Achieving High Performance I/O with HDF5

HDF5 Tutorial @ ECP Annual Meeting 2020

M. Scot Breitenfeld, Elena Pourmal
The HDF Group

Suren Byna, Quincey Koziol
Lawrence Berkeley National Laboratory

Houston, TX
Feb 6th, 2020

Outline, Announcements and Resources

INTRODUCTION

Tutorial Outline

- Foundations of HDF5
- Parallel I/O with HDF5
- ECP HDF5 features and applications
- Logistics
  - Live Google doc for instant questions and comments: https://tinyurl.com/uoxkwaq

https://tinyurl.com/uoxkwaq

Announcements

- HDF5 User Group meeting in June 2020
- The HDF Group Webinars
  https://www.hdfgroup.org/category/webinar/
  - Introduction to HDF5
  - HDF5 Advanced Features
  - HDF5 VOL connectors

https://tinyurl.com/uoxkwaq

Resources

- HDF5 home page: http://hdfgroup.org/HDF5/
- Latest releases:

➢HDF5 1.10.6 https://portal.hdfgroup.org/display/support/Downloads

➢HDF5 1.12.0 https://gamma.hdfgroup.org/ftp/pub/outgoing/hdf5_1.12/

- HDF5 repo: https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5/
- HDF5 Jira https://jira.hdfgroup.org
- Documentation https://portal.hdfgroup.org/display/HDF5/HDF5

https://tinyurl.com/uoxkwaq

FOUNDATIONS OF HDF5

Elena Pourmal

Foundations of HDF5

- Introduction to
  - HDF5 data model, software and architecture
  - HDF5 programming model
- Overview of general best practices

https://tinyurl.com/uoxkwaq

Why HDF5?

- Have you ever asked yourself:
  - How do I organize and share my data?
  - How can I use visualization and other tools with my data?
  - What will happen to my data if I need to move my application to another system?
  - How will I deal with one-file-per-processor in the exascale era?
  - Do I need to be an “MPI I/O and Lustre, or Object Store, etc.” pro to do my research?
- HDF5 is an answer to the questions above and can hides all complexity so you can concentrate on Science

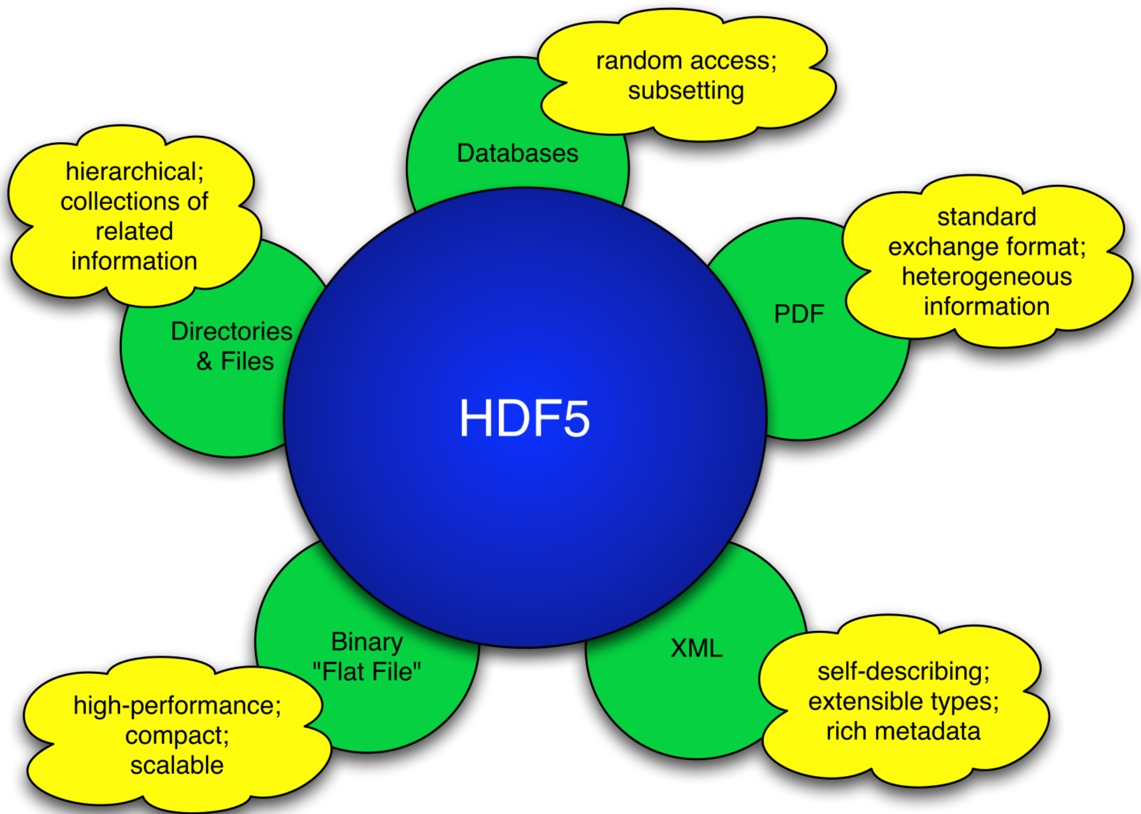
https://tinyurl.com/uoxkwaq

WHAT IS HDF5?

What is HDF5?

- Hierarchical Data Format version 5 (**HDF5**)
  - An extensible **data model**
    - Structures for data organization and specification
  - Open source **software** (I/O library and tools)
    - Performs I/O on data organized according to the data model
    - Works with POSIX and other types of backing store: Object Stores (DAOS, AWS S3, AZURE, Ceph, etc.), memory hierarchies and other storage devices
  - Open **file format** (POSIX storage only)

HDF5 is like …



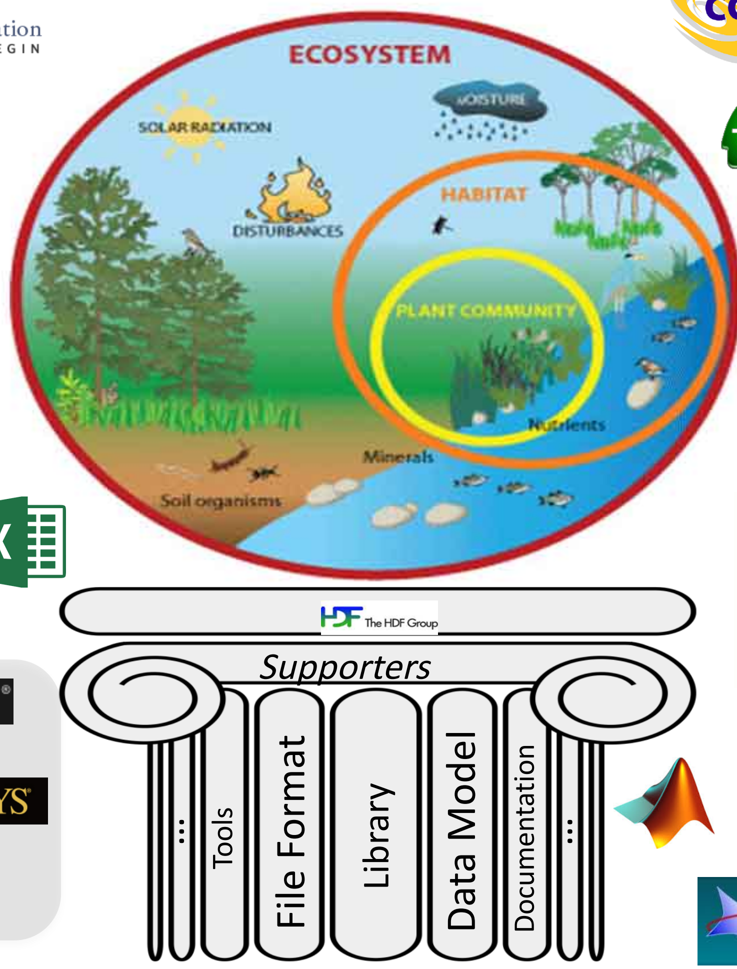
https://tinyurl.com/uoxkwaq

HDF5 is designed for…

- High volume and complex data
  - HDF5 files of GBs sizes are common
- Every size and type of system (portable)
  - Works on from embedded systems, desktops and laptops to exascale systems
- Flexible, efficient storage and I/O
  - See variety of backing store
- Enabling applications to evolve in their use of HDF5 and to accommodate new models
  - Data can be added, removed and reorganized in the file
- Supporting long-term data preservation
  - Petabytes of remote sensing data including data for long term climate research is in NASA archives now

https://tinyurl.com/uoxkwaq

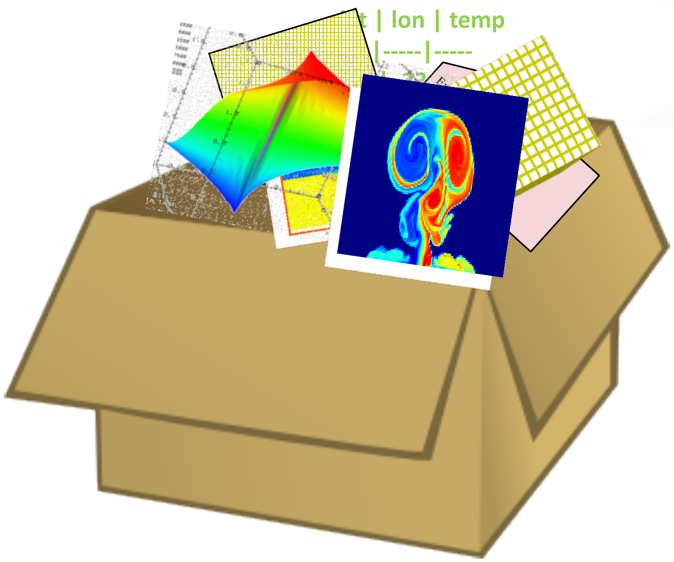
HDF5 Ecosystem



HDF5 DATA MODEL

HDF5 File

An HDF5 file is a **container** that holds data objects.



https://tinyurl.com/uoxkwaq

HDF5 Data Model

Dataset

Link

Group

Datatype

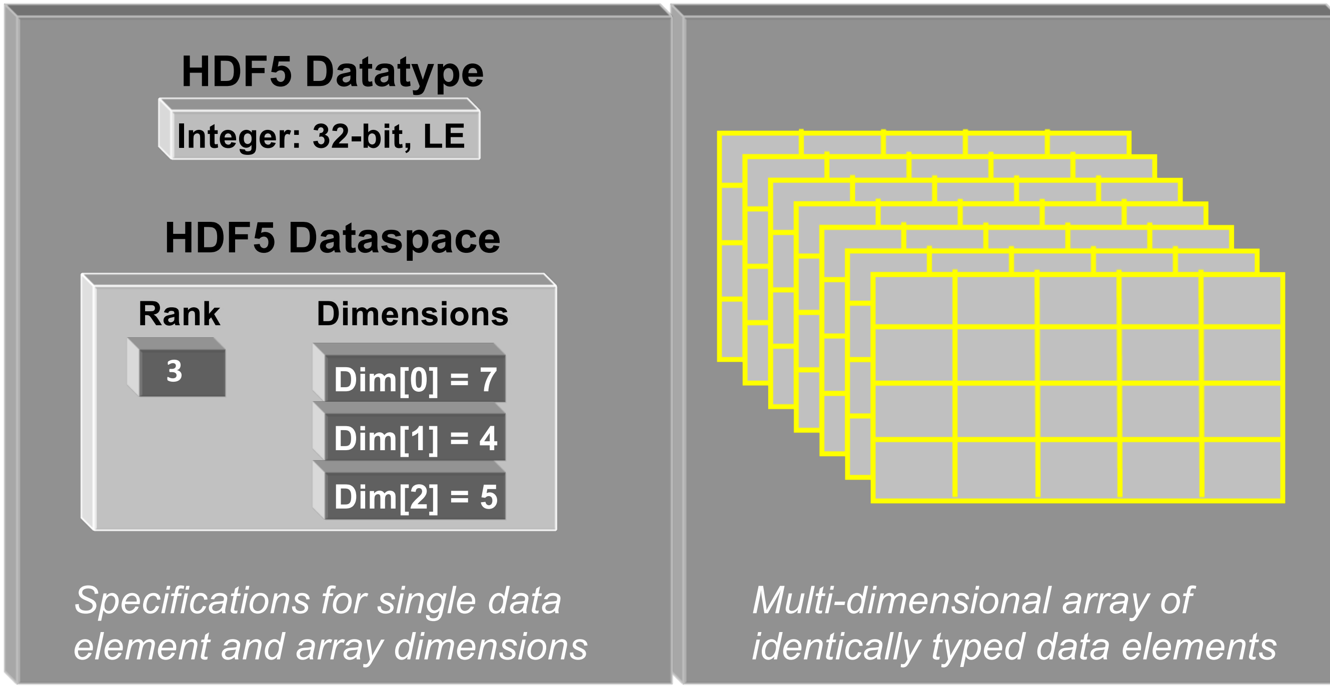
Attribute

Dataspace

File

https://tinyurl.com/uoxkwaq

HDF5 Dataset



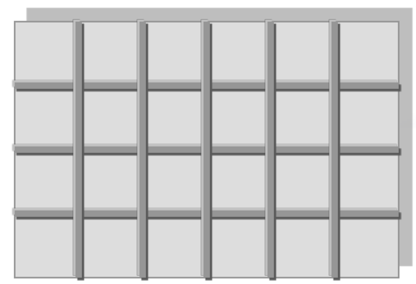
- HDF5 datasets **organize and contain** data elements
  - HDF5 datatype describes individual data elements
  - HDF5 dataspace describes the logical layout of the data elements

HDF5 Dataspace

Two roles:

<u>Spatial information for Datasets and Attributes</u>

- Empty sets and scalar values
- Multidimensional arrays
  - Rank and dimensions
- Permanent part of object definition



Rank = 2

Dimensions = 4 x 6

<u>Partial I/O</u>: Dataspace and selection describe application's data buffer and data elements participating in I/O

Rank = 1

Dimension = 10

https://tinyurl.com/uoxkwaq

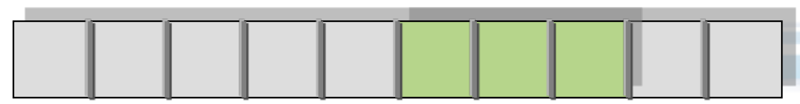
How to describe a subset in HDF5?

