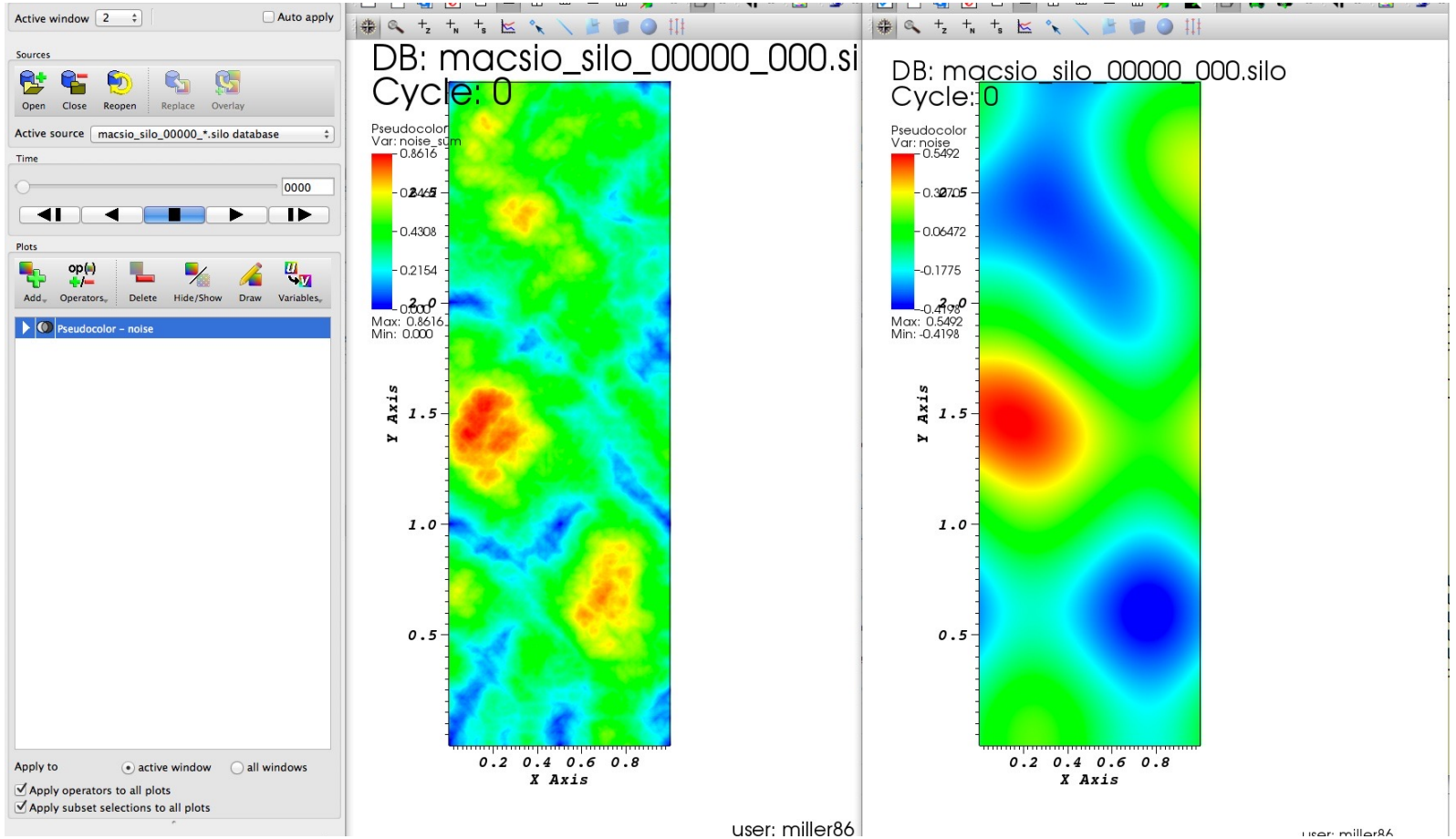
***Important: when publishing results from one or more datasets presented in this webpage, please*:**

- ***Cite**:* SDRBench: *https://sdrbench.github.io*
- ***Please also cite**:* K. Zhao, S. Di, X. Liang, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors", International Workshop on Big Data Reduction (IWBDR2020), in conjunction with IEEE Bigdata20.
- ***Acknowledge***: the source of the dataset you used, the DOE NNSA ECP project, and the ECP CODAR project.
- ***Check**:* the condition of publications (some dataset sources request prior check)
- ***Contact**:* the compressor authors to get the correct compressor configuration according to each dataset and each comparison metrics.
- ***Dimension***: the order of the dimensions shown in the 'Format' column of the table is in row-major order (aka. C order), which is consistent with well-known I/O libraries such as HDF5. For example, for the CESM-ATM dataset (1800 x 3600), 1800 is higher dimension (changing slower) and 3600 is lower dimension (changing faster). For most compressors (such as SZ, ZFP and FPZIP), the dimensions should be given in the reverse order (such as -2 3600 1800) for their executables. If you are not sure about the order of dimension, one simple method is trying different dimension orders and selecting the results with highest compression ratios.

Data sets:

Name	Type	Format	Size (data)	Command Examples	Link
CESM-ATM *Source:* *Mark Taylor (SNL)*	Climate simulation	Dataset1 : 79 fields: 2D, 1800 x 3600 ; Dataset2 : 1 field : 3D, 26x1800x3600. Both are single precision, binary 8 bit Entropy / 32 bit Entropy: avg: 6.5 / 19.8 min: 1.5 / 2.3 max: 7.8 / 22.6	Dataset1 (raw): 1.47GB Dataset1 (cleared): 1.47GB Dataset2: 17GB (cleared data zeroed all background data)	**SZ(Compress)**: sz -z -f -i CLDHGH_1_1800_3600.f32 -M REL -R 1E-2 -2 3600 1800 **SZ(Decompress)**: sz -x -f -i CLDHGH_1_1800_3600.f32 -2 3600 1800 -s CLDHGH_1_1800_3600.f32.sz -a **ZFP**: zfp -f -i CLDHGH_1_1800_3600.f32 -z CLDHGH_1_1800_3600.f32.zfp -o CLDHGH_1_1800_3600.f32.zfp.out -2 3600 1800 -a 1E-2 -s **LibPressio:** pressio -b compressor=$COMP -i CLDHGH_1_1800_3600.f32 -d 3600 -d 1800 -t float -o rel=1e-2 -m time -m size -M all where $COMP can be sz, zfp, sz3, mgard, etc… **Z-checker-installer**: ./runZCCase.sh -f REL CESM-ATM raw-data-dir f32 3600 1800	Dataset1 (raw) Dataset1 (cleared) Dataset2 (raw) Metadata Dataset1's property Dataset2's property
EXAALT *Source:*	Molecular dynamics	6 fields: x,y,z,vx,vy,vz, Each field stored separately	Dataset1:	**SZ(Compress)**: sz -z -f -i xx.f32 -M REL -2 -1 2869440	Dataset1 Metadata1

Benchmark Data Sources: Idea 2 (MACSio data generation feature w/Perlin noise)



Benchmark Data: Things to consider

- Size
- Data types
- Dimensionality
- Noise and/or smoothness properties and over which dimensions
- File system being tested
- Parallelism

HDF5 Command-line compression plugin tool (Like gzip/gunzip command-line tools but for HDF5 files)

For writes

- Reads **raw binary data** file into memory (e.g. foo.dat)
- Accepts command-line arguments specifying…
  - File name containing raw binary data
  - Input data file type and dimensionality
  - Plugin (name or id)
  - Plugin-specific options/params
  - Partial I/O params (optional)
  - HDF5 lib relevant feature params (chunk sizing, cache sizing, data type, VFD, lib version, etc.)
- Sets up/performs H5Dwrite() with data read from file
- Reports time/space performance on stdout
  - Initial read and memory for raw data factored out

For reads

- Opens hdf5 file (from writer) for reading
- Accepts command-line arguments specifying…
  - File name containing raw binary data
  - Input data file type and dimensionality
  - Plugin (name or id)
  - Plugin-specific options/params
  - Partial I/O params (optional)
  - HDF5 lib relevant feature params (chunk sizing, cache sizing)
- Sets up/performs H5Dread()
- Reports time/space performance on stdout

Standard command-line compression tools to compare against

- For example…
  - gzip, gunzip
  - xz, unxz
  - zip, unzip
  - bzip, bunzip
- Assume available on benchmarking system or build from sources
- Run and get time/space performance on the same raw binary file
- Report HDF5 performance as a proportion of these tools
  - Include final file size too

Benchmarking scripts

- Performs a whole suite of benchmarking runs gathering data
  - Probing relevant dimensions of the performance space
  - Versions of HDF5 API/File format (H5Pset_libver_bounds())
- Packages up results into json or yaml file
- Submit PR with machine/config info to H5Zbm.io GitHub repo

Documentation

- Document common use cases
- Document common pitfalls and how to mitigate
- Refer to sections of HDF5 Reference manual for relevant props

Web Site With Published Results

- Kinda sorta like a CDash thingy (maybe really via CDash)
- Tables and plots
- Important notices regarding issues in HDF5 releases (if any needed)
- Yearly releases/updates

Other things to consider

- Time and Space performance
- Compress / Decompress performance
- CPU Architectures???
- HDF5 versions (compile-time selection)
- GPU / CPU
- Threaded compression plugins and threaded command-line tools
- How to characterize HDF5 performance relative to “best” native tool