Expanding HDF5 Capabilities

Elena Pourmal elena.pourmal@lifeboat.llc
John Mainzer john.mainzer@lifeboat.llc

Lifeboat

Outline

- Introduction to Lifeboat, LLC
- Multi-threaded access to data in HDF5
- Support for sparse and variable-length data in HDF5

Lifeboat, LLC

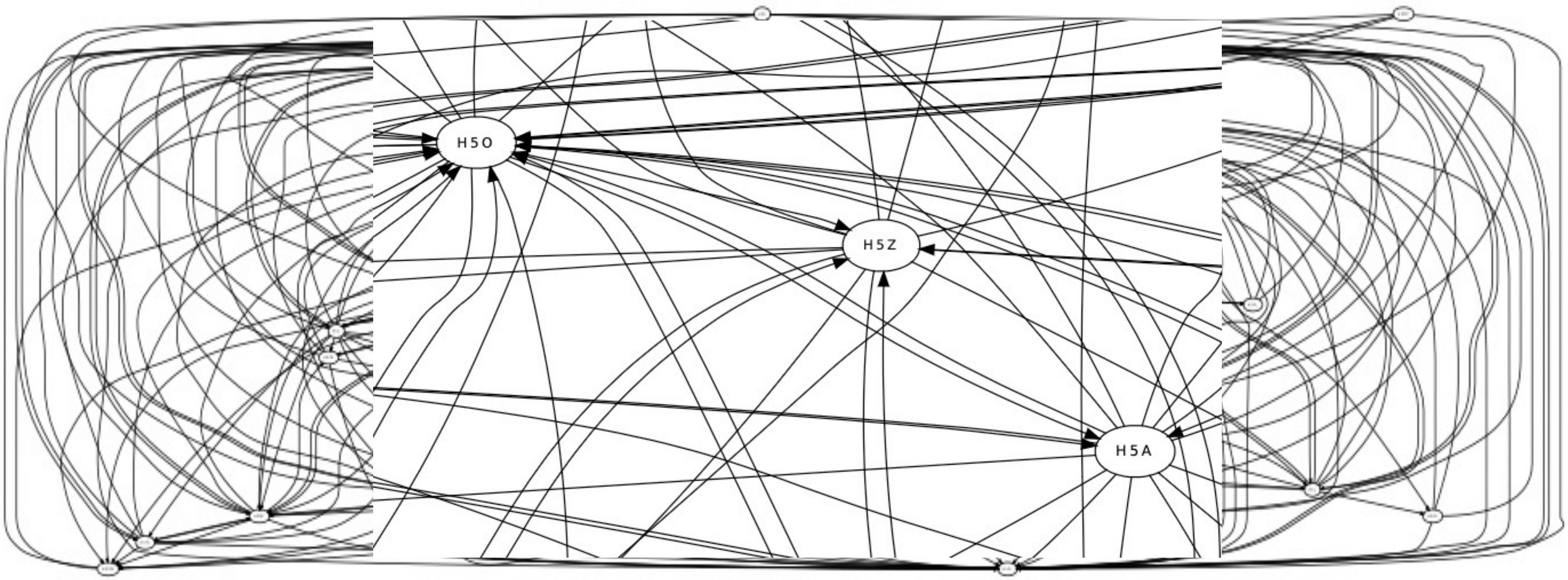
U.S. DEPARTMENT OF ENERGY AWARDEE™

We don't make HDF5… we make HDF5 *better*

- Goal: Sustain and enhance open source HDF5
  - Founded in August 2021
  - Located in Champaign, IL and Laramie, WY
  - www.lifeboat.llc
  - info@lifeboat.llc
- Funded by DOE SBIR/STTR Program
  - Phase I and Phase II: "*Toward multi-threaded concurrency in HDF5*"
  - Phase I: "*Supporting sparse data in HDF5*"