- Before writing and reading a subset of data one has to describe it to the HDF5 Library.
- HDF5 APIs and documentation refer to a subset as a "selection" or "hyperslab selection".
- If specified, HDF5 performs I/O on a selection *only* and not on all elements of a dataset.

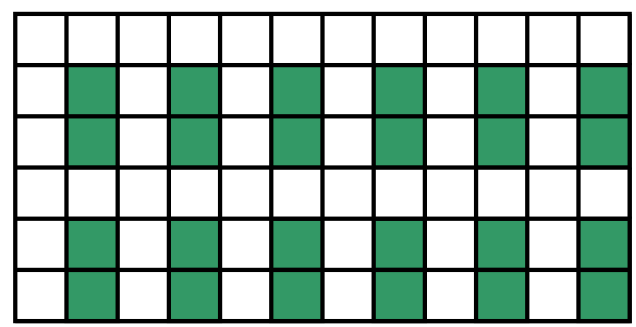
https://tinyurl.com/uoxkwaq

Describing elements for I/O: HDF5 Hyperslab

- *Everything is "measured" in number of elements; 0-based*
- Example 1-dim:
  - Start - starting location of a hyperslab (5)
  - Block - block size (3)



- Example 2-dim:
  - Start - starting location of a hyperslab (1,1)
  - Stride - number of elements that separate each block (3,2)
  - Count - number of blocks (2,6)
  - Block - block size (2,1)
- All other selections are build using set operations



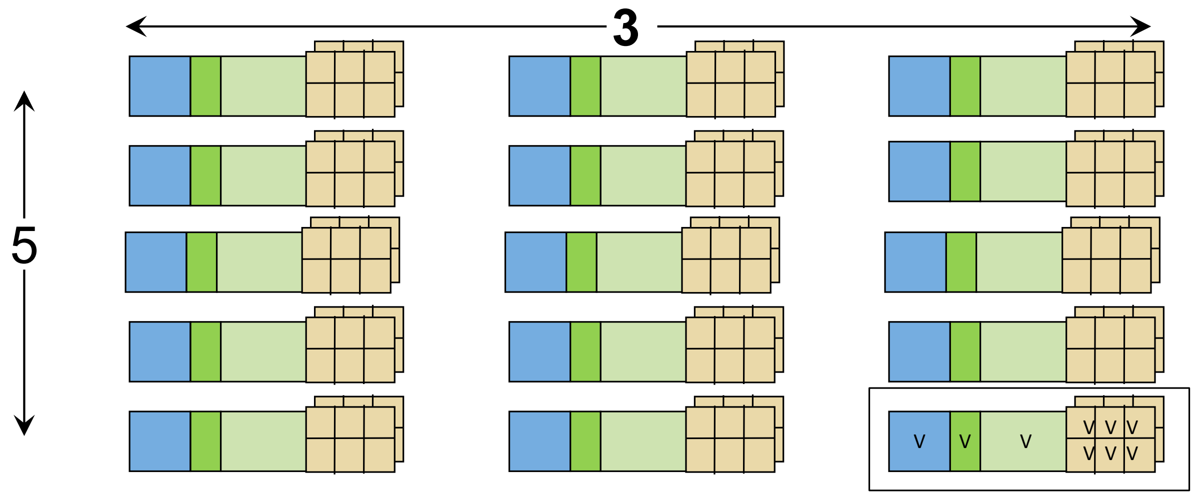
https://tinyurl.com/uoxkwaq

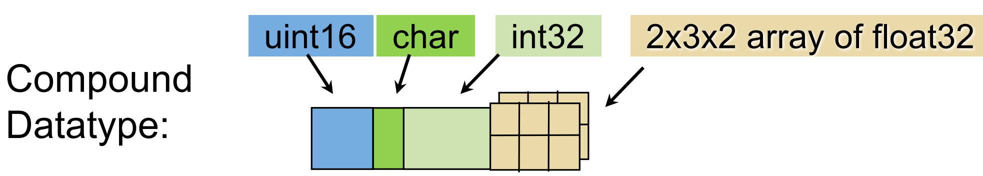
HDF5 Datatypes

- Describe individual data elements in an HDF5 dataset
- Wide range of datatypes is supported
  - Atomic types: integer, floats
  - User-defined (e.g., 12-bit integer, 16-bit float)
  - Enum
  - References to HDF5 objects and selected elements of datasets
  - Variable-length types (e.g., strings, vectors)
  - Compound (similar to C structures or Fortran derived types)
  - Array (similar to matrix)
  - More complex types can be built from types above
- HDF5 library provides predefined symbols to describe atomic datatypes

https://tinyurl.com/uoxkwaq

HDF5 Dataset with Compound Datatype

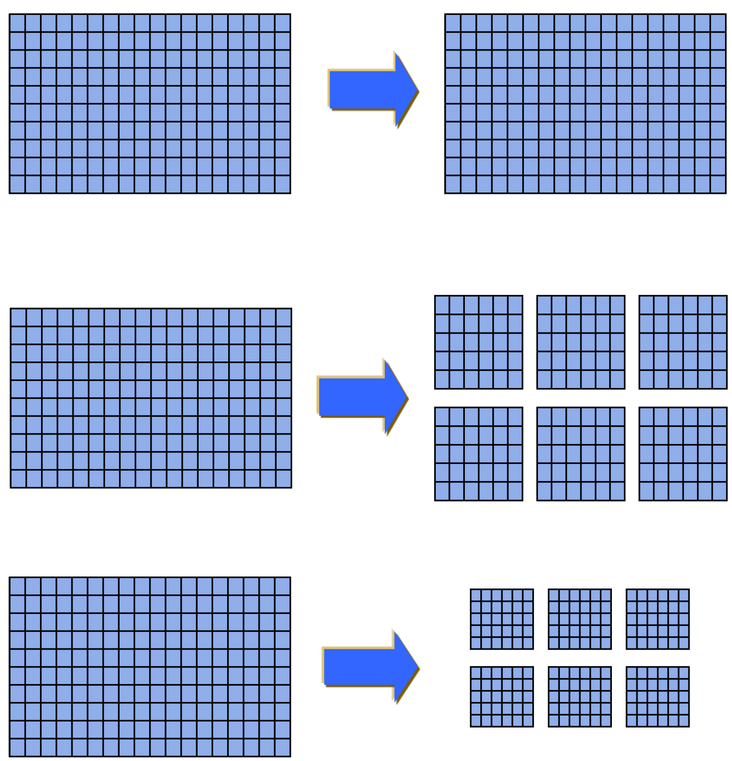




Dataspace: Rank = 2
Dimensions = 5 x 3

How are data elements stored? (1/2)

Buffer in memory | Data in the file



Contiguous (default): Data elements stored physically adjacent to each other

Chunked: Better access time for subsets; extendible

Chunked & Compressed: Improves storage efficiency, transmission speed

https://tinyurl.com/uoxkwaq

Compression and filters in HDF5

- GZIP and SZIP (free version is available from German Climate Computing Center)
- Other compression methods registered with The HDF Group at https://portal.hdfgroup.org/display/support/Contributions#Contributions-filters
  - BZIP2, JPEG, LZF, BLOSC, MAFISC, LZ4, Bitshuffle, and ZFP, etc.
    - Listed above are available as dynamically loaded plugins
    - See https://www.hdfgroup.org/downloads/hdf5/
- Filters:
  - Fletcher32 (checksum)
  - Shuffle
  - Scale+offset
  - n-bit

https://tinyurl.com/uoxkwaq

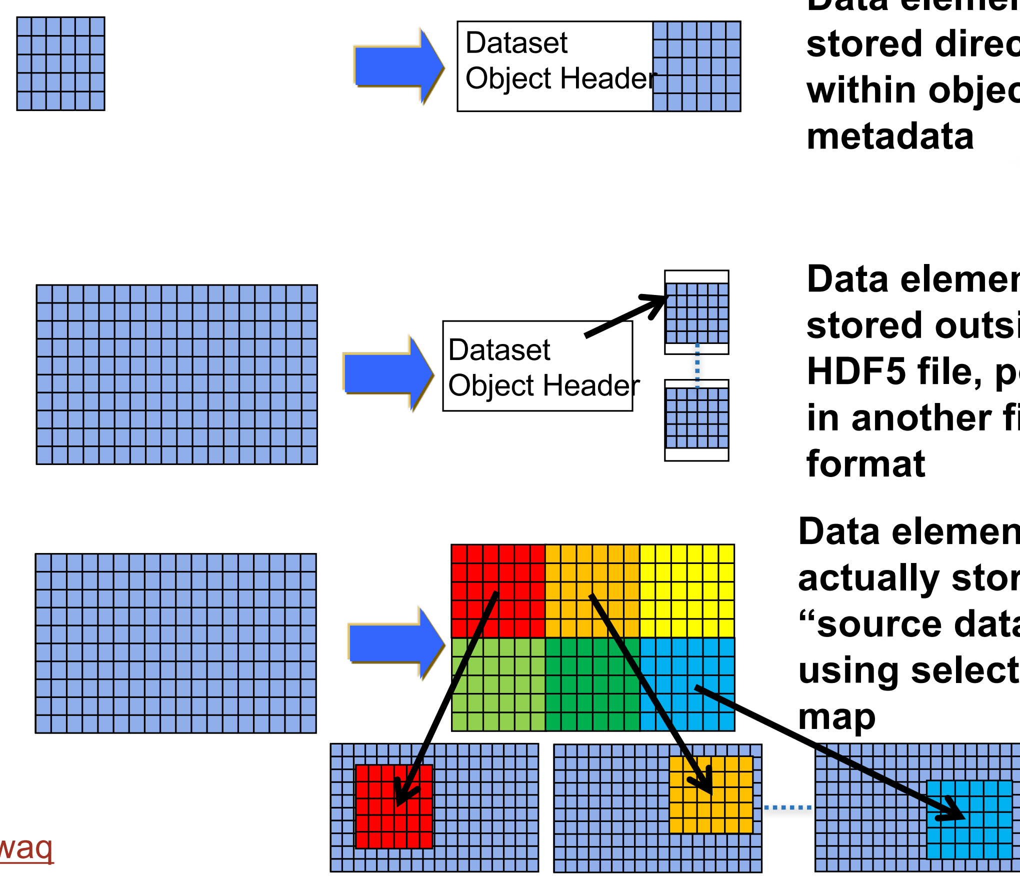
How are data elements stored? (2/2)

Buffer in memory | Data in the file

Compact

External

Virtual



Data elements stored directly within object's metadata

Data elements stored outside the HDF5 file, possibly in another file format

Data elements actually stored in "source datasets", using selections to map

https://tinyurl.com/uoxkwaq

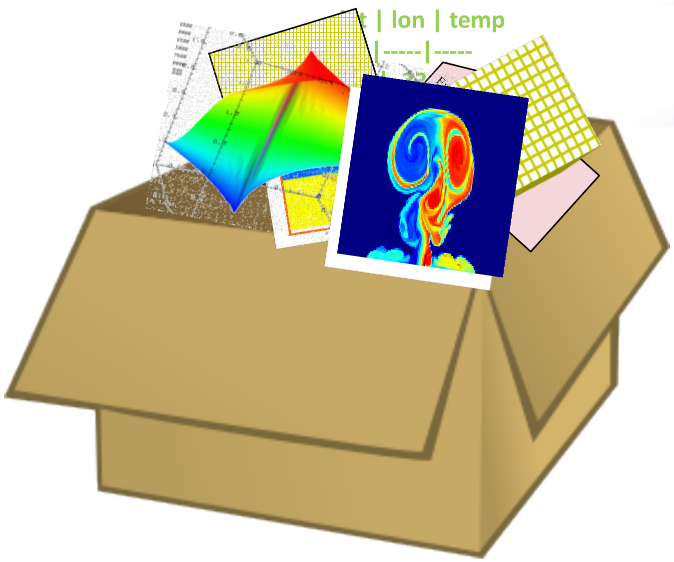
HDF5 Attributes

- Attributes “decorate” HDF5 objects
- Contain *user-defined* metadata
- Similar to Key-Values:
  – Have a unique name (for that object) and a value
- Analogous to a dataset
  – “Value” is described by a datatype and a dataspace
  – Do not support partial I/O operations; nor can they be compressed or extended

https://tinyurl.com/uoxkwaq

HDF5 File

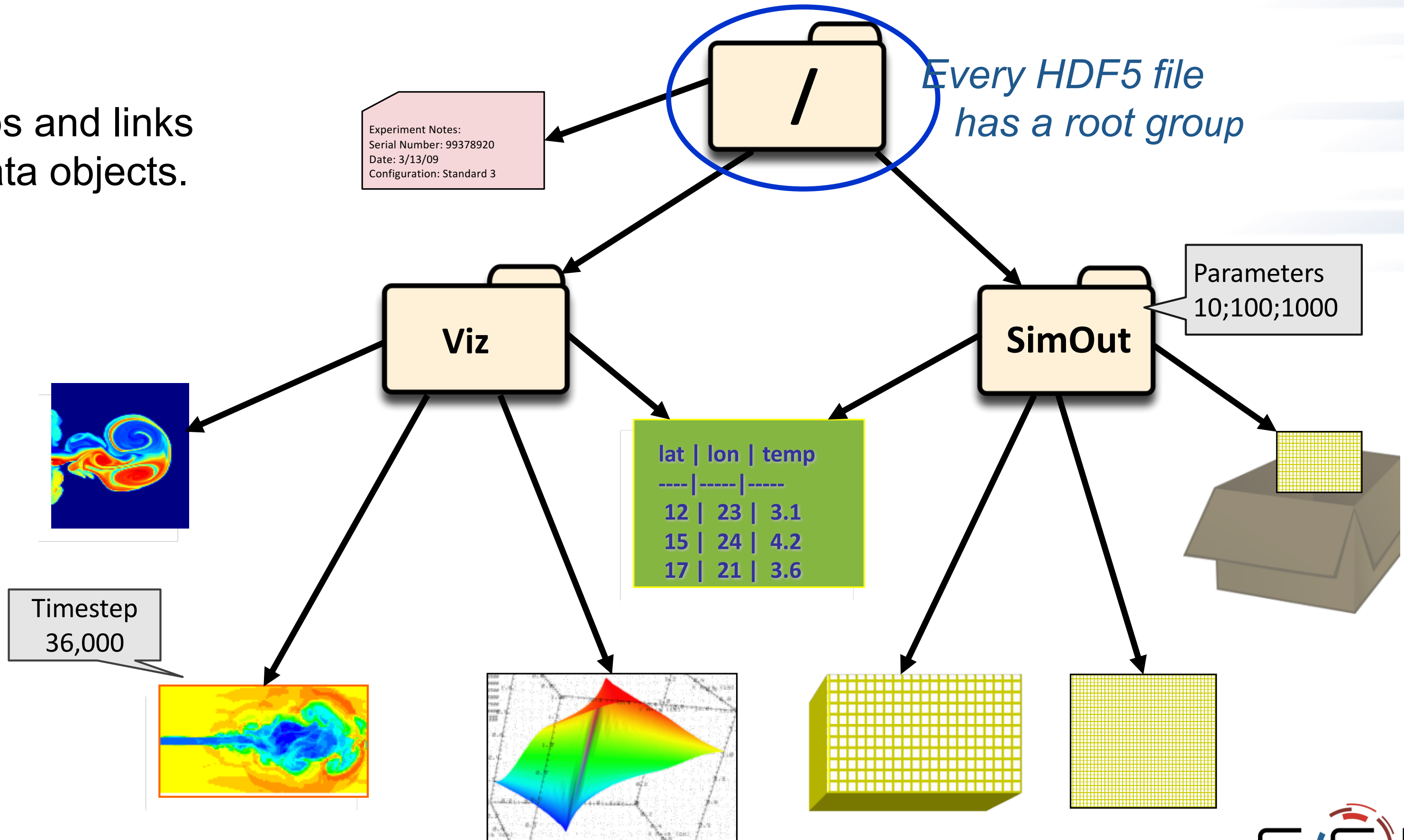
An HDF5 file is a **structured container** that holds data objects.



https://tinyurl.com/uoxkwaq

HDF5 Groups and Links

HDF5 groups and links **organize** data objects.



HDF5 SOFTWARE AND ARCHITECTURE

HDF5 Software

HDF5 home page: http://hdfgroup.org/HDF5/

- Latest release: HDF5 1.10.6 (1.12.0 February 2020)

HDF5 source code:

- Written in C, and includes optional C++, Fortran, Java APIs, and High Level APIs
- Contains command-line utilities (h5dump, h5repack, h5diff, ..) and compile scripts

HDF5 pre-built binaries:

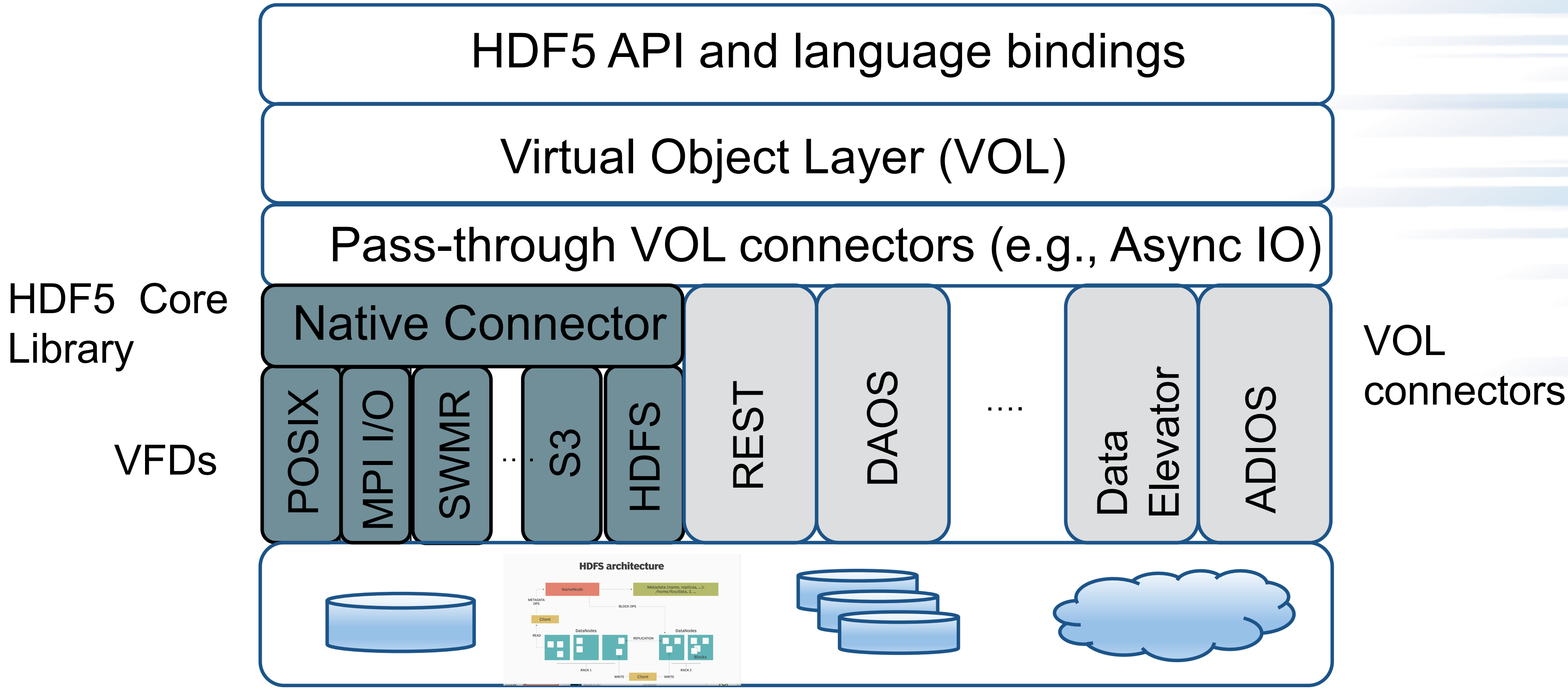
- When possible, include C, C++, Fortran, Java, and High Level libraries. Check ./lib/libhdf5.settings file.
- Built with the SZIP and ZLIB external libraries

3rd party software:

- h5py (Python)
- http://h5cpp.org/ (Contemporary C++ including support for MPI I/O)

https://tinyurl.com/uoxkwaq

HDF5 Library Architecture (1.12.0)



https://tinyurl.com/uoxkwaq

HDF5 PROGRAMMING MODEL AND API

The General HDF5 API

- C, FORTRAN, Java, and C++
- C routines begin with prefix: H5?

  ? corresponds to the type of object the function acts on

Example Functions:

H5D :	**D**ataset interface	*e.g.,* **H5Dread**
H5F :	**F**ile interface	*e.g.,* **H5Fopen**
H5S :	data**S**pace interface	*e.g.,* **H5Sclose**

- Other language wrappers follow the same trend
- There are more than 300 APIs – pne can start just with

https://tinyurl.com/uoxkwaq

General Programming Paradigm

- Properties of object are optionally defined
  - Creation properties (e.g., use chunking storage)
  - Access properties (e.g., using MPI I/O driver to access file)
- Object is opened or created
  - Creation properties applied
  - Access properties applied
  - Supporting objects are defined (datatype, dataspace)
- Object is accessed, possibly many times
  - Access property can be changed
- Object is closed

https://tinyurl.com/uoxkwaq

Standard HDF5 program "Skeleton"

H5Fcreate (H5Fopen)	create (open) File
H5Screate_simple/H5Screate	create dataSpace
H5Dcreate (H5Dopen)	create (open) Dataset
H5Dread, H5Dwrite	**access Dataset**
H5Dclose	close Dataset
H5Sclose	close dataSpace
H5Fclose	close File

https://tinyurl.com/uoxkwaq

GENERAL BEST PRACTICES

Memory considerations

- **Open Objects**
  - Open objects use up memory. The amount of memory used may be substantial when many objects are left open. Application should:
    - Delay opening of files and datasets as close to their actual use as is feasible.
    - Close files and datasets as soon as their use is completed.
    - If writing to a portion of a dataset in a loop, be sure to close the dataspace with each iteration, as this can cause a large temporary "memory leak".
- There are APIs to determine if objects are left open. H5Fget_obj_count will get the number of open objects in the file, and H5Fget_obj_ids will return a list of the open object identifiers.

https://tinyurl.com/uoxkwaq

Memory considerations (cont'd)

- **Metadata Cache**
  - The metadata cache can also memory usage. Modify the metadata cache settings to minimize the size and growth of the cache as much as possible without decreasing performance.
  - By default the metadata cache is 2 MB in size, and it can be allowed to increase to a maximum of 32 MB per file. The metadata cache can be disabled or modified. Memory used for the cache is not released until the datasets or file are closed.
  - https://portal.hdfgroup.org/display/HDF5/Metadata+Caching+in+HDF5
  - See https://portal.hdfgroup.org/display/HDF5/H5P_GET_MDC_CONFIG to get default MD cache configurations and https://portal.hdfgroup.org/display/HDF5/H5P_SET_MDC_CONFIG to set new configuration
  - To keep MD cache from growing consider evicting objects on close https://portal.hdfgroup.org/display/HDF5/H5P_SET_EVICT_ON_CLOSE

https://tinyurl.com/uoxkwaq

HDF5 Dataset I/O

- Issue large I/O requests
  - At least as large as file system block size
- Avoid **datatype conversion**
  - Use the same data type in the file as in memory
- Avoid **dataspace conversion**
  - One dimensional buffer in memory to two dimensional array in the file

https://tinyurl.com/uoxkwaq

HDF5 Dataset - Storage

- Use **contiguous storage** if no data will be added and compression is not used
  - Data will no be cached by HDF5
- Use **compact** storage when working with small data (<64K)
  - Data becomes part of HDF5 internal metadata and is cached (metadata cache)
- If you have **binary files** that you would like to convert to HDF5 consider **external storage** and use h5repack tool
- Avoid data duplication to reduce file sizes
  - Use links to point to datasets stored in the same or external HDF5 file
  - Use VDS to point to data stored in other HDF5 datasets

https://tinyurl.com/uoxkwaq

HDF5 Dataset – Chunked Storage

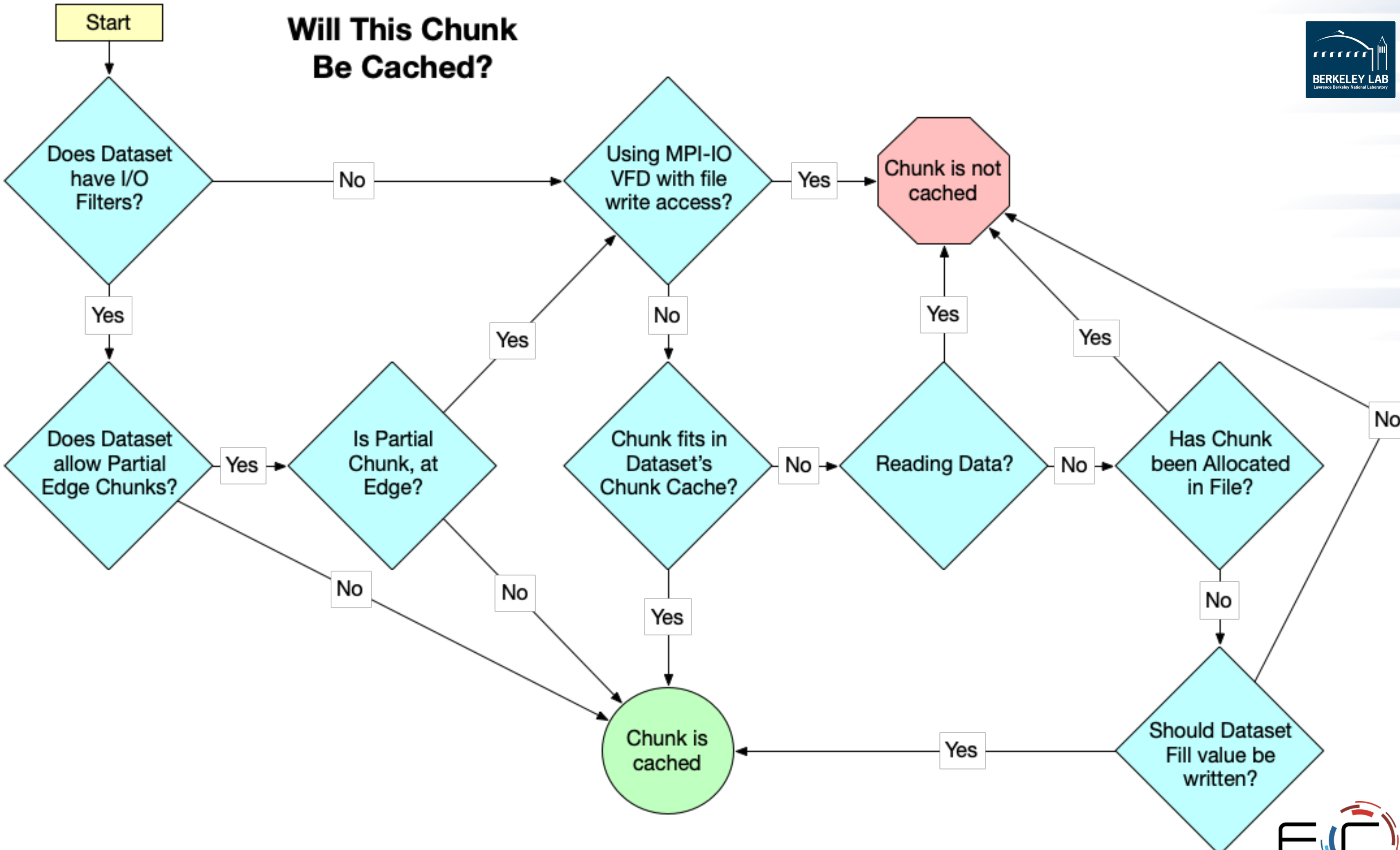
- Chunking is required when using extendibility and/or compression and other filters
- **I/O** is always performed **on a whole chunk**
- Understand how **chunking cache** works https://portal.hdfgroup.org/display/HDF5/Chunking+in+HDF5 and consider
  - Do you access the same chunk often?
  - What is the best chunk size (especially when using compression)?
  - Do you need to adjust chunk cache size (1 MB default; can be set up per file or per dataset)?
  - `H5Pset_chunk_cache` sets raw data chunk cache parameters for a dataset
    - `H5Pset_chunk_cache (dapl, …);`
  - `H5Pset_cache` sets raw data chunk cache parameters for all datasets in a file
    - `H5Pset_cache (fapl, …);`
  - Other parameters to control chunk cache

https://tinyurl.com/uoxkwaq

HDF5 Dataset – Chunked Storage (cont'd)

- Cache size is important when doing partial I/O to avoid many I/O operations
- With the 1 MB cache size, a chunk will not fit into the cache
  - All writes to the dataset must be immediately written to disk
  - With compression, the entire chunk must be read and rewritten every time a part of the chunk is written to
    - Data must also be decompressed and recompressed each time
    - Non sequential writes could result in a larger file
- Without compression, the entire chunk must be written when it is first written to the file.
- To write multiple chunks at once increase the cache size to hold more chunks

https://tinyurl.com/uoxkwaq



https://tinyurl.com/uoxkwaq

Effect of chunk cache size on read

- When compression is enabled, the library must always read entire chunk once for each call to **`H5Dread`** (unless it is in cache)
- When compression is disabled, the library's behavior depends on the cache size relative to the chunk size.
  - If the chunk fits in cache, the library reads entire chunk once for each call to **`H5Dread`**
  - If the chunk does not fit in cache, the library reads only the data that is selected
    - More read operations, especially if the read plane does not include the fastest changing dimension
    - Less total data read

https://tinyurl.com/uoxkwaq

Effect of cache size on read (cont'd)

- On read cache size does not matter when compression is enabled.
- Without compression, the cache must be large enough to hold all of the chunks to get good performance.
- The optimum cache size depends on the exact shape of the data, as well as the hardware, as well as access pattern.

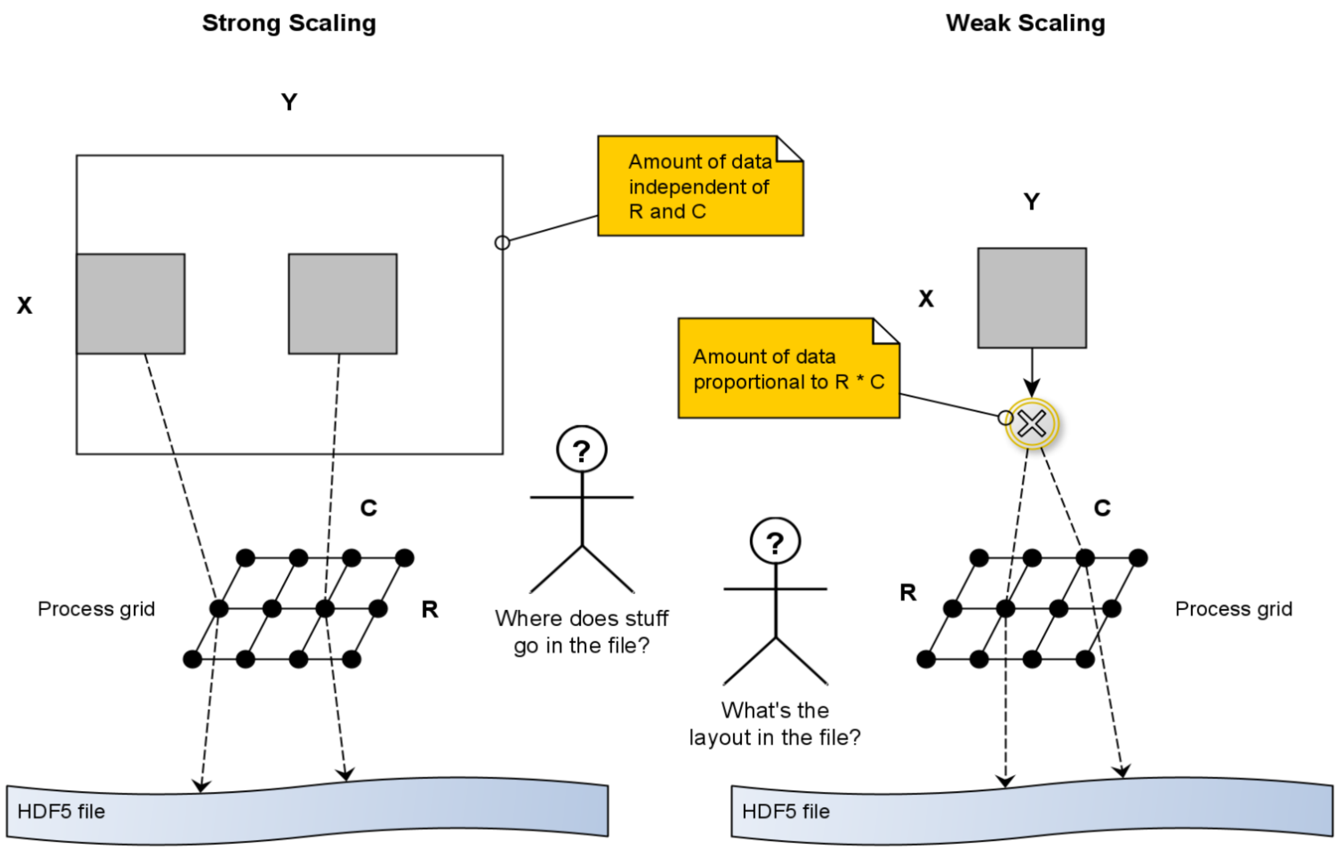
https://tinyurl.com/uoxkwaq

What is the best way to organize data in HDF5 file?

- It depends on your goals!
- Ask yourself
  - Do I need performance on write, read or both?
  - Do I read all data (variables) at once?
  - Do I want to use visualization tool that requires special organization of data?
  - ?

https://tinyurl.com/uoxkwaq

How to organize data in HDF5 file?



I/O Test Pseudocode (T=20, A=500, X=100, Y=200)

START the clock

ACROSS P processes arranged in a **R** x **C** process grid

FOREACH <u>step</u> 1 .. **T**

FOREACH <u>count</u> 1 .. **A**

CREATE a `double` ARRAY of size **[X,Y]** | **[RX,CY]** (strong | weak)

(WRITE | READ) the ARRAY (to | from) a single HDF5 file

END

END

END

STOP the clock and REPORT the time / throughput

https://tinyurl.com/uoxkwaq

Basic data organization options & variations

T - steps, **A** - arrays, **X** - rows, **Y** - columns (strong scaling)

1. All data goes into a single 4D dataset **[T, A, X, Y]** or **[A, T, X, Y]**
2. A separate dataset for each step, i.e., **T** 3D datasets **[A, X, Y]**
3. A separate step series for each array, i.e., **A** 3D datasets **[T, X, Y]**
4. A separate dataset for each array for each step, i.e., **T*A** 2D datasets **[X,Y]**

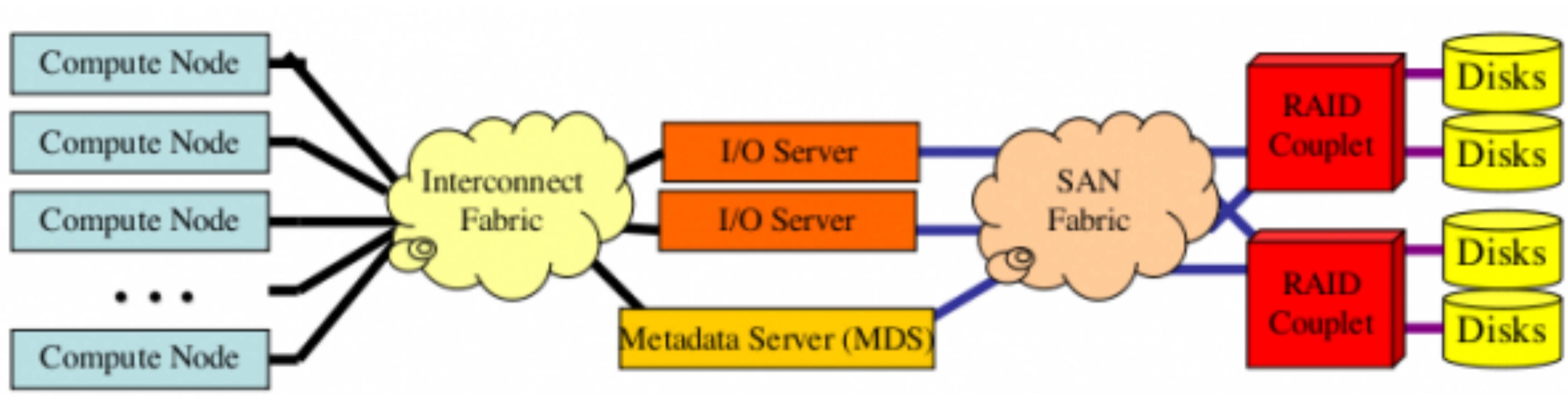
<u>Variations</u>:

- If all parameters are known in advance, you can get away w/ fixed layout
- If **T** is unknown, you can
  - Still use fixed layout under 2. And 4.
  - Pad (chunk size > 1 in the slowest dimension) your allocations under 3.
- Add optical sugar through the use of groups
- More than a dozen ways to implement this, depending on assumptions

PARALLEL I/O WITH HDF5

Quincey Koziol and Scot Breitenfeld

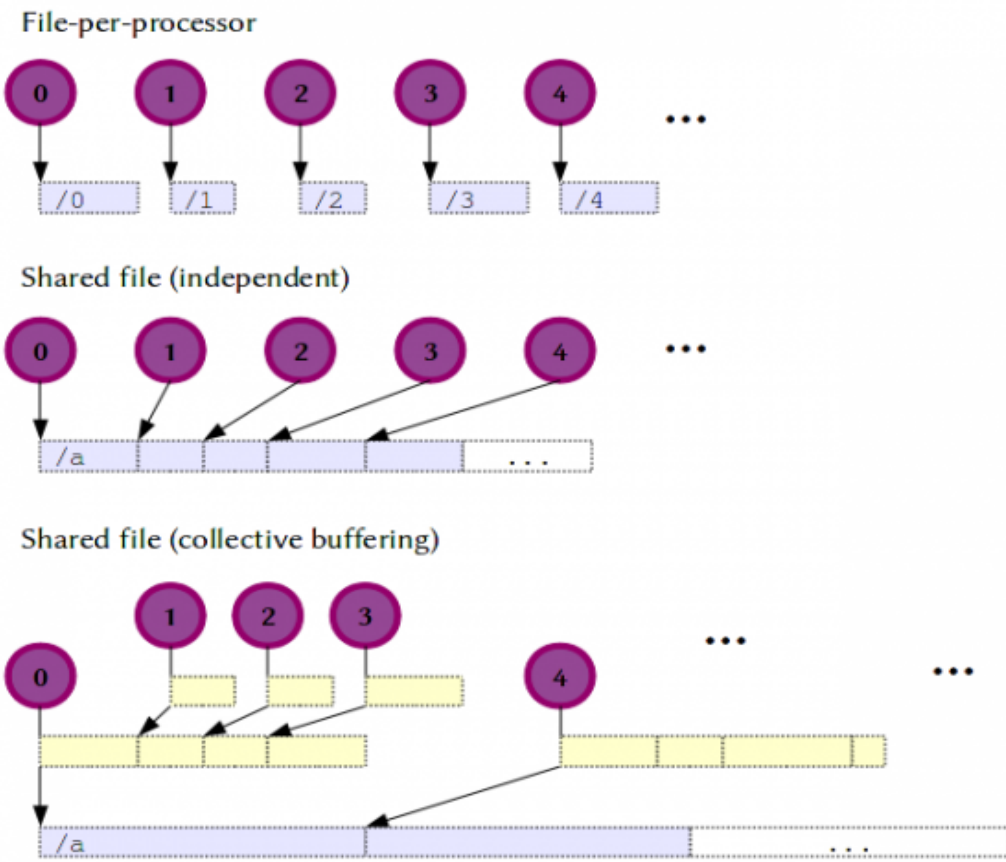
Parallel File Systems – Lustre, GPFS, etc.



- Scalable, POSIX-compliant file systems designed for large, distributed-memory systems
- Uses a client-server model with separate servers for file metadata and file content

https://tinyurl.com/uoxkwaq

Types of Application I/O to Parallel File Systems



https://tinyurl.com/uoxkwaq

Why Parallel HDF5?

- Take advantage of high-performance parallel I/O while reducing complexity
  - Use a well-defined high-level I/O layer instead of POSIX or MPI-IO
  - Use only a single or a few shared files
    - “Friends don’t let friends use file-per-process!” ☺
- Code, Performance and Data Portability
  - Rely on HDF5 to optimize for underlying storage system

https://tinyurl.com/uoxkwaq

What We'll Cover Here

- Parallel vs. Serial HDF5
- Implementation Layers
- HDF5 files (= composites of data & metadata) in a parallel file system
- Parallel HDF5 (PHDF5) I/O modes: collective vs. independent
- Data and Metadata I/O

https://tinyurl.com/uoxkwaq

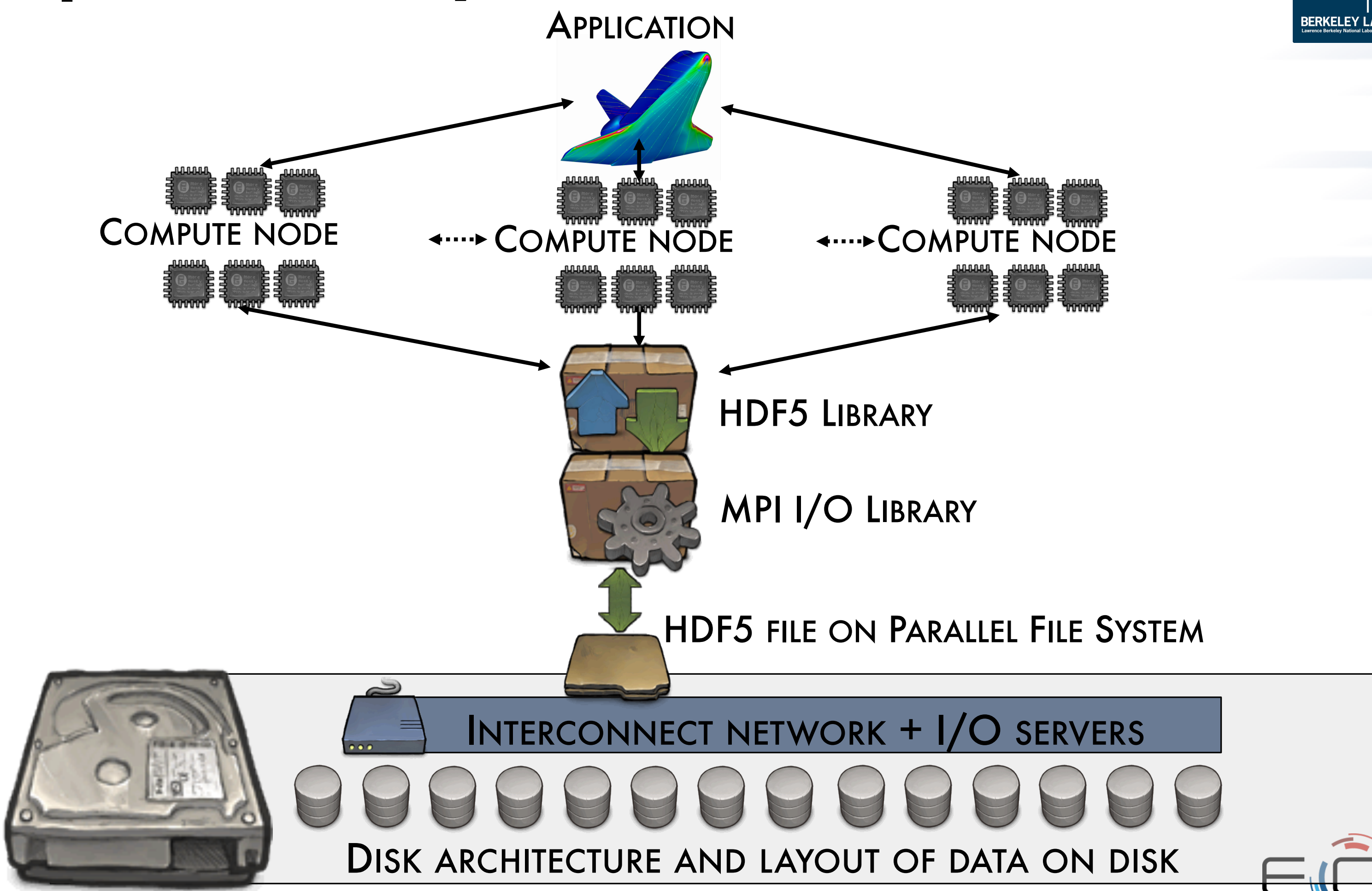
Terminology

- DATA – “problem-size” data, e.g., large arrays
- METADATA – is an overloaded term
- In this presentation:

  Metadata “=“ HDF5 metadata
  - For each piece of application metadata, there are many associated pieces of HDF5 metadata
  - There are also other sources of HDF5 metadata
    - Chunk indices, heaps to store group links and indices to look them up, object headers, etc.

https://tinyurl.com/uoxkwaq

PHDF5 implementation layers



(MPI-)Parallel vs. Serial HDF5

- PHDF5 allows multiple MPI processes in an MPI application to perform I/O to a single HDF5 file
- Uses a standard parallel I/O interface (MPI-IO)
- Portable to different platforms
- PHDF5 files ARE HDF5 files conforming to the HDF5 file format specification
- The PHDF5 API consists of:
  - The standard HDF5 API
  - A few extra knobs and calls
  - A parallel “etiquette”

https://tinyurl.com/uoxkwaq

General Programming model

- Each process defines memory and file hyperslabs using `H5Sselect_hyperslab`
- Each process executes a write/read call using hyperslabs defined, which can be either collective or independent
- The hyperslab parameters define the portion of the dataset to write to
  - Contiguous hyperslab
  - Regularly spaced data (column or row)
  - Pattern
  - Blocks

https://tinyurl.com/uoxkwaq

Example of a PHDF5 C Program

Starting with a simple serial HDF5 program:

```
file_id = H5Fcreate(FNAME, …, H5P_DEFAULT);
space_id = H5Screate_simple(…);
dset_id = H5Dcreate(file_id, DNAME, H5T_NATIVE_INT, space_id, …);



status = H5Dwrite(dset_id, H5T_NATIVE_INT, …, H5P_DEFAULT);
```


https://tinyurl.com/uoxkwaq

Example of a PHDF5 C Program

A parallel HDF5 program has a few extra calls:

```
MPI_Init(&argc, &argv);
…
fapl_id = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_mpio(fapl_id, comm, info);
file_id = H5Fcreate(FNAME, …, fapl_id);
space_id = H5Screate_simple(…);
dset_id = H5Dcreate(file_id, DNAME, H5T_NATIVE_INT, space_id, …);
xf_id = H5Pcreate(H5P_DATASET_XFER);
H5Pset_dxpl_mpio(xf_id, H5FD_MPIO_COLLECTIVE);
status = H5Dwrite(dset_id, H5T_NATIVE_INT, …, xf_id);
…
MPI_Finalize();
```


https://tinyurl.com/uoxkwaq

PHDF5 Etiquette

- PHDF5 opens a shared file with an MPI communicator
  - Returns a file ID (as usual)
  - All future access to the file via that file ID
- Different files can be opened via different communicators
- All processes must participate in collective PHDF5 APIs
- All HDF5 APIs that modify the HDF5 namespace and structural metadata are collective!
  - File ops., group structure, dataset dimensions, object life-cycle, etc.

https://support.hdfgroup.org/HDF5/doc/RM/CollectiveCalls.html

https://tinyurl.com/uoxkwaq

Collective vs. Independent Operations

- MPI Collective Operations:
  - All processes of the communicator must participate, in the right order. E.g.,

	Process1	Process2	
	call A(); call B();	call A(); call B();	…CORRECT
	call A(); call B();	call B(); call A();	…WRONG

- Collective operations are not necessarily synchronous, nor must they require communication
  - It could be that only internal state for the communicator changes
- Collective I/O attempts to combine multiple smaller independent I/O ops into fewer larger ops; neither mode is preferable *a priori*

https://tinyurl.com/uoxkwaq

Writing and Reading Hyperslabs

- Distributed memory model: data is split among processes
- PHDF5 uses HDF5 hyperslab model
- Each process defines memory and file hyperslabs

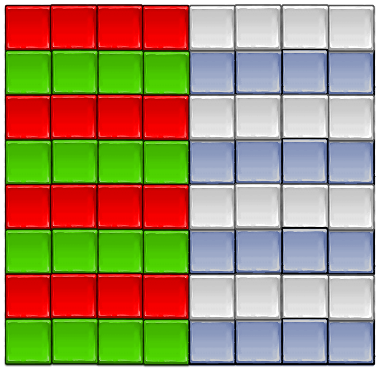
```
H5Sselect_hyperslab(space_id, H5S_SELECT_SET,
                    offset, stride, count, block)
```

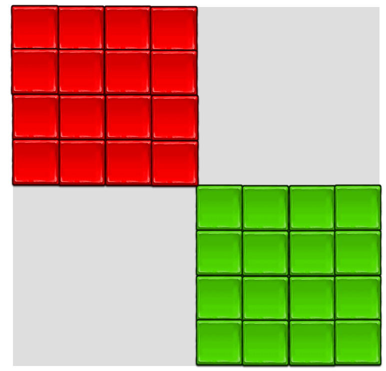
- Each process executes partial write/read call
  - Collective calls
  - Independent calls

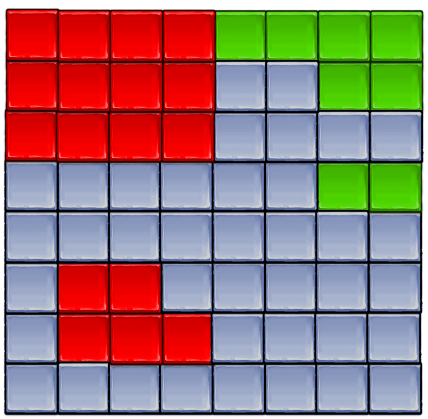
https://tinyurl.com/uoxkwaq

Complex data patterns

HDF5 doesn't have restrictions on data patterns and balance







Irregular hyperslabs created by union operators

```
H5Sselect_hyperslab(space_id, op,
start, stride, count, block )
```


https://tinyurl.com/uoxkwaq

Complex data patterns -- Selection

H5S_SELECT_SET

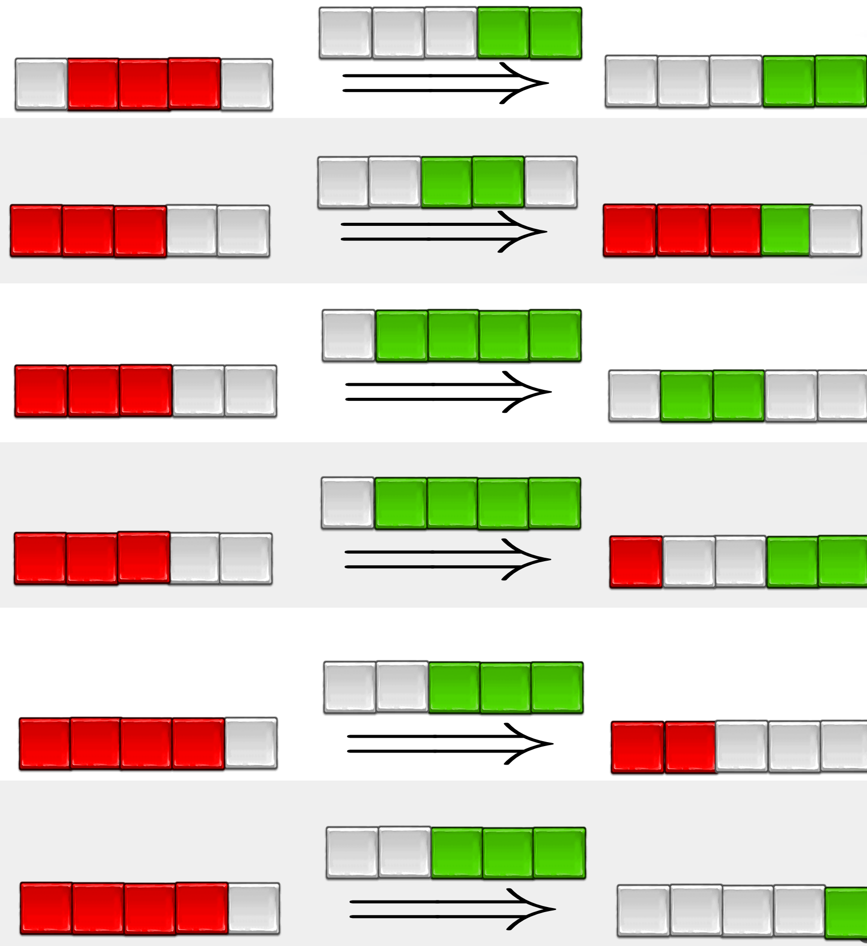
H5S_SELECT_OR

H5S_SELECT_AND

H5S_SELECT_XOR

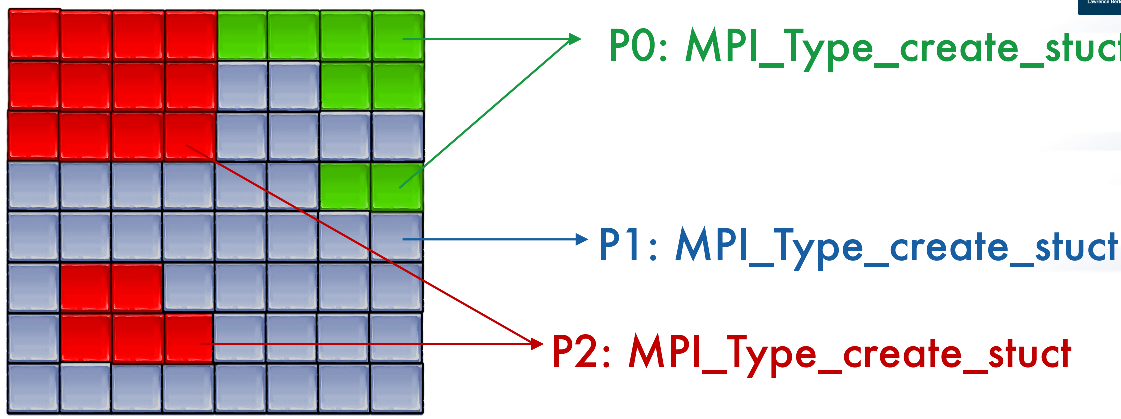
H5S_SELECT_NOTB

H5S_SELECT_NOTA



https://tinyurl.com/uoxkwaq

Examples of irregular selection

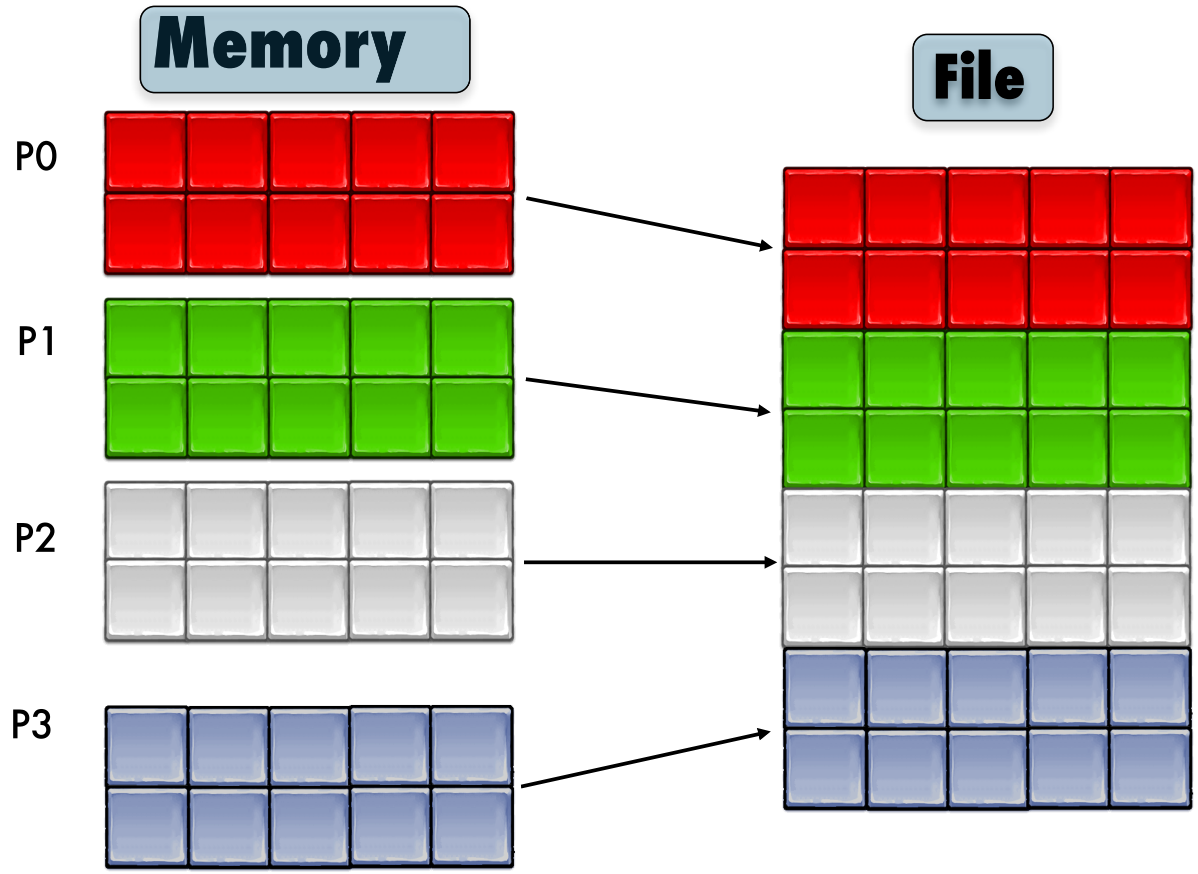


Internally…

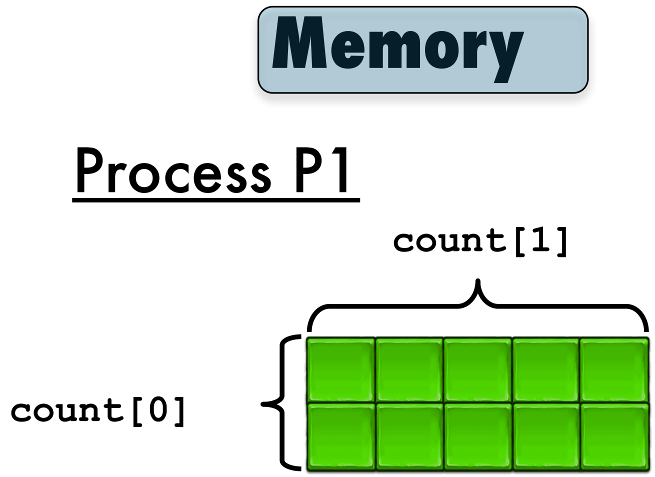
1. The HDF5 library creates an MPI datatype for each lower dimension in the selection
2. It then combines those types into one large structured MPI datatype

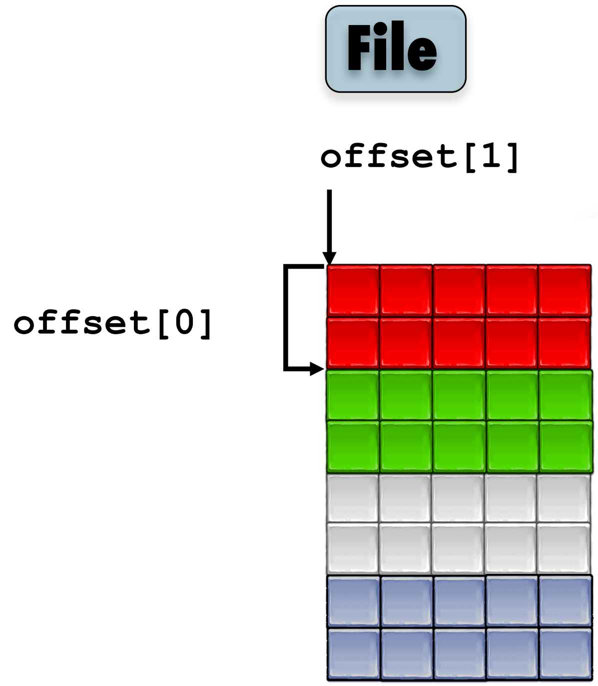
https://tinyurl.com/uoxkwaq

Example 1: Writing dataset by rows



Example 1: *Writing dataset by rows*






```
count[0] = dimsf[0]/mpi_size
count[1] = dimsf[1];
offset[0] = mpi_rank * count[0];  /* = 2 */
offset[1] = 0;
```

Example 1: *Writing dataset by rows*

```
/*
 * Each process defines dataset in memory and
 * writes it to the hyperslab
 * in the file.
 */
 count[0] = dimsf[0]/mpi_size;
 count[1] = dimsf[1];
 offset[0] = mpi_rank * count[0];
 offset[1] = 0;
 memspace = H5Screate_simple(RANK,count,NULL);

/*
 * Select hyperslab in the file.
 */
 filespace = H5Dget_space(dset_id);
 H5Sselect_hyperslab(filespace,
     H5S_SELECT_SET,offset,NULL,count,NULL);
```

C Example: Collective write and read

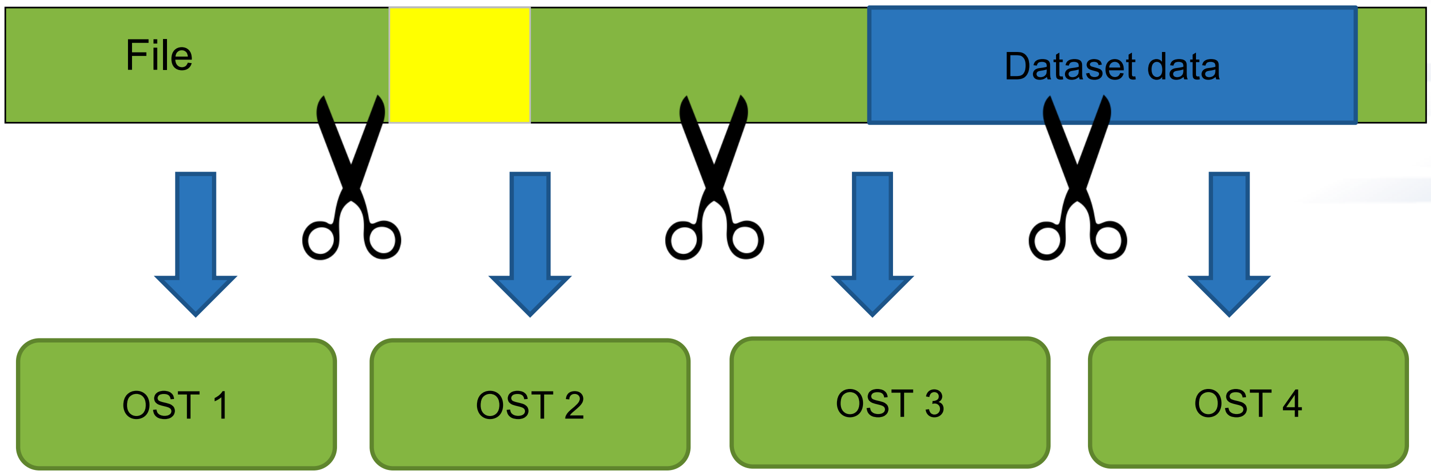
```
/*
 * Create property list for collective dataset write.
 */
plist_id = H5Pcreate(H5P_DATASET_XFER);
H5Pset_dxpl_mpio(plist_id, H5FD_MPIO_COLLECTIVE);

status = H5Dwrite(dset_id, H5T_NATIVE_INT,
                  memspace, filespace, plist_id, data);
```

```
/*
 * Collective dataset read.
 */

status = H5Dread(dset_id, H5T_NATIVE_INT,
                  memspace, filespace, plist_id, data);

```

Writing by rows: *Output of h5dump*

```
HDF5 "SDS_row.h5" {
GROUP "/" {
   DATASET "IntArray" {
      DATATYPE  H5T_STD_I32BE
      DATASPACE  SIMPLE { ( 8, 5 ) / ( 8, 5 ) }
      DATA {
         10, 10, 10, 10, 10,
         10, 10, 10, 10, 10,
         11, 11, 11, 11, 11,
         11, 11, 11, 11, 11,
         12, 12, 12, 12, 12,
         12, 12, 12, 12, 12,
         13, 13, 13, 13, 13,
         13, 13, 13, 13, 13
      }
   }
}
}
```

In a Parallel File System



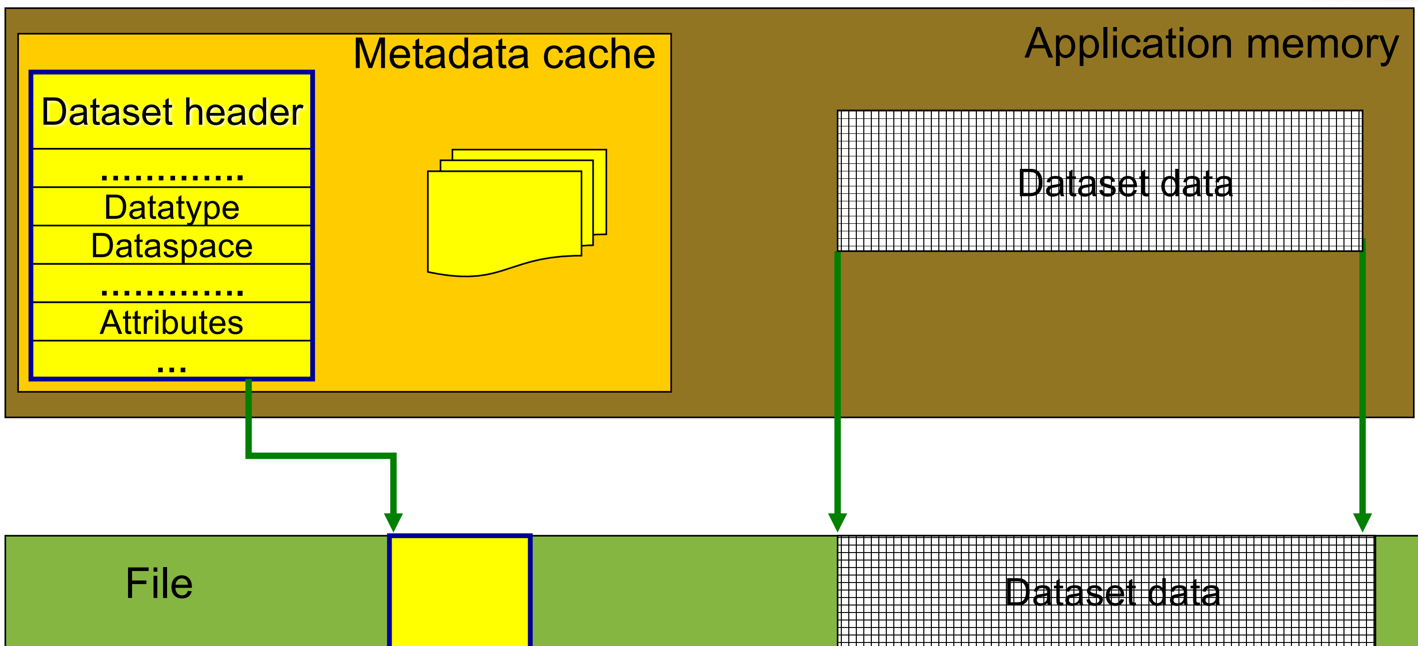
The file is striped over multiple “disks” (e.g. Lustre OSTs) depending on the stripe size and stripe count with which the file was created.

And it gets worse before it gets better…

https://tinyurl.com/uoxkwaq

Contiguous Storage

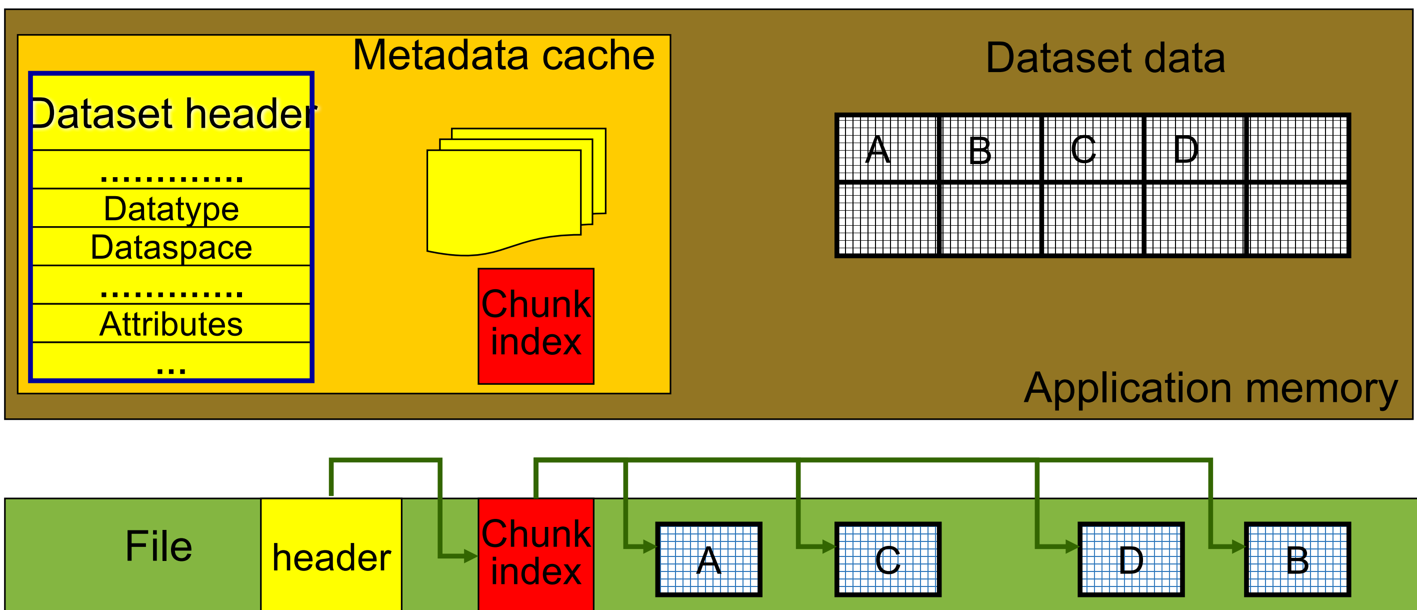
- Metadata header separate from dataset data
- Data stored in one contiguous block in HDF5 file



https://tinyurl.com/uoxkwaq

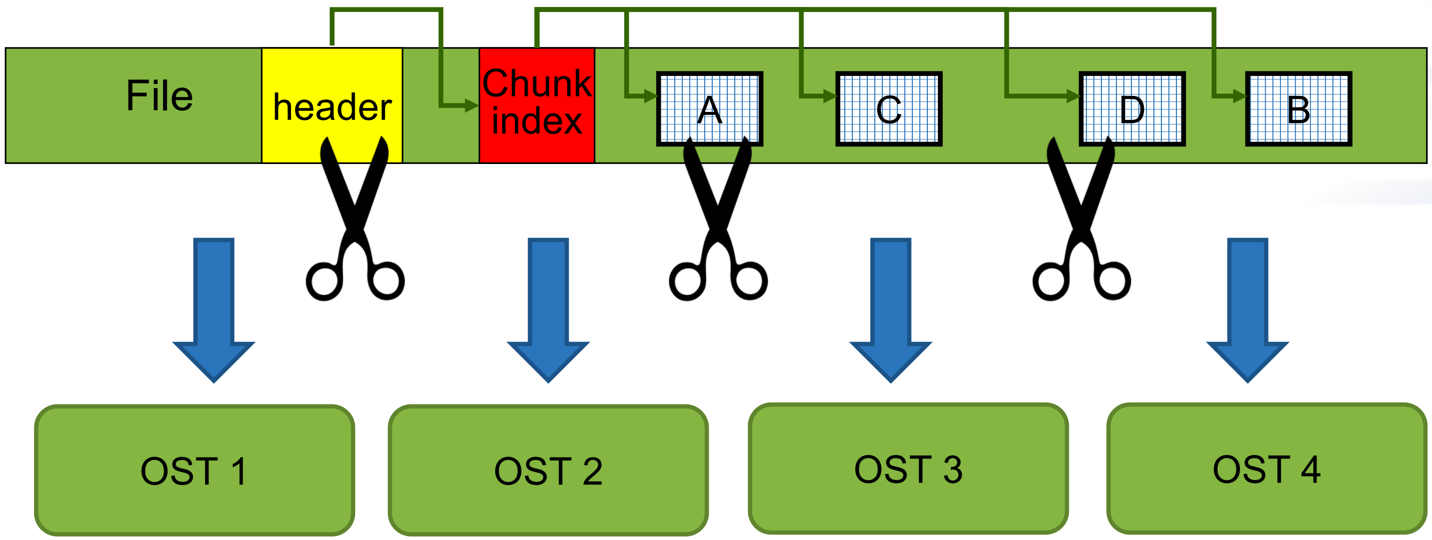
Chunked Storage

- Dataset data is divided into equally sized blocks (chunks).
- Each chunk is stored separately as a contiguous block in HDF5 file.



https://tinyurl.com/uoxkwaq

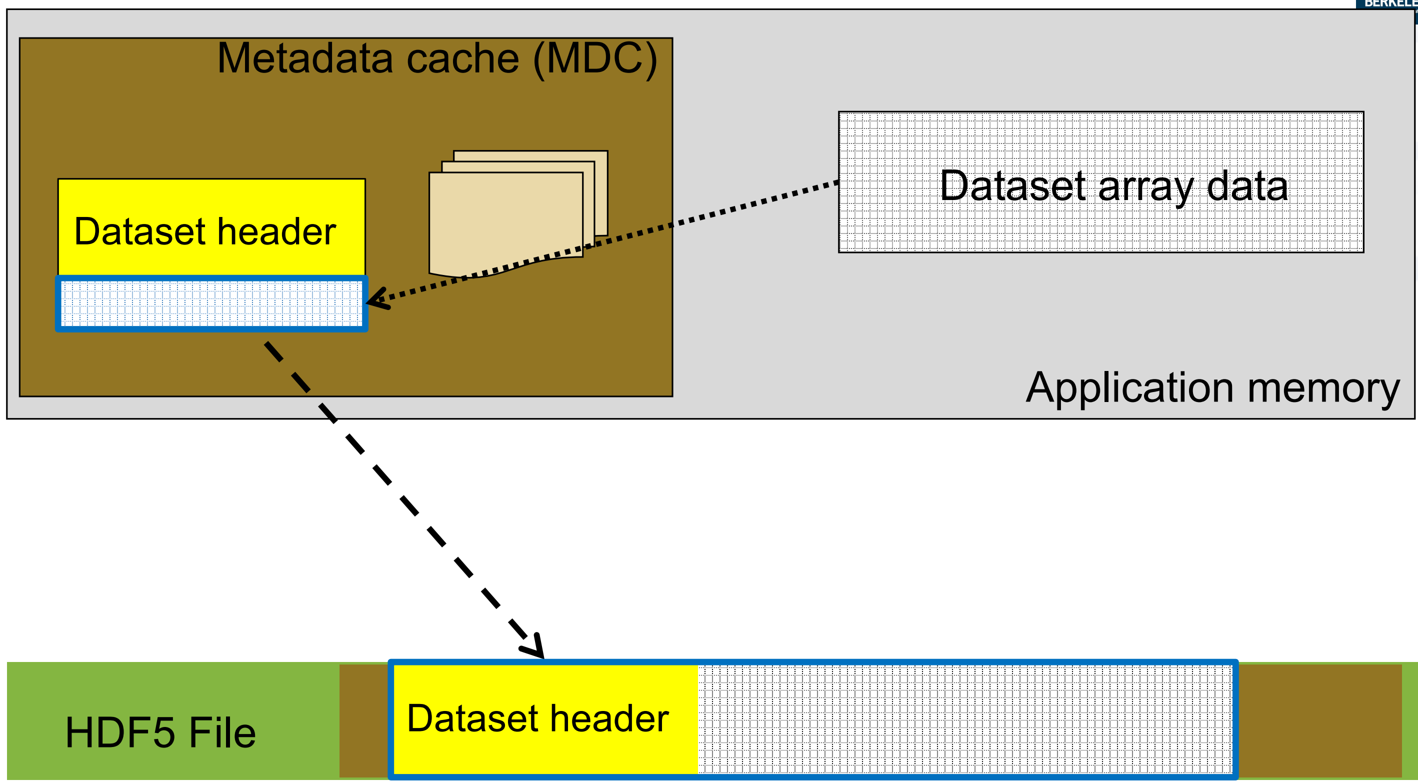
In a Parallel File System



The file is striped over multiple OSTs depending on the stripe size and stripe count with which the file was created.

https://tinyurl.com/uoxkwaq

Compact dataset



Raw data is written when object header is written

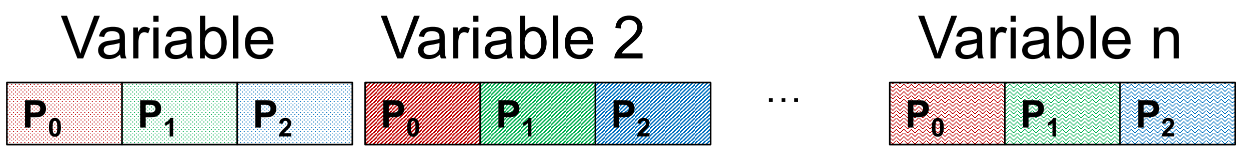
https://tinyurl.com/uoxkwaq

PERFORMANCE TUNING

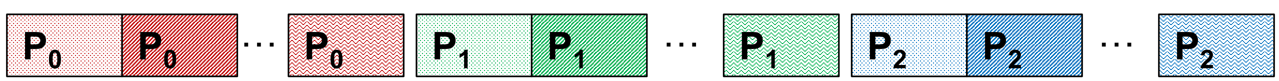
Scot Breitenfeld

Write Pattern Effects

Pattern 1 – General HDF5 pattern



Pattern 2- MPI-IO pattern

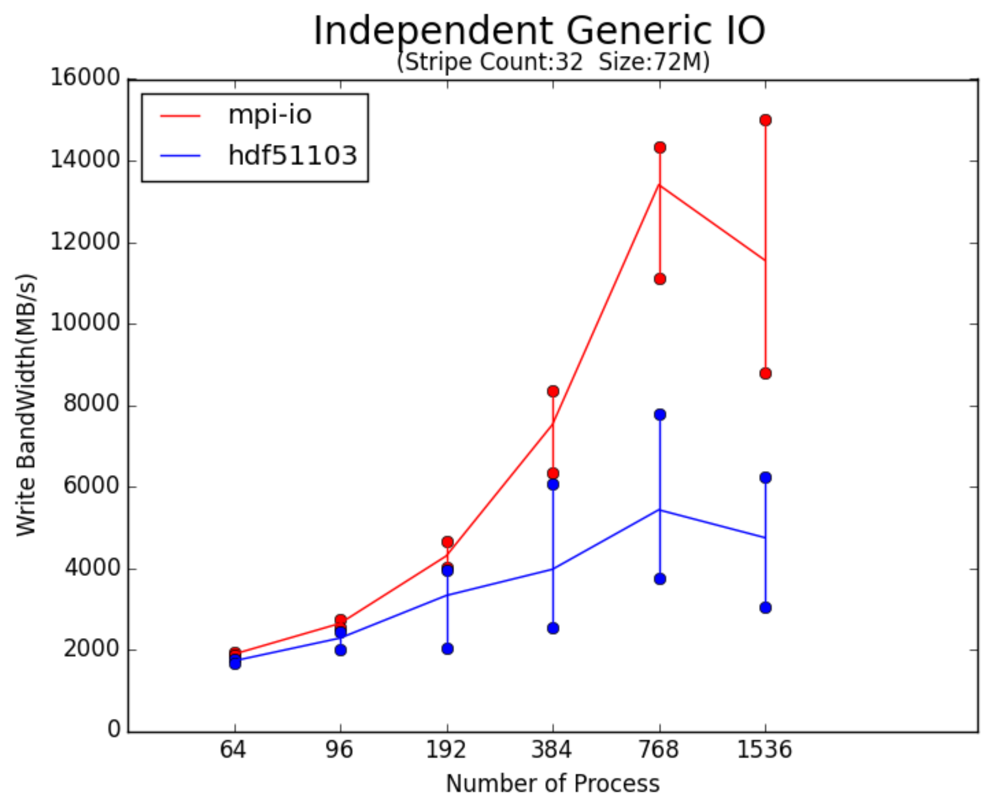


https://tinyurl.com/uoxkwaq

Case Study – Data Layout Effects

Benchmark:

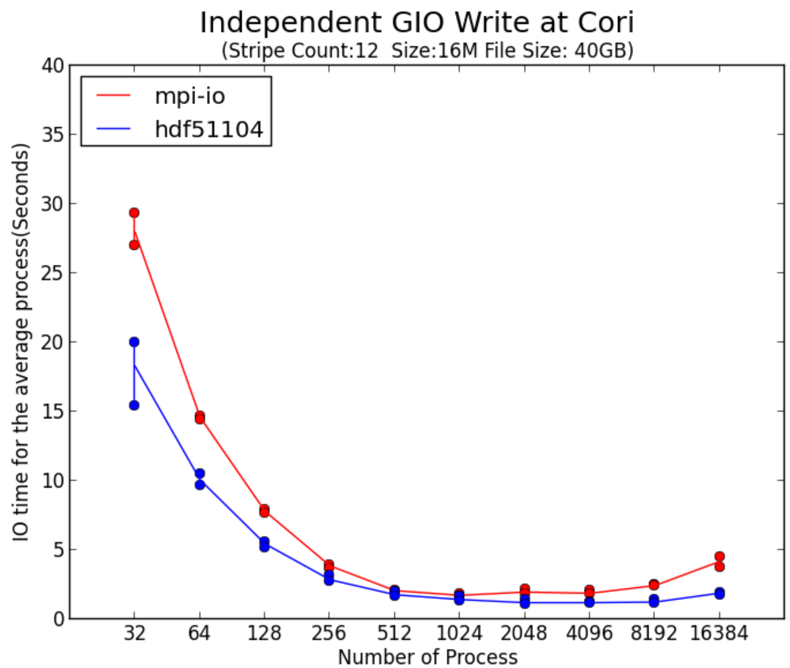
- 9 1-D variables with the same number of elements (~1e9).
- Total file size is about 40GB.
- Can switch between writing with MPI-IO or HDF5.
- Used independent IO for write.



https://tinyurl.com/uoxkwaq

HDF5 Pattern 2 Implementation

- Use HDF5 compound datatype, then one big HDF5 write for each process



- An optional multi-dataset, access pattern specifier is in development.

https://tinyurl.com/uoxkwaq

CGNS

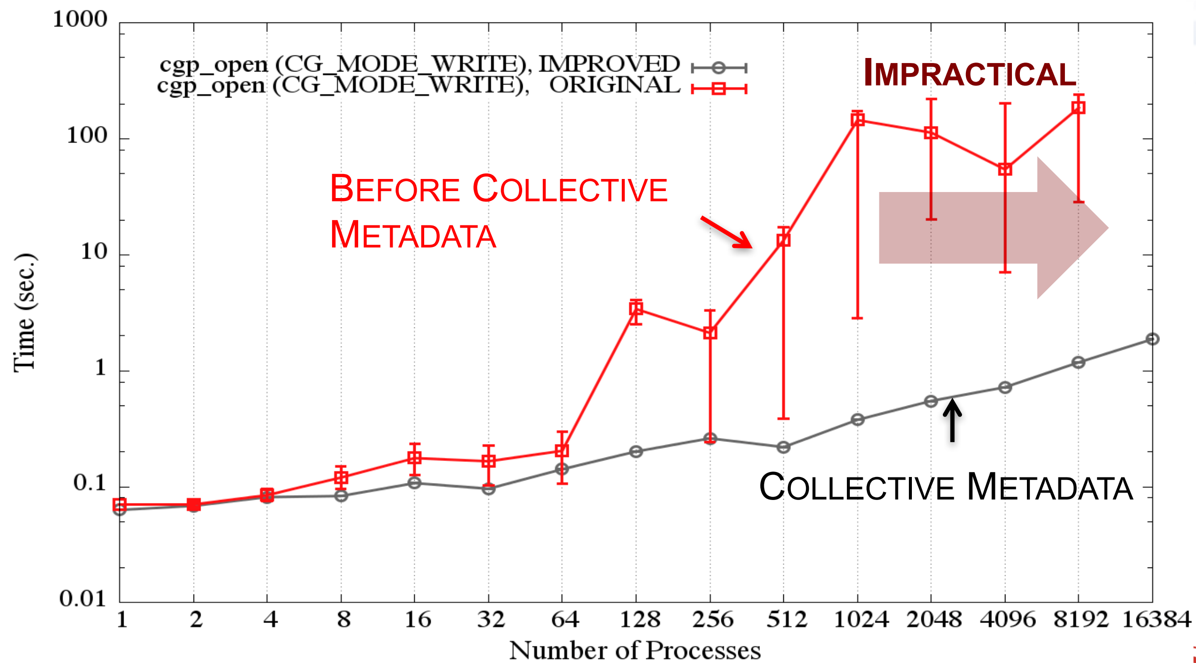
- CGNS = Computational Fluid Dynamics (CFD) General Notation System
- An effort to standardize CFD input and output data including:
  - Grid (both structured and unstructured), flow solution
  - Connectivity, boundary conditions, auxiliary information.
- Two parts:
  - A standard format for recording the data
  - Software that reads, writes, and modifies data in that format.
- An American Institute of Aeronautics and Astronautics Recommended Practice

https://tinyurl.com/uoxkwaq

Performance issue: Slow opening of an HDF5 File

- Opening an existing file
  - CGNS reads the entire HDF5 file structure, loading a lot of (HDF5) metadata
  - Reads occur independently on ALL ranks competing for the same metadata

➔"Read Storm"



https://tinyurl.com/uoxkwaq

Metadata Read Storm Problem (I)

- All metadata “write” operations are required to be collective:

```
if(0 == rank)
    H5Dcreate(“dataset1”);
else if(1 == rank)
    H5Dcreate(“dataset2”);
```

✗

```
/* All ranks have to call */
H5Dcreate(“dataset1”);
H5Dcreate(“dataset2”);
```

✓

- Metadata read operations are not required to be collective:

```
if(0 == rank)
    H5Dopen(“dataset1”);
else if(1 == rank)
    H5Dopen(“dataset2”);
```

✓

```
/* All ranks have to call */
H5Dopen(“dataset1”);
H5Dopen(“dataset2”);
```

✓

https://tinyurl.com/uoxkwaq

Metadata Read Storm Problem (II)

- Metadata read operations are treated by the library as independent read operations.
- Consider a very large MPI job size where all processes want to open a dataset that already exists in the file.
- All processes
  - Call `H5Dopen("/G1/G2/D1");`
  - Read the same metadata to get to the dataset and the metadata of the dataset itself
    - IF metadata not in cache, THEN read it from disk.
  - Might issue read requests to the file system for the same small metadata.

➔ READ STORM

https://tinyurl.com/uoxkwaq

Avoiding a Read Storm

- Hint that metadata access is done collectively
  - `H5Pset_coll_metadata_write, H5Pset_all_coll_metadata_ops`
- A property on an access property list
- If set on the file access property list, then all metadata read operations will be required to be collective
- Can be set on individual object property list
- If set, MPI rank 0 will issue the read for a metadata entry to the file system and broadcast to all other ranks

https://tinyurl.com/uoxkwaq

Improve the performance of reading/writing H5S_all selected datasets

(1) New in HDF5 1.10.5

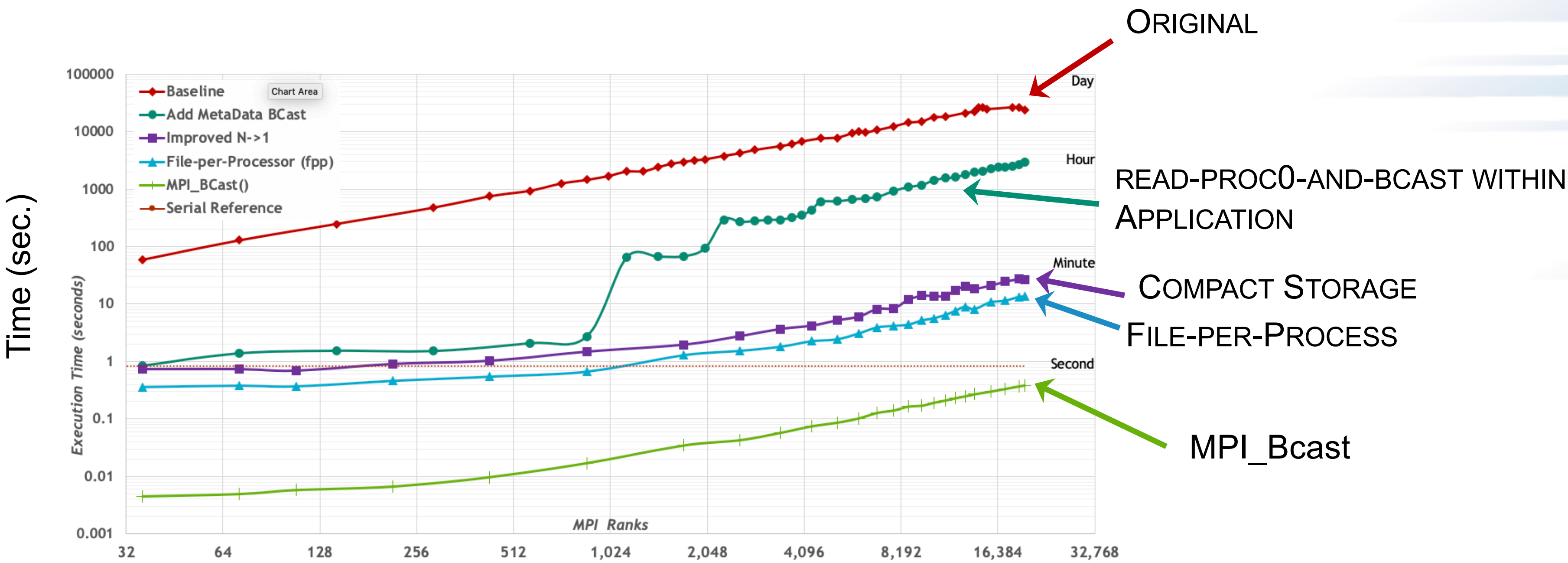
- If:
  - All the processes are reading/writing the same data
  - And the dataset is less than 2GB
- Then
  - The lowest process id in the communicator will read and broadcast the data or will write the data.

(2) Use of compact storage, or

- For compact storage, this same algorithm gets used.

https://tinyurl.com/uoxkwaq

SCALING OPTIMIZATIONS



Greg Sjaardema, Sandia National Labs

Don’t Forget: It’s a Multi-layer Problem

Application
(Semantic organization, standards compliance …)

HDF5
(cache chunk size, independent/collective …)

MPI-IO
(Number of collective buffer nodes, collective buffer size, …)

Parallel File System
(Lustre – stripe factor and stripe size)

Storage Hardware

https://tinyurl.com/uoxkwaq

Tools

DIAGNOSTICS AND INSTRUMENTATION

“Poor Man’s Debugging”

- Build a version of PHDF5 with

```
./configure --enable-debug --enable-parallel …
```

- This allows the tracing of MPIO I/O calls in the HDF5 library such as `MPI_File_read_xx` and `MPI_File_write_xx`
- Don’t forget to `setenv H5FD_mpio_Debug “rw”`

- You’ll get something like this…

https://tinyurl.com/uoxkwaq

Chunked by Column

```
% setenv H5FD_mpio_Debug 'rw'
% mpirun -np 4 ./a.out 1000       # Indep., Chunked by column.
in H5FD_mpio_write  mpi_off=0                size_i=96
in H5FD_mpio_write  mpi_off=0                size_i=96
in H5FD_mpio_write  mpi_off=0                size_i=96
in H5FD_mpio_write  mpi_off=0                size_i=96
in H5FD_mpio_write  mpi_off=3688             size_i=8000
in H5FD_mpio_write  mpi_off=11688            size_i=8000
in H5FD_mpio_write  mpi_off=27688            size_i=8000
in H5FD_mpio_write  mpi_off=19688            size_i=8000
in H5FD_mpio_write  mpi_off=96               size_i=40
in H5FD_mpio_write  mpi_off=136              size_i=544
in H5FD_mpio_write  mpi_off=680              size_i=120
in H5FD_mpio_write  mpi_off=800              size_i=272
…
```

Metadata

Dataset elements

Metadata

•

https://tinyurl.com/uoxkwaq

I/O monitoring and profiling tools

- Two kinds of tools:
  - I/O benchmarks for measuring a system's I/O capabilities
  - I/O profilers for characterizing applications' I/O behavior
- Two examples:
  - h5perf (in the HDF5 source code distro)
  - Darshan (from Argonne National Laboratory)
- Profilers have to compromise between
  - A lot of detail ➔ large trace files and overhead
  - Aggregation ➔ loss of detail, but low overhead

https://tinyurl.com/uoxkwaq

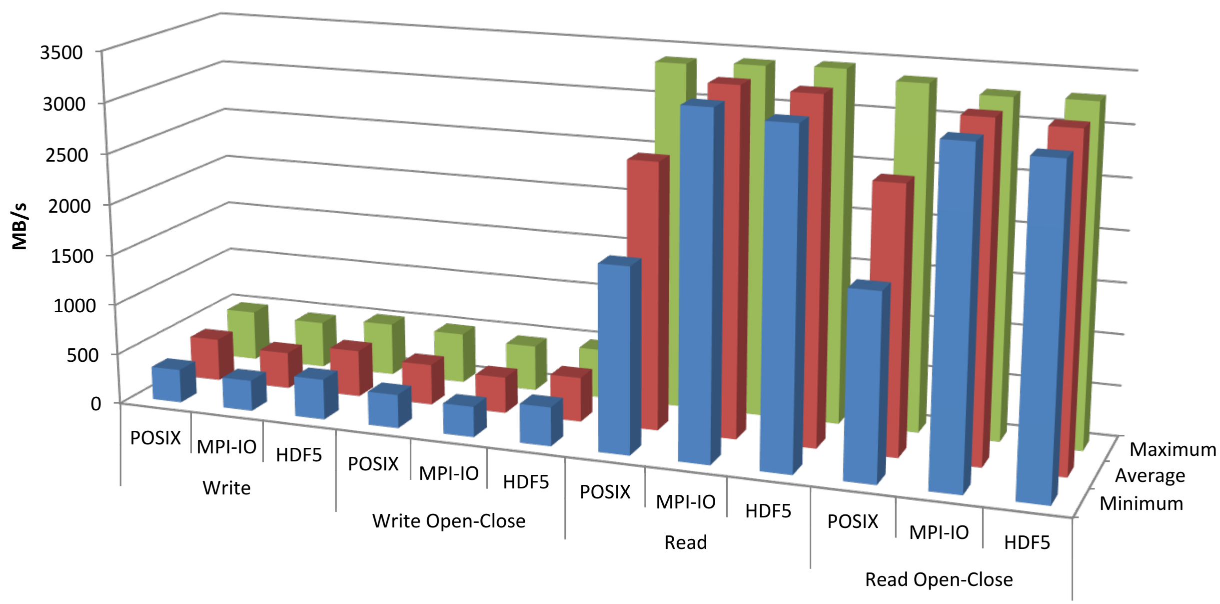
h5perf(_serial)

- Measures performance of a filesystem for different I/O patterns and APIs
- Three File I/O APIs for the price of one!
  – POSIX I/O (open/write/read/close…)
  – MPI-I/O (MPI_File_{open,write,read,close})
  – HDF5 (H5Fopen/H5Dwrite/H5Dread/H5Fclose)
- An indication of I/O speed ranges and HDF5 overheads
- Expectation management…

https://tinyurl.com/uoxkwaq

A Parallel Run

h5perf, 3 MPI processes, 3 iterations, 3 GB dataset (total), 1 GB per process, 1 GB transfer buffer, HDF5 dataset contiguous storage, HDF5 SVN trunk, NCