HDF5 Features for Managing Experimental and Observational Data and Superfacility

Quincey Koziol, Suren Byna, and John Mainzer

Lawrence Berkeley Laboratory, The HDF Group, and Texas Tech University

Team members: Houjun Tang, Tony Li, Gerd Heber, Scot Breitenfeld, Jake Smith, Wei Zhang, Chenxu Niu, and Yong Chen

Experimental and Observational Data - Use cases

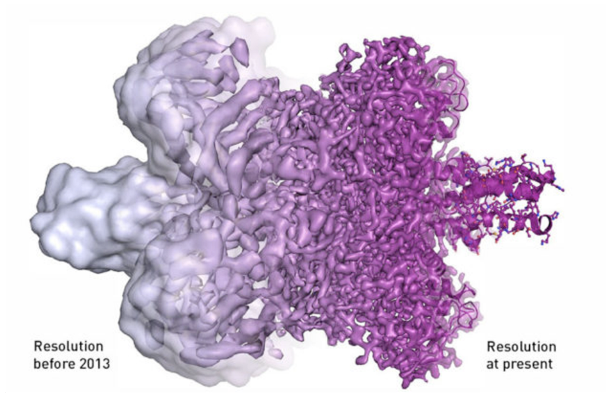


Illustration courtesy of Martin Hogborn/The Royal Swedish Academy of Sciences

NCEM is working on cryoEM (electron microscopy at cryogenic temperatures), will generate high-res images at high frequency

HDF5 reqs: Remote / realtime file transfer, multiple writers - multiple readers (MWMR), and sparse representations

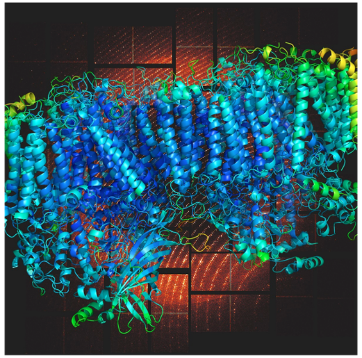
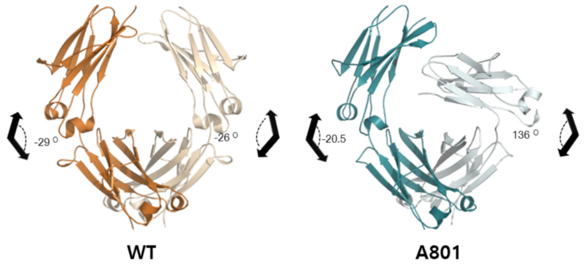


Illustration courtesy of N. Sauter/Berkeley Lab (Data taken with the LCLS X-ray laser)

LCLS-II upgrade will have high repetition rate (1-MHz) and very high data throughput (100GB/s)

HDF5 reqs: Remote / realtime file transfer w/ low latency SWMR, multiple writers - multiple readers (MWMR), sparse representations, efficient variable length data representations, avoid file corruption on crash, data reduction via compression and removing uninteresting data, ...



Solved structures of the Fc tail domain of the IgG antibody, wild type (WT) and engineered (A801) -- Illustration courtesy of Chang-Han Lee, UT Austin

ALS data production rates are anticipated to reach 80 GB/s (1.5 PB/month) over its lifetime. SPOT suite uses HDF5 for managing data.

HDF5 reqs: revising data types / schema while keeping the original data unmodified, manage metadata of several HDF5 files (containers), querying metadata within HDF5 containers, ...

