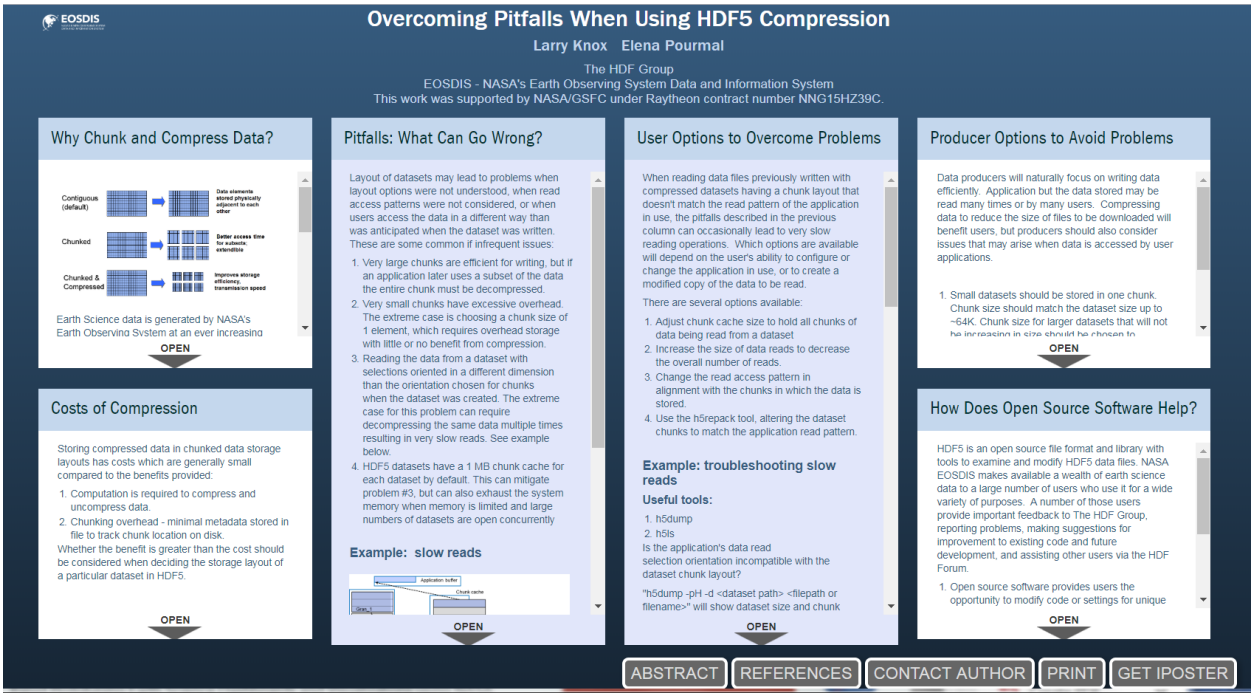
Overcoming Pitfalls When Using HDF5 Compression

Larry Knox | Elena Pourmal

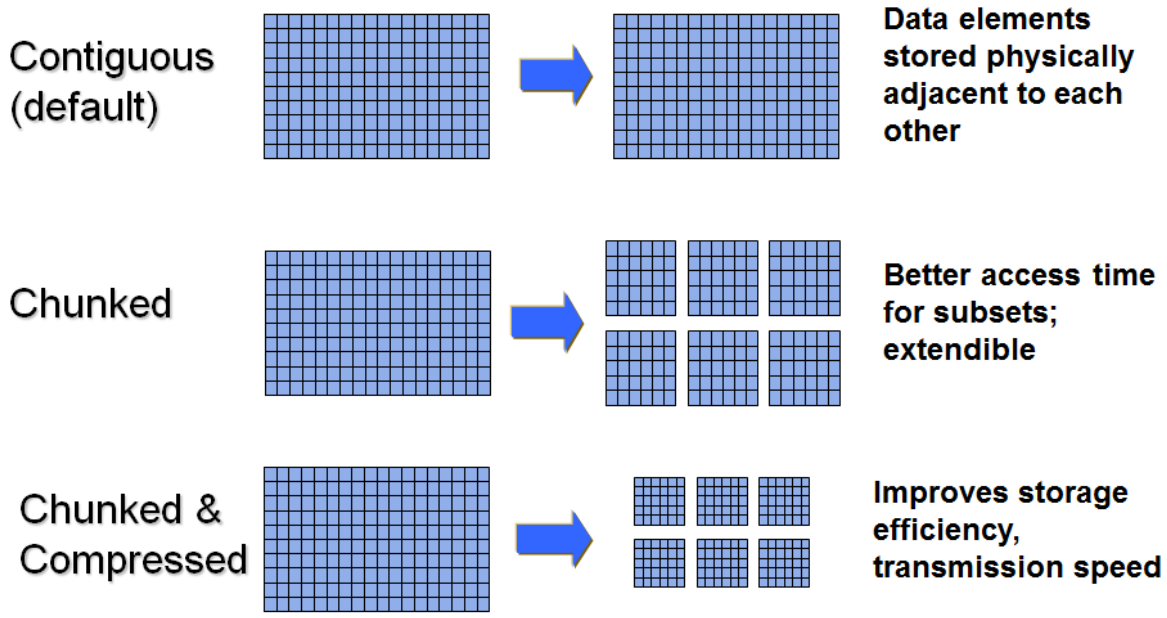
The HDF Group | NASA | EED2

This document has the contents of an eLightning poster for AGU 2018 Fall Meeting. Screenshot:



Link: https://agu2018fallmeeting-agu.ipostersessions.com/default.aspx?s=92-39-21-91-C3-AD-9B-39-82-50-5A-9B-90-F4-44-A5

Why Chunk and Compress Data?



Earth Science data is generated by NASA's Earth Observing System at an ever increasing rate, which in turn increases the challenge of storing, transferring and processing that data. The HDF5 file format and library provide flexibility to use a variety of data compression filters on individual datasets in an HDF5 file. Compressed data is stored in chunks and automatically uncompressed by the library and filter plugin when a chunk is accessed. Utilizing chunking and compression can provide these benefits:

1. Required storage space is reduced.
2. I/O time is reduced because the size of the data written to storage is smaller.
3. Download time and amount of data transferred are reduced when downloading data files.
4. I/O time is reduced by reading less data when reading a subset of the data in a dataset.
5. Only the chunks containing a subset of data are decompressed when pulling a subset of the data.
6. More data can be added to a dataset along any dimension without rewriting existing data.

Costs of Compression

Storing compressed data in chunked data storage layouts has costs which are generally small compared to the benefits provided. However it is not entirely without cost.

1. Computation is required to compress and uncompress data.
2. Chunking is required for compression in HDF5, and with it comes overhead - minimal metadata stored in file to track chunk location on disk.

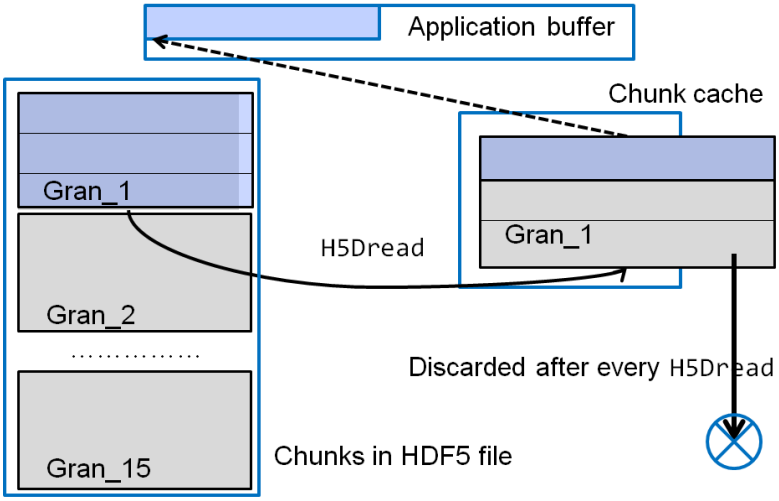
Whether the benefit is greater than the cost should be considered when deciding the storage layout of a particular dataset in HDF5.

Pitfalls: What Can Go Wrong?

Layout of datasets may lead to problems when layout options were not understood, when read access patterns were not considered, or when users access the data in a different way than was anticipated when the dataset was written. These are some common if infrequent issues:

1. Very large chunks are efficient for writing, but if an application later uses a subset of the data the entire chunk must be decompressed.
2. Very small chunks have excessive overhead. The extreme case is choosing a chunk size of 1 element, which requires overhead storage with little or no benefit from compression.
3. Reading the data from a dataset with selections oriented in a different dimension than the orientation chosen for chunks when the dataset was created. The extreme case for this problem can require decompressing the same data multiple times resulting in very slow reads. See example below.
4. HDF5 datasets have a 1 MB chunk cache for each dataset by default. This can mitigate problem #3, but can also exhaust the system memory when memory is limited and large numbers of datasets are open concurrently.

Example: slow reads



To demonstrate the worst case scenario for a compressed dataset, we repacked an NPP data file SCRIS_npp_d20140522_t0754579_e0802557_b13293_c20140522142425734814_noaa_pop.h5 , applying shuffle and gzip filters. The time to read a 60x30x9x717 ES_ImaginaryLW dataset by each 1x1x1x717 element "row" increased from .1 seconds in the original file to 345 seconds in the compressed file. The time increased by more than 300 times because each of the 16200 reads decompressed the entire dataset. Read the User Options to Overcome Problems for possible solutions to the problem.

User Options to Overcome Problems

When reading data files previously written with compressed datasets having a chunk layout that doesn't match the read pattern of the application in use, the pitfalls described in the previous column can occasionally lead to very slow reading operations. Which options are available will depend on the user's ability to configure or change the application in use, or to create a modified copy of the data to be read. There are several options available:

1. Adjust chunk cache size to hold all chunks of data being read from a dataset
2. Increase the size of data reads to decrease the overall number of reads.
3. Change the read access pattern in alignment with the chunks in which the data is stored.
4. Use the h5repack tool, altering the dataset chunks to match the application read pattern.

Example: troubleshooting slow reads

Useful tools:

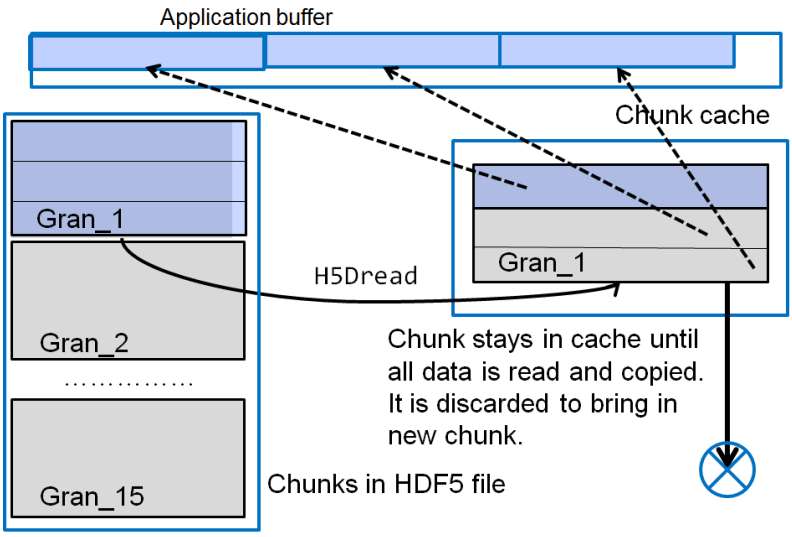
1. h5dump
2. h5ls

Is the application's data read selection orientation incompatible with the dataset chunk layout?

"h5dump -pH -d <dataset path> <filepath or filename>" will show dataset size and chunk sizes

Adjust the chunk cache size

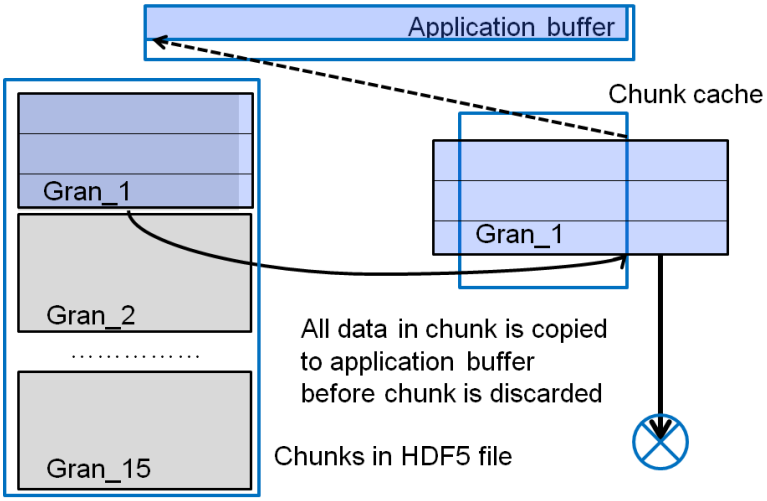
An application can adjust default cache size by calling either H5Pset_cache or H5Pset_chunk_cache functions. H5Pset_cache sets cache size for all chunked datasets in a file, while H5Pset_chunk_cache sets cache size for a particular dataset.



Increase size of data read

The HDF5 library performs I/O on the whole chunk. If chunk doesn't fit into chunk cache, library copies requested data from the chunk into a conversion buffer. Then the library performs datatype conversion and copies the data to an application buffer. The chunk is evicted and will be brought back if more data from the same chunk is requested, i.e., chunk cache is completely ignored in this case.

It is clear that if application keeps reading data from the same chunk, and adjusting cache is not an option, application can change access pattern to read as much data from the chunk as possible with one read operation. For example, instead of reading by one row, read several rows at once.



Change Chunk Size

Chunk size is set at dataset creation time and cannot be changed later. One can use the h5repack tool to change storage layout properties. This command will repack and change the chunk size of the /All_Data/CrIS-SDR_All/ES_ImaginaryLW dataset from 4x9x30x717 to 1x1x30x717, making chunk size ~82K instead of the original 3MB size:

% h5repack -l /All_Data/CrIS-SDR_All/ES_ImaginaryLW:CHUNK=1x1x30x717 gz6_SCRIS_npp_d20140522_t0754579_e0802557_b13293__noaa_pop.h5 new.h5

Recommendations

When compression is enabled for an HDF5 dataset, the library must always read the entire chunk for each call to H5Dread unless the chunk is already in cache. To avoid trashing the cache, make sure that chunk cache size is big enough to hold the chunks containing the data read or an application reads the whole chunk at once.

When compression is disabled, the library's behavior depends on the cache size relative to the chunk size. If the chunk fits in the cache, the library reads the entire chunk for each call to H5Dread unless it is in the cache. If the chunk doesn't fit the cache, the library reads only the data that is selected. There will be more read operations, especially if the read plane does not include the fastest changing dimension.

Producer Options to Avoid Problems

Data producers will naturally focus on writing data efficiently. Application but the data stored may be read many times or by many users. Compressing data to reduce the size of files to be downloaded will benefit users, but producers should also consider issues that may arise when data is accessed by user applications.

Small datasets should be stored in one chunk. Chunk size should match the dataset size up to ~64K. Chunk size for larger datasets that will not be increasing in size should be chosen to approximately fit the total size of the dataset.

1. Chunk shape for creating datasets that may be accessed according to various dimensions should not be biased in any particular dimension. For example, a large 2D dataset should not be divided by rows so that each chunk includes just a few rows but spans all columns.

Open Source Benefits

HDF5 is an open source file format and library with tools to examine and modify HDF5 data files. NASA EOSDIS makes available a wealth of earth science data to a large number of users who use it for a wide variety of purposes. A number of those users provide important feedback to The HDF Group, reporting problems, making suggestions for improvement to existing code and future development, and assisting other users via the HDF Forum.

1. Open source software provides users the opportunity to modify code or settings for unique situations that would not be available otherwise.
2. An appropriate compression method can be chosen from a number of available methods.
3. Ideally users of open source software will provide feedback to providers and advice to other users in user forums.
4. Providers of open source software should provide clear and accurate documentation to promote the usefulness of their software.

References

1. The GZIP home page, http://www.gzip.org/
2. SZIP compression, https://www.hdfgroup.org/doc_resource/SZIP/
3. “Chunking in HDF5”, The HDF Group, http://www.hdfgroup.org/HDF5/doc/Advanced/Chunking/index.html
4. “HDF5 User’s Guide”,, Section 5.4. , http://www.hdfgroup.org/HDF5/doc/UG/
5. “Using Compression in HDF5”, The HDF Group, http://www.hdfgroup.org/HDF5/faq/compression.html
6. The HDF Group. “HDF5 Tutorial”, http://www.hdfgroup.org/HDF5/Tutor/introductory.html
7. The HDF Group. “HDF5 User’s Guide”, Section 5.4.2 http://www.hdfgroup.org/HDF5/doc/UG/index.html
8. The HDF Group. “Filters in HDF5”, http://www.hdfgroup.org/HDF5/doc/H5.user/Filters.html
9. The HDF Group. “HDF5 Dynamically Loaded Filters”, http://www.hdfgroup.org/HDF5/doc/Advanced/DynamicallyLoadedFilters/HDF5DynamicallyLoadedFilters.pdf
10. The HDF Group. “HDF5 Reference Manual”, Property Lists http://www.hdfgroup.org/HDF5/doc/RM/RM_H5P.html#Property-FilterBehavior.
11. The HDF Group, “HDF5 File Format”, Description of the Filter Mask, http://www.hdfgroup.org/HDF5/doc/H5.format.html#V1Btrees
12. The HDF Group. “SZIP compression in HDF products”, http://www.hdfgroup.org/doc_resource/SZIP/
13. “Joint Polar Satellite System (JPSS) Common Data Format Control Book (External) Volume I – Overview”, http://npp.gsfc.nasa.gov/sciencedocuments/2014-07/474-00001-01_JPSS-CDFCB-X-Vol-I_0200-.pdf
14. The HDF Group. “Chunking in HDF5”, http://www.hdfgroup.org/HDF5/doc/Advanced/Chunking/index.html, October 5, 2010.
15. Elena Pourmal “Why doesn't HDF5 compression work?, published location TBD (May 24, 2014).
16. The HDF Group. “HDF5 Advanced Topics: HDF5 chunking and Performance Issues”, http://www.hdfeos.net/workshops/ws13/presentations/day1/HDF5-EOSXIII-Advanced-Chunking.ppt, HDF/HDF-EOS Workshop XIII, November 3-5, 2009.