Multi-threaded access to data in HDF5

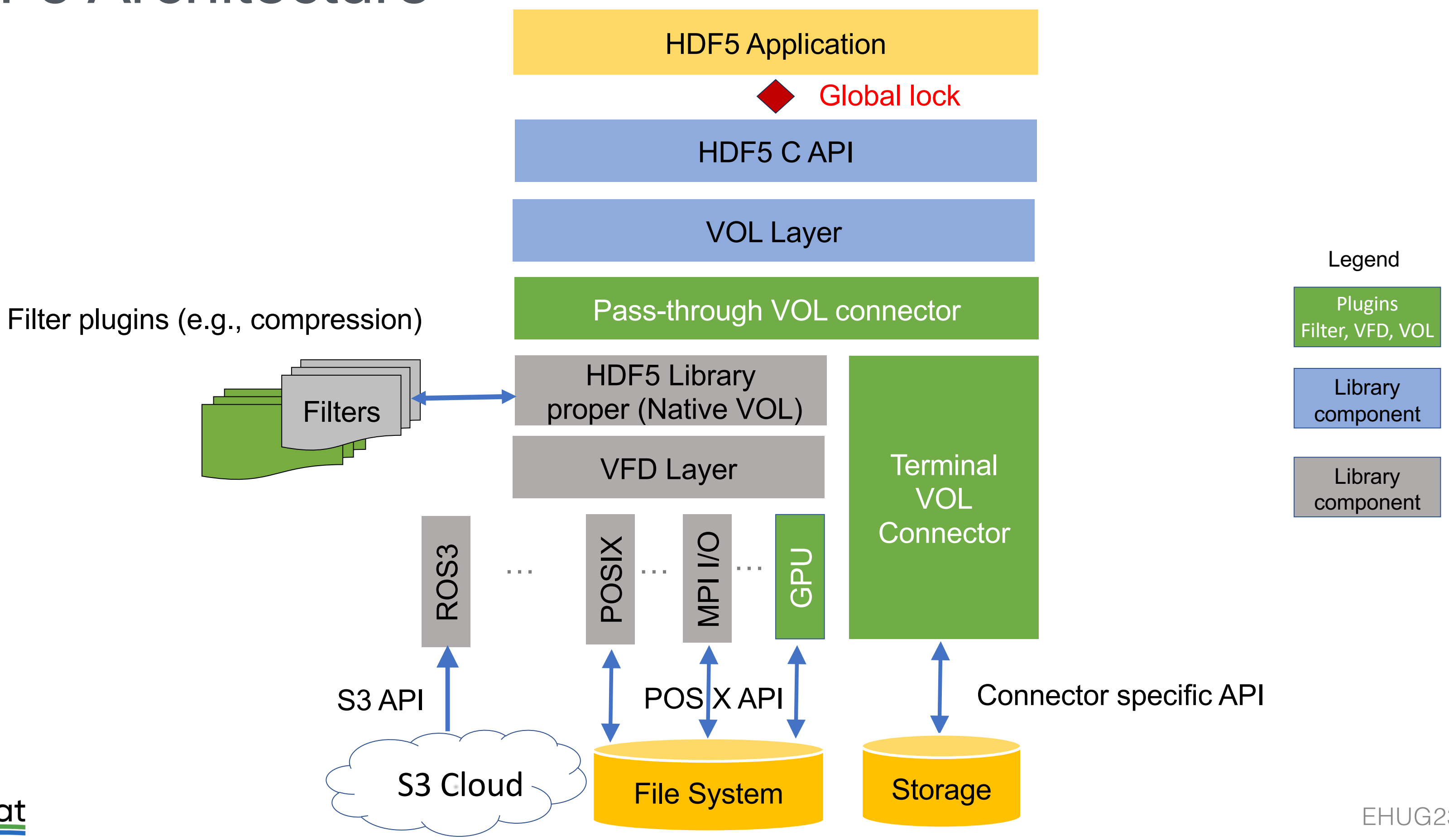
Feasibility of Multi-Threaded HDF5 library



Calling graph of a CGNS library test shows interdependencies between HDF5 packages.

Is multi-threading even possible?

HDF5 Architecture

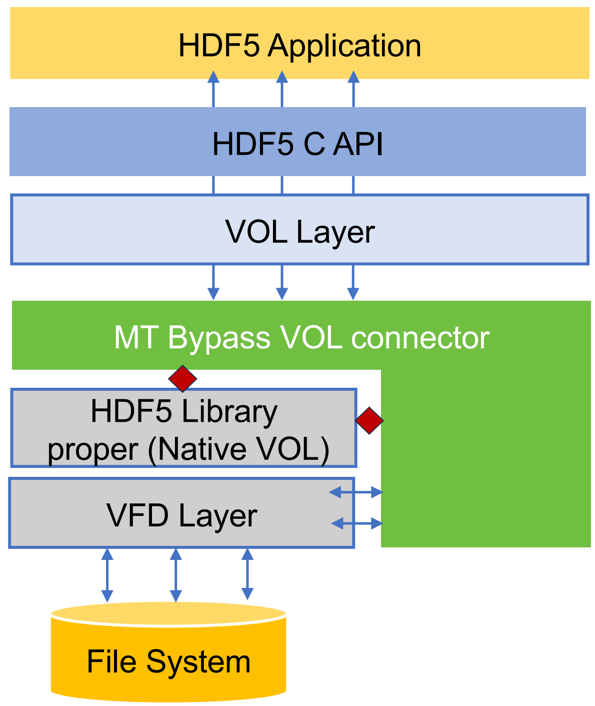


Bypass VOL Connector

Multi-threaded HDF5 modules required by VOL connectors

H5I, H5E, H5P, H5CX, H5VL, H5S, H5FD

Work in progress | Low priority for the first release



Bypass VOL Concept

Initially will support I/O for contiguous and chunked datasets; no data filtering

- Checks if I/O is supported; routs to native VOL or to connector proper (hits mutex)
- Queries HDF5 library for the location of raw data (hits mutex)
- Executes raw data I/O in parallel in multiple threads
- Functionality will be extended as parts of the HDF5 library (e.g., metadata cache) are converted
- See documentation https://github.com/LifeboatLLC/MT-HDF5
- Check HUG23 “Toward Multi-Threaded Concurrency in HDF5” talk by John Mainzer

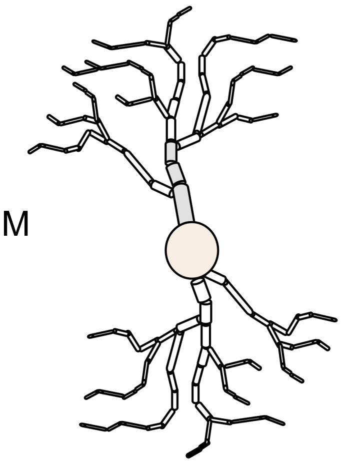
Bypass VOL Connector (cont'd)

- Prototyped VOL connector functionality was implemented by Luc Grosheintz, EPFL, Blue Brain Project in collaboration with John Mainzer and Elena Pourmal

- Next slides show achieved performance improvements

Digitally Reconstructed Neurons – Blue Brain Project

1k - 100M
neurons



```
{
  "0000": {
    "points”: np.empty((9610, 3), np.float32),
    “offsets": np.empty(21, np.uint64)
  },
  "0001": {
    "points": np.empty((14983, 3), np.float32),
    "offsets": np.empty(48, np.uint64)
  },
  ...
}
```

Hardware:

- Intel Xeon Gold 6140
  - 2x 18 cores
  - 6 memory channels
- 100 Gb/s InfiniBand
- SpectreScale/GPFS:
  - 2x GS14KX
  - 8x EDR
  - HDD

Synthetic Data Presented:

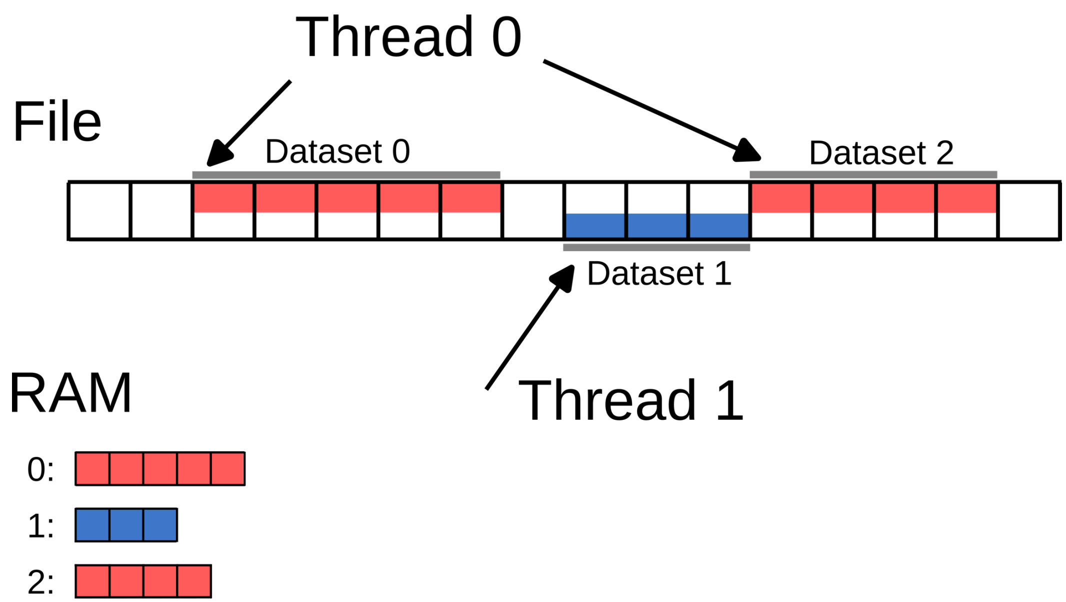
Datasets: 20'000
Total size: 17 GB
File Space Strategy: Paged allocation
Page size: 64 kB

Lifeboat

Slide courtesy of Luc Grosheintz, Blue Brain Project, EPFL

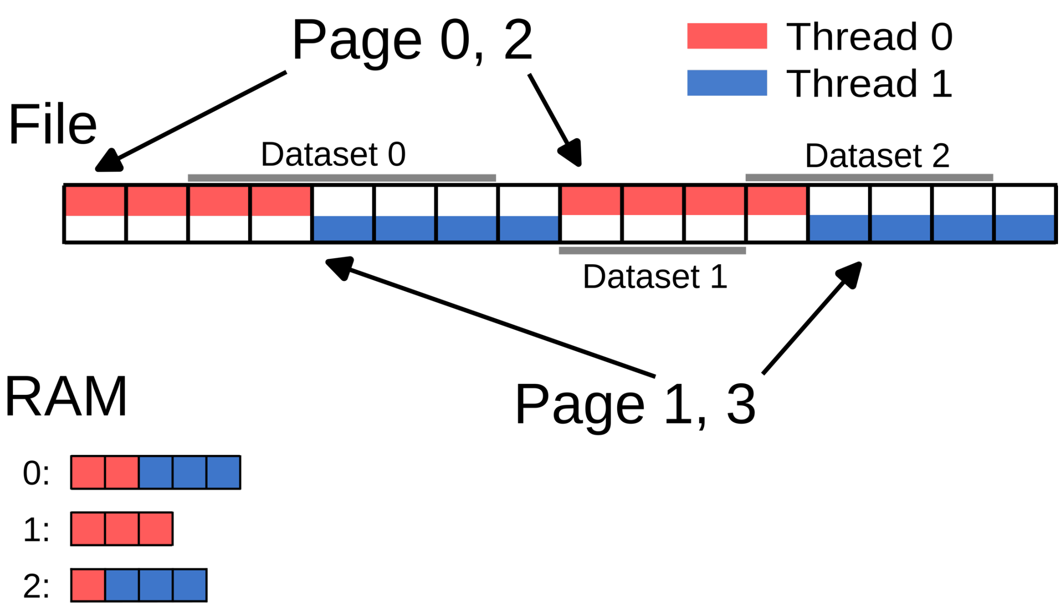
EPFL Blue Brain Project

Direct OpenMP HDF5 Prototype



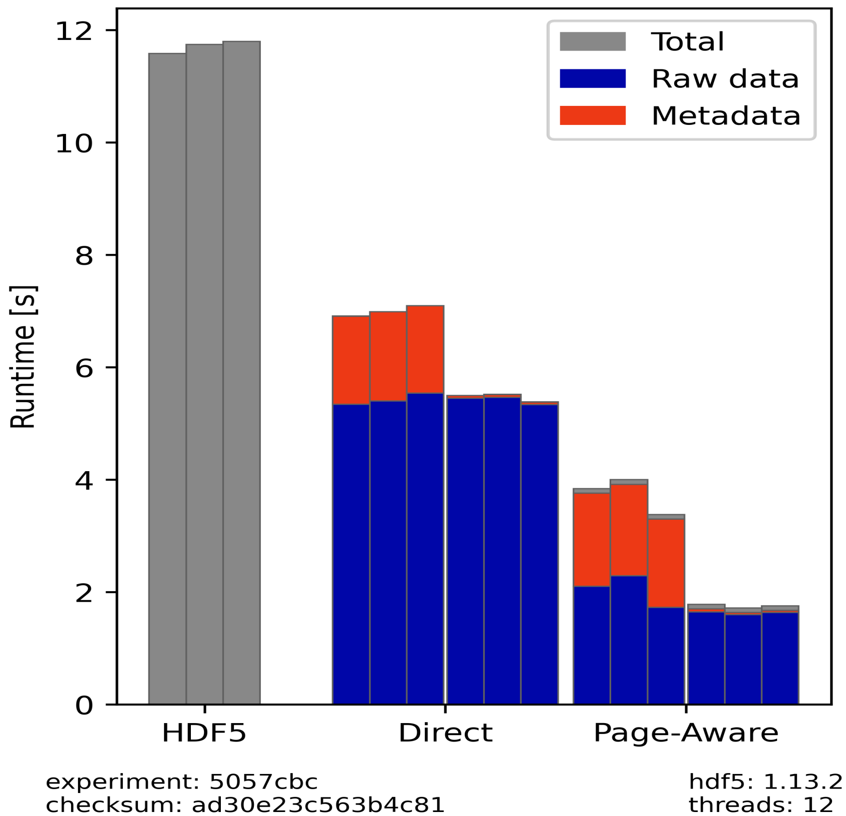
1. Sequentially, read shape/size and offset from beginning of the file for each dataset.
2. Concurrently, `std::fseek` & `std::fread` individual datasets.

Page-aware OpenMP HDF5 Prototype



1. Sequentially, read shape/size and offset from beginning of the file for each dataset.
2. Pre-allocate datasets.
3. Sort by page.
4. Concurrently loop over pages.

Results HDF5 VOL Connector Prototype



Experimental Setup:

- 12 Threads
- 3 measurements

HDF5: Plain HDF5 with 512 MB page buffer, 75% reserved for raw data.

Direct / Page-Aware: The two variants of the prototype.

- **Left:** Read metadata using HDF5
- **Right:** Read metadata from JSON

Best result achieves the effective single node bandwidth of GPFS over InfiniBand.

Support for sparse and variable-length data in HDF5

What is Sparse Data?

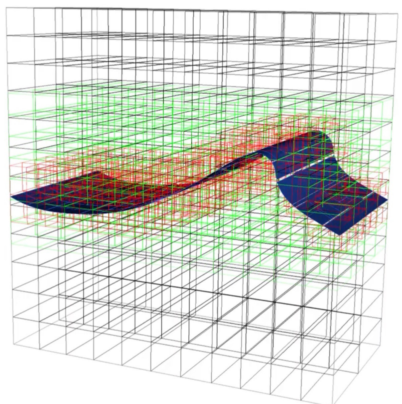
- Sparse data is ubiquitous; examples come from the experimental sciences and computer modeling:
  - High Energy Physics (HEP); Neutron and X-Ray scattering; Mass Spectrometry experiments; Transmission electron microscopy; Genomics; AMR
- There is no ”standard” definition of “sparse data”.
  - **Linear algebra** – data is considered sparse if less than 30% of matrix elements are non-zeros.
  - **Experimental sciences** - only 0.1% to 10% of gathered data is of interest, but it may contain a bigger percentage.

Use Cases

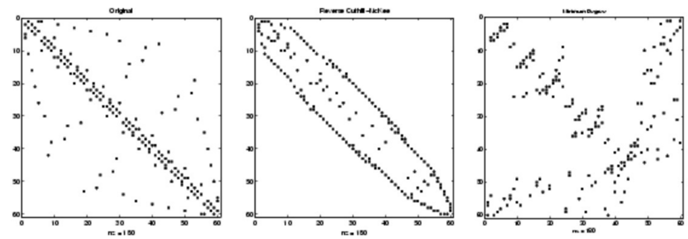
See notes for references

Computer modeling

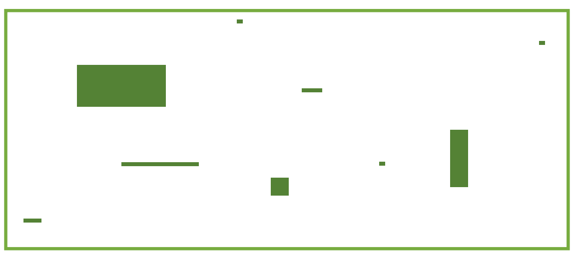
AMR



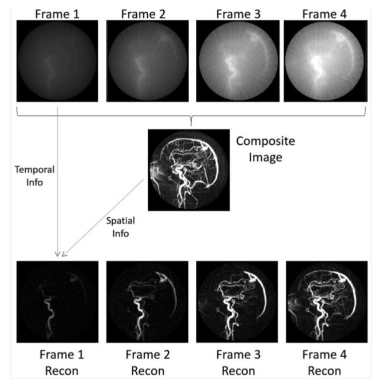
Linear algebra



SLAC use case: LCLS-II images



Sparse Reconstruction in MRI



Motivation for Sparse Storage: LCLS-II Use Case

- Experiments produce a stream of two-dimensional images.
- For each image it is possible to automatically identify either:
  - A rectangular **R**egion **o**f **I**nterest (ROI) in each image which will typically comprise about 10% of the image, or
  - 50 – 100 small subsections in each image (typically 5 to 10 contiguous points or pixels).
  - The number, size, configurations, and locations of ROI or the small subsections change over time.
- For each image in the stream it is desired to store
  - Only the ROI or the point list in a three-dimensional HDF5 dataset
    - One must be able to recover both the location and contents of the ROI and/or the elements of the point list.
  - Every N^{th} two-dimensional image in full, where N is constant over any given experiment. Note that the ROI or point list of each “full” two-dimensional image must be recoverable as well.

LCLS-II Use Case (cont'd)

- To meet this requirement, we propose to implement sparse datasets:
  - Only the entries that have been written explicitly are defined.
  - The defined entries can be readily identified, and read. To the above minimal requirement, we also add:
  - ***Compatibility with dense datasets*** – thus code designed for the existing dense datasets will still work, reading defined values if available, and the fill value (default 0) where not.
  - ***Ability to use filtering*** (compression).
  - ***Ability to erase defined values*** – that is to remove them from the set of defined values.
  - ***Data is portable***, i.e., doesn't depend on data storage in memory
- See Reference slide for pointers to RFCs

New Storage Paradigm: Idea of Structured Chunk

Chunked dataset

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	66	69	72	75	78	81	0	0
0	0	96	99	102	105	108	111	0	0
0	0	126	129	132	135	138	141	0	0
0	0	0	0	0	0	0	0	0	2
100	0	-100	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	3	0

0 may represent a value that is not-defined

If we write a shown sub-array using hyperslab selection how the chunk will be stored in the file?

Chunked storage: all chunk elements are stored

`0 0 0 0 0 0 0 0 0 0 0 0 66 69 72 0 0 96 99 96 102`

Structured Chunk storage:
Locations and values of defined elements specified by the hyperslab selection are stored in different sections of the chunk

Section 0	Encoded selection
Section 1	66 69 72 96 99 96 102

Structured Chunk for Fixed and Variable-size Data

Fixed-size datatype

<table>
<tr><th>byte</th><th>byte</th><th>byte</th><th>byte</th></tr>
<tr><td colspan="4">Section 0: Encoded Selection of Defined Elements</td></tr>
<tr><td colspan="4">Section 0 Checksum</td></tr>
<tr><td colspan="4">Section 1: Fixed Length Data Section</td></tr>
</table>

VL-size datatype

<table>
<tr><th>byte</th><th>byte</th><th>byte</th><th>byte</th></tr>
<tr><td colspan="4">Section 0: Encoded Selection of Defined Elements</td></tr>
<tr><td colspan="4">Section 0 Checksum</td></tr>
<tr><td colspan="4">Section 1: Fixed Length Data Section</td></tr>
<tr><td colspan="4">Section 1 Checksum</td></tr>
<tr><td colspan="4">Section 2: Variable-size data heap</td></tr>
<tr><td colspan="4">Section 2 Checksum</td></tr>
</table>

Lifeboat

Programming Model

```
/*
 * Create the dataset creation property list, add the gzip filter to compress all
 * sections of the sparse chunk using DEFLATE filter.
 */
dcpl   = H5Pcreate (H5P_DATASET_CREATE);
status = H5Pset_layout (dcpl, H5D_SPARSE_CHUNK);
status = H5Pset_chunk (dcpl, 2, chunk_dims);
status = H5Pset_deflate (dcpl, 9);

/* Create the dataset */
dset = H5Dcreate (file, DATASET, H5T_STD_I32LE, space, H5P_DEFAULT, dcpl, H5P_DEFAULT);

/* Create hyperslab selection in memory and in the file */
 …..

/* Write the data to the dataset */
status = H5Dwrite (dset, H5T_NATIVE_INT, mspace_id, fspace_id, H5P_DEFAULT, buf[0]);
```

Proposed New APIs

Function Name	Short Description
H5Dget_defined	Retrieves a dataspace object with the defined elements
H5Derase	Deletes elements from a dataset
H5Dwrite_struct_chunk	Writes structured chunk
H5Dread_struct_chunk	Reads structured chunk
H5Dget_struct_chunk_info	Gets structured chunk info
H5Dget_struct_chunk_info_by_coord	Retrieves the structured chunk information
H5Dstruct_chunk_iter	Iterates over all structured chunks in the dataset
H5Pset_filter2	Adds a filter to a filter pipeline for a specified section of sparse structured chunk
H5Pget_nfilter2	Returns the number of filters in the pipeline for a section of structured chunk
More filter functions	….

H5Pset_filter2

- We want to address deficiency of the current API for passing filter's data

```
herr_t H5Pset_filter2 (hid_t plist_id,
                       uint64_t section_number,
                       H5Z_filter_t filter,
                       uint64_t flags,
                       size_t buf_size,
                       const void *buf)
```

new parameter

new datatype

Programming model (cont’d)

```
/* Apply compression methods to different sections of
 * a structured chunk. In this example, sparse chunk has two sections.
 * We are using gzip compression on the encoded selection section
 * and szip on the fixed-size data section.
 */
flags = H5Z_FLAG_OPTIONAL;
status = H5Pset_filter2 (dcpl, H5Z_FLAG_SPARSE_SELECTION,
                               H5Z_FILTER_DEFALTE, flags, nelem, &data);

status = H5Pset_filter2 (dcpl, H5Z_FLAG_SPARSE_FIXED_DATA,
                               H5Z_FILTER_SZIP, flags, …);
```

Acknowledgement

This work is supported by the U.S. Department of Energy, Office of Science under Award number DE-SC0023583 and "Toward multi-threaded concurrency in HDF5".

This work is supported by the U.S. Department of Energy, Office of Science under Award number DE-SC0023583 for SBIR project "Supporting Sparse Data in HDF5".

We thank The HDF Group developers and Quincey Koziol, Principal Engineer, AWS HPC for reviewing our technical proposal and for providing a constructive feedback.

References

1. https://github.com/LifeboatLLC/MT-HDF5/tree/main/design_docs
2. https://github.com/LifeboatLLC/MT-HDF5/tree/main/LFHT
3. https://github.com/HDFGroup/hdf5/discussions/3257
4. John Mainzer, Elena Pourmal, “RFC: File Format Changes for Enabling Sparse Storage in HDF5”. Available from https://github.com/LifeboatLLC/SparseHDF5/
5. John Mainzer, Elena Pourmal, “RFC: Programming Model to Support Sparse Data in HDF5” Available from https://github.com/LifeboatLLC/SparseHDF5/
6. J. Mainzer *et al*., "Sparse Data Management in HDF5," *2019 IEEE/ACM 1st Annual Workshop on Large-scale Experiment-in-the-Loop Computing (XLOOP)*, Denver, CO, USA, 2019, pp. 20-25, doi: 10.1109/XLOOP49562.2019.00009
7. The HDF Group, “Draft RFC: Sparse Chunks”_ https://docs.hdfgroup.org/hdf5/rfc/RFC_Sparse_Chunks180830.pdf
8. The HDF Group, Variable-Length Data in HDF5 Sketch Design, https://docs.hdfgroup.org/hdf5/rfc/var_len_data_sketch_design_190715.pdf

Thank you!

Questions?