Examples of experimental and observational data use case - LCLS

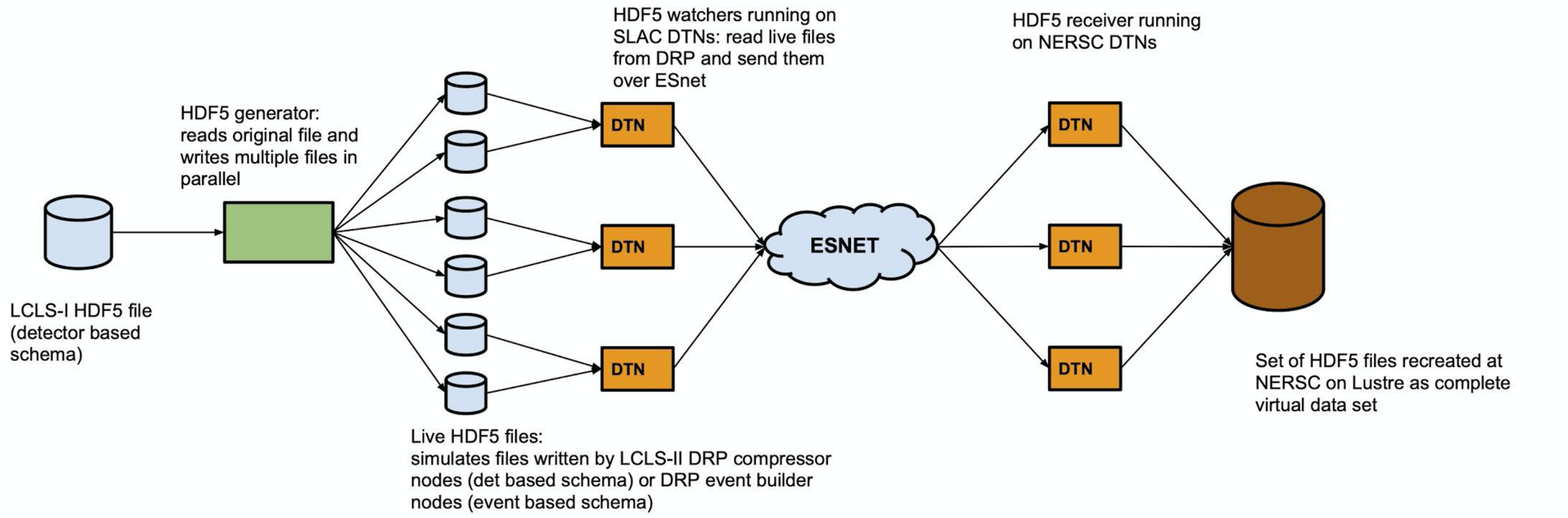

HDF5 watchers running on SLAC DTNs: read live files from DRP and send them over ESnet
HDF5 receiver running on NERSC DTNs
HDF5 generator: reads original file and writes multiple files in parallel
DTN
DTN
DTN
ESNET
DTN
DTN
DTN
LCLS-I HDF5 file (detector based schema)
Set of HDF5 files recreated at NERSC on Lustre as complete virtual data set
Live HDF5 files: simulates files written by LCLS-II DRP compressor nodes (det based schema) or DRP event builder nodes (event based schema)

Image credit: LCLS from SLAC

Examples of experimental and observational data use case - ALS

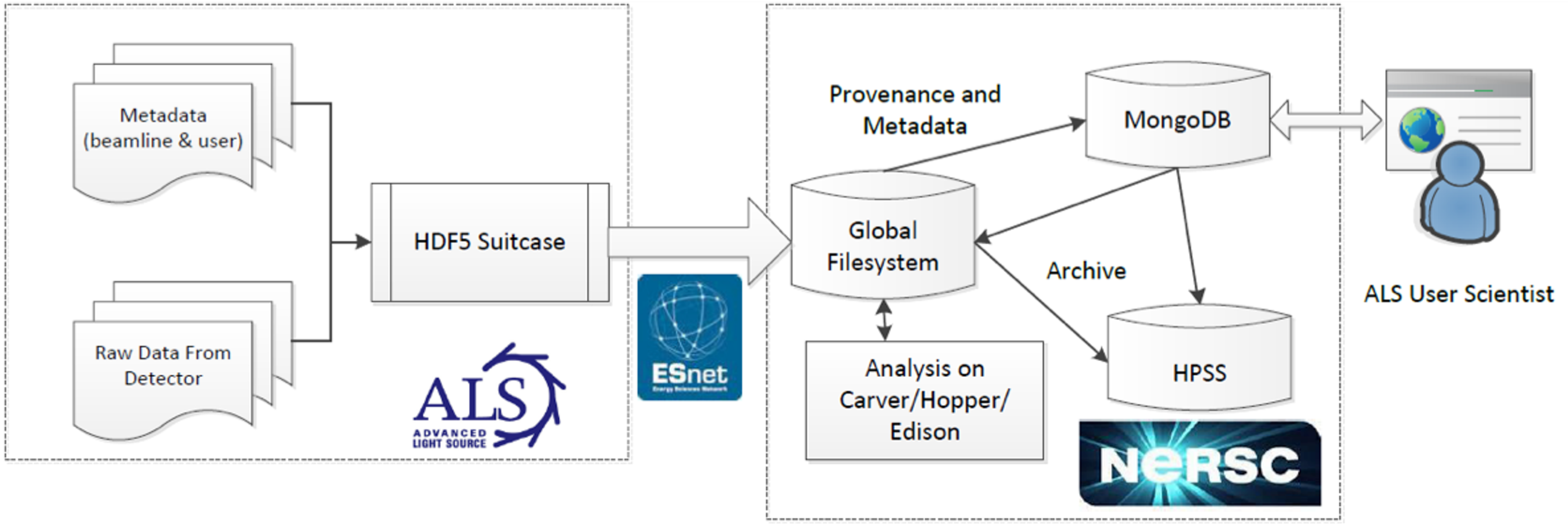


Image credit: Spot Suite slides by Craig Tull, LBNL, http://spot.nersc.gov/

Requirements of EOD use cases

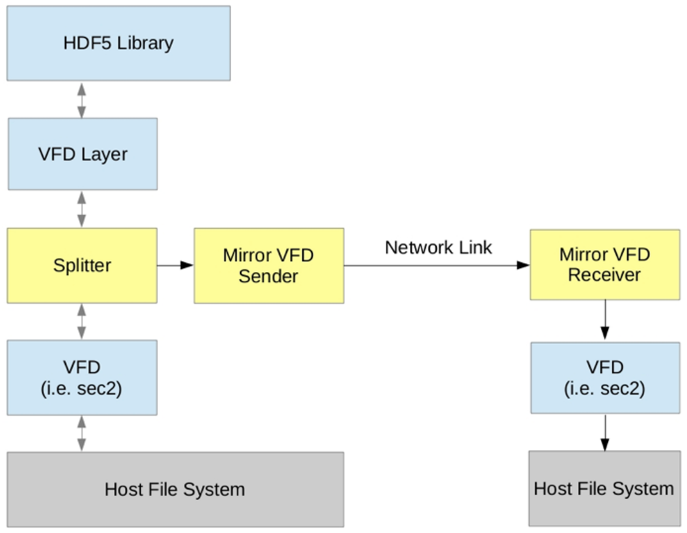
- Requirements of NCEM data
    - Support for sparse data management
    - Support for remote streaming synchronization
    - Multiple producers and multiple consumers of data
- Requirements of LCLS-II data
    - Handle multiple producers and multiple consumers of data
    - Support remote streaming synchronization
    - Support for variable length modes of data
- Requirements of ALS data
    - Extend data models to support changes in data and data schemas
    - Metadata and provenance management for multiple HDF5 containers

EOD-Focused Features in Development

- *Splitter* virtual file driver (VFD) - Split data to different files
- *Mirror* VFD - Replicate HDF5 files in different systems (“streaming rsync”)
    - Probably should be called “client-server” VFD
- Sparse Data management
- *h5prov* - Pass-through virtual object layer (VOL) to capture provenance
- *Onion* VFD - Version control on HDF5 files
- *MIQS* - Metadata and indexing for self-describing formats
- *XTC2 VOL connector* - Terminal VOL connector, to read XTC2 format files
- Variable length data storage improvements
- Streaming API routines - Efficient support for adding records to datasets

Splitter and Mirror VFDs for remote streaming sync

- Remote streaming synchronization
  - Use case: LCLS-2 at SLAC will need to write HDF5 files locally and simultaneously duplicate the files on Cori at NERSC
  - Developed Splitter and Mirror Virtual File Drivers (VFD)
  - Mirror-Writer can send data to a receiver in a different system
  - Testing in progress for transferring data between SLAC and NERSC

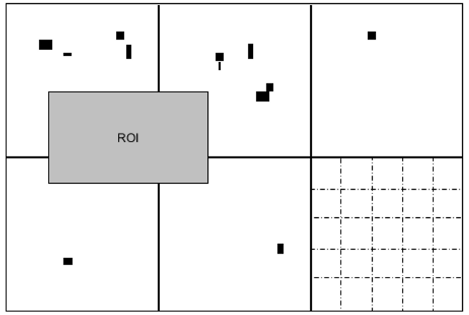


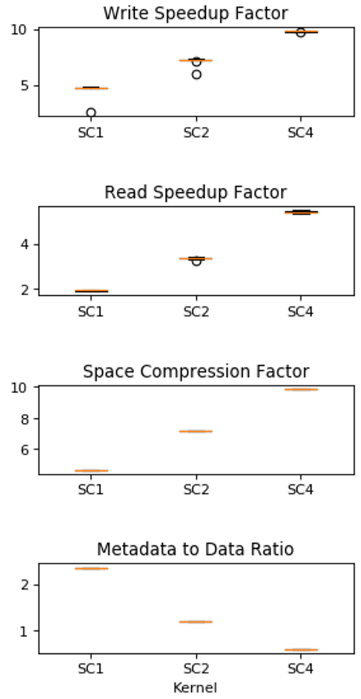
Credit: From John Mainzer's RFC; More on this in John's talk later.

Sparse data management

- A large number of science datasets are sparse
- There is no concept of sparseness in the HDF5 data model
- A few workarounds
  - Chunked layout + Compression, Mimic sparse formats
- Issues with workarounds: hard to determine “written-to” entries, doesn’t preserve array abstraction
- Idea: *Sparse chunks*
  - “Decorated” with a selection that marks the positions of non-fill values
  - Preserves array abstraction and provides space savings
- Initial evaluation of sparse chunks
  - Observed read / write speedups and space compression

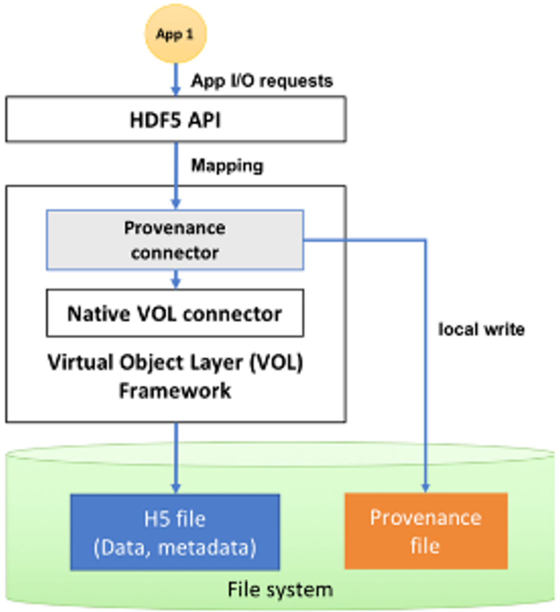
John, Mainzer, Neil Fortner, Gerd Heber, Elena Pourmal, Quincey Koziol, Suren Byna, and Marc Paterno, “Sparse Data Management in HDF5”, 1st IEEE/ACM Annual Workshop on Large-scale Experiment-in-the-Loop Computing, XLOOP@SC19, 2019





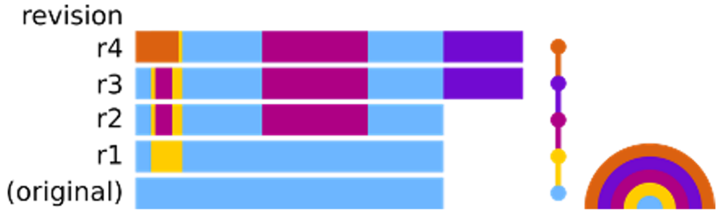
Provenance Capture with h5prov VOL connector

- h5prov Virtual Object Layer (VOL) connector
- Collects:
  - User
  - Application
  - Data – file, datasets within files
  - Timestamps
  - File operations: opens and closes
  - Dataset operations: reads and writes
    - File (physical) offset, array (logical) offset
- Use cases
  - How many unique users and applications are using HDF5
  - Data usage: Which files are used and is there a pattern?
  - Identifying patterns of HDF5 dataset usage to trigger prefetching da
    from parallel file systems



Tonglin Li, Quincey Koziol, Houjun Tang, Jialin Liu, and Suren Byna, "H5Prov: I/O Performance Analysis of Science Applications Using HDF5 File-level Provenance", CUG 2019 - https://sdm.lbl.gov/~sbyna/research/papers/2019/201905-CUG-H5Prov-Provenance.pdf

Onion VFD for version control

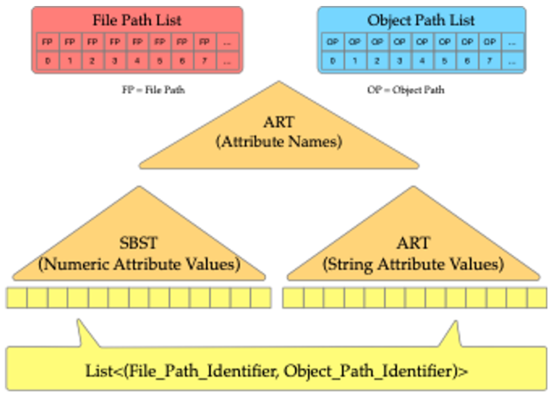


- ALS users require support for tracking modifications to metadata attributes
- There are several requests for tracking modifications to an HDF5 file while preserving or access to the file as it existed in the past
- Solution: Onion Virtual File Driver (VFD)
  - The original file exists with data layered atop one another from an original file to the most recent revision
- Each revision, once committed, cannot itself be modified – rather, a subsequent revision must supply the amended content.
- Revision index to map locations in the logical file to the correct bytes in the backing store
- Browsing revision is possible with a new HDF5 function - H5FDfctl
- Feature is in active development

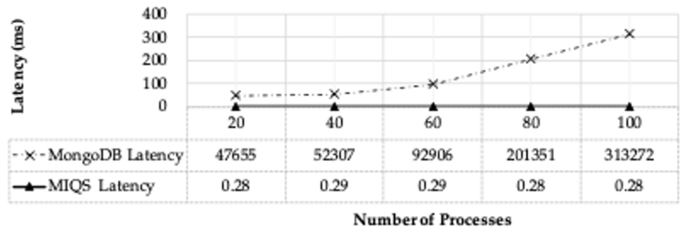
Credit: From John Mainzer's RFC; More on this in John's talk later.

MIQS: Metadata indexing and querying for self-describing files

- Self-describing file formats that store metadata along with data, such as HDF5 and netCDF, are extensively used to store scientific data
- Use case: ALS users require support for
  - Querying attribute data to find data objects
- A typical approach is extracting the metadata and storing it in a database
- To search directly on the metadata in HDF5 files, developed an in memory indexing and querying method that's much faster than databases
  - Because MIQS indexes are in memory compared to MongoDB, MIQS is very fast (172,000X faster)


File Path List
FP = File Path
Object Path List
OP = Object Path
ART (Attribute Names)
SBST (Numeric Attribute Values)
ART (String Attribute Values)
List<(File_Path_Identifier, Object_Path_Identifier)>

A hybrid index for numeric values and string values of attributes



	20	40	60	80	100
MongoDB Latency	47655	52307	92906	201351	313272
MIQS Latency	0.28	0.29	0.29	0.28	0.28

Wei Zhang, Suren Byna, Houjun Tang, Brody Williams, and Yong Chen, "MIQS: Metadata Indexing and Querying Service for Self-Describing File Formats", SC19, https://sdm.lbl.gov/~sbyna/research/papers/2019/201911-EOD_HDF5_Metadata_Search-SC19.pdf

LLANA: SLAC-NERSC Superfacility Project

HDF5 Tasks:

- XTC2 VOL Connector
  - Read-only VOL connector to access XTC2 data through HDF5 API
- Performance Enhancements to Streamed Data I/O
  - “Streaming API” that streamlines appending records to datasets
- Variable-Length Data Storage Improvements
  - “Hybrid chunk” storage that combines VL-data with fixed-data in each chunk
  - “Stream-focused” chunk index,

XTC2 VOL Connector

Goal: Access XTC2 files from HDF5 applications

- Maps HDF5 data model to XTC2 file format
  - A good example of using HDF5 VOL framework to access data in non-HDF5 files
- HDF5 Applications can transparently access data in XTC2 files, without modifying code or even rebuilding
  - Use environment variable to choose XTC2 VOL connector
- Available in the LCLS2 git repository:
  - https://github.com/slac-lcls/lcls2/tree/dev_tony

XTC2 VOL Connector

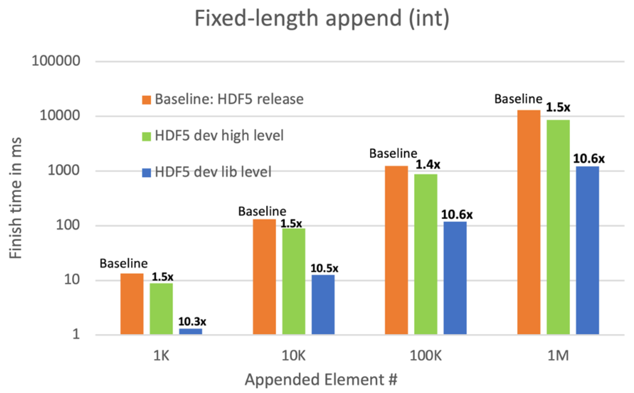
Goal: Access XTC2 files from HDF5 applications

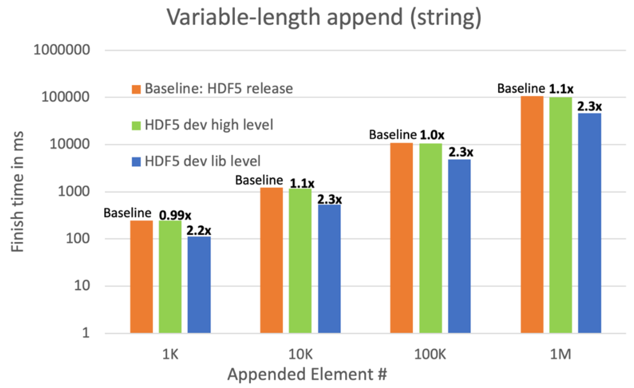
- Maps HDF5 data model to XTC2 file format
  - A good example of using HDF5 VOL framework to access data in non-HDF5 files
- HDF5 Applications can transparently access data in XTC2 files, without modifying code or even rebuilding
  - Use environment variable to choose XTC2 VOL connector
- Available in the LCLS2 git repository:
  - https://github.com/slac-lcls/lcls2/tree/dev_tony

Streaming API

Goal: Improve performance of streamed access to dataset records

- Amortize "setup / teardown" cost of append operations
- New "cursor" API routines:
  - H5Dcursor_create
  - H5Dcursor_write_next / H5Dcursor_read_next
  - H5Dcursor_close



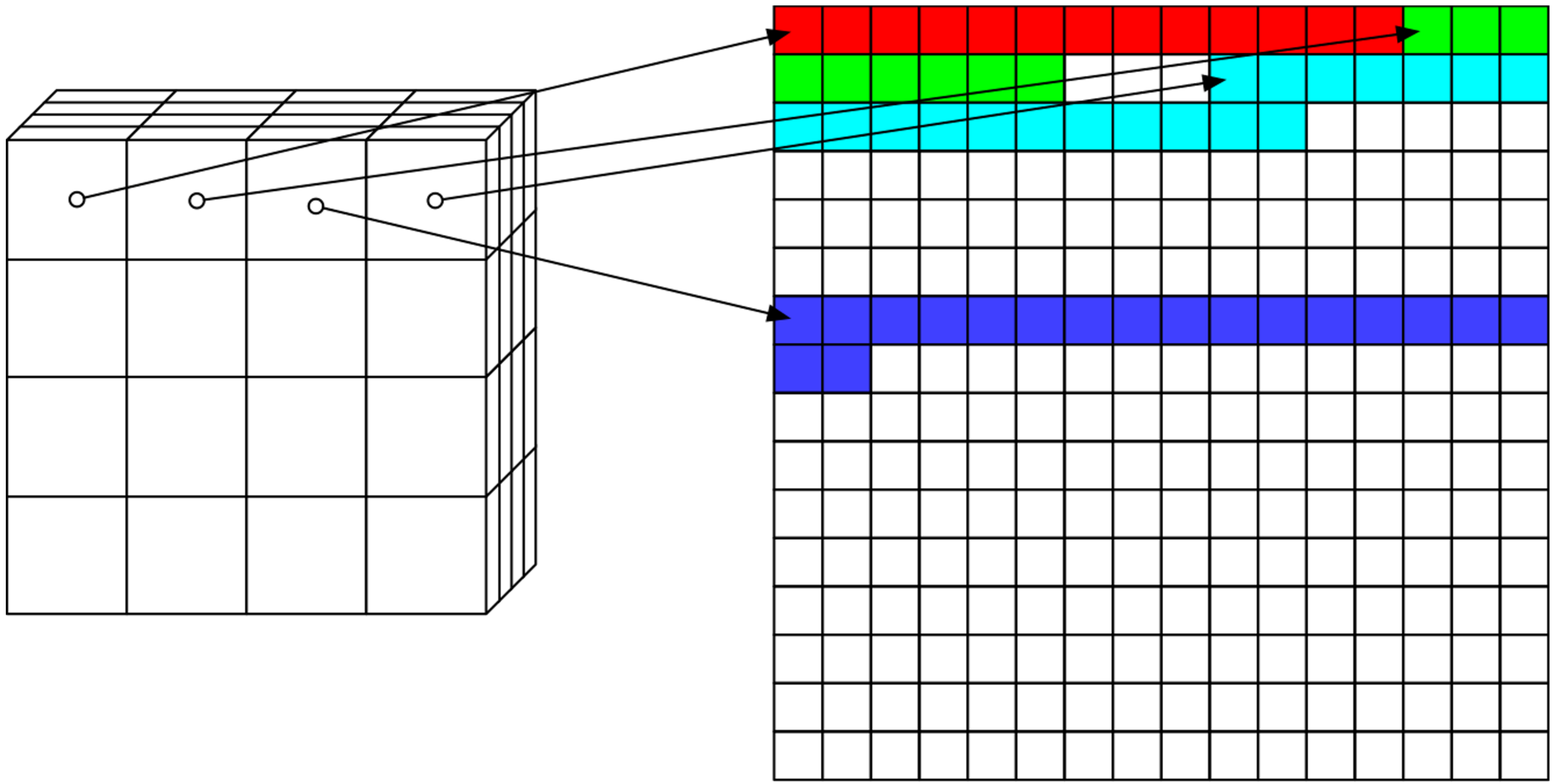


Variable-length data storage

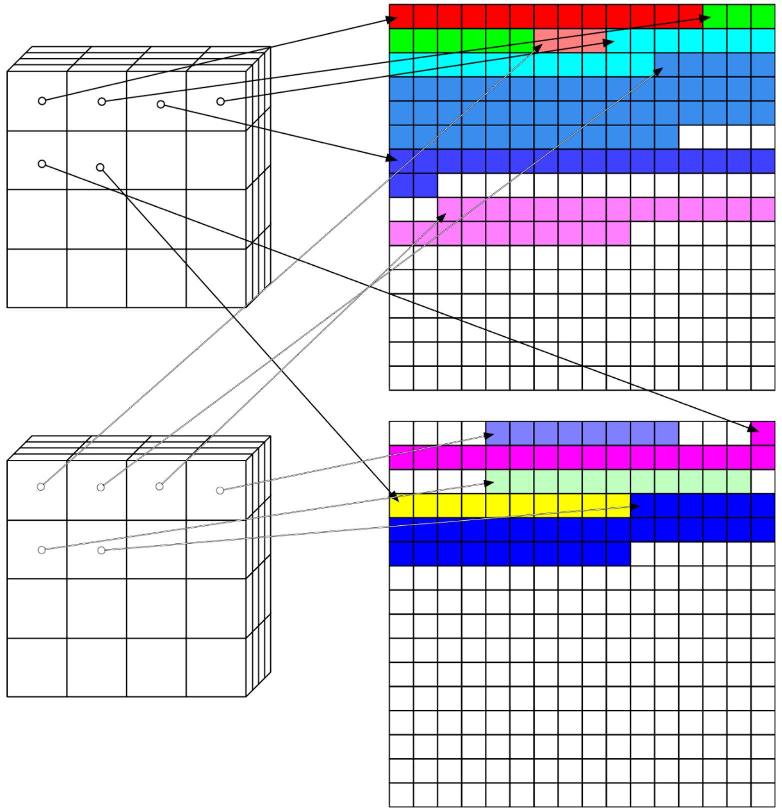
Goal: Improve access time and file size of [streamed] variable-length (VL) data in HDF5 files

- Currently:
    - Separate “heaps” in HDF5 files for storing VL data elements
    - Random-access chunk indices
- Need:
    - “Unified” fixed & VL data element storage
    - Streaming-enabled chunk indices

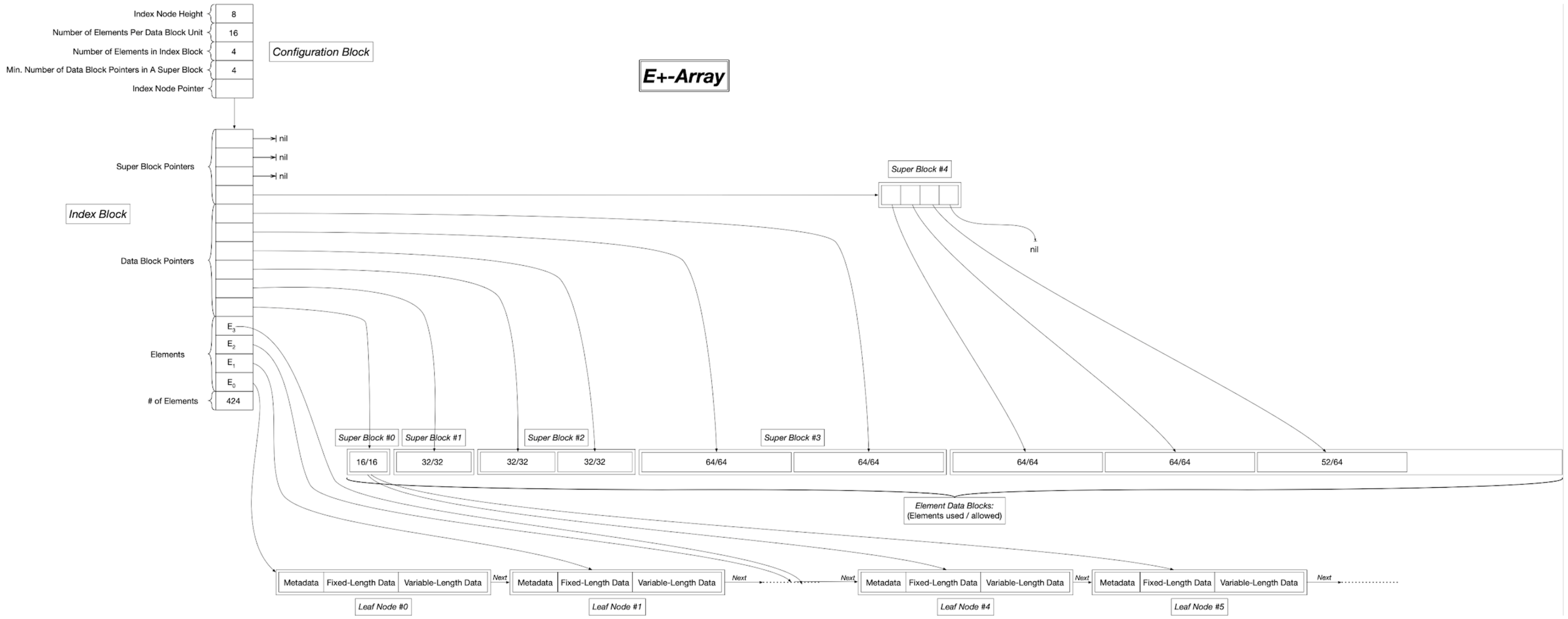
Variable-length data storage



Variable-length data storage



Variable-length data storage



Conclusions

Thanks to:

U.S. DEPARTMENT OF ENERGY | Office of Science

Many efforts toward Experimental & Observational Data needs

- Now:
  - *Splitter* virtual file driver (VFD) - Split data to different files
  - *Mirror* VFD - Replicate HDF5 files in different systems ("streaming rsync")
  - Sparse Data management
  - *h5prov* - Pass-through virtual object layer (VOL) to capture provenance
  - *Onion* VFD - Version control on HDF5 files
  - *MIQS* - Metadata and indexing for self-describing formats
  - *XTC2 VOL connector* - Terminal VOL connector, to read XTC2 format files
  - Variable length data storage improvements
  - Streaming API routines - Efficient support for adding records to datasets
- Future:
  - Pursuing continued funding for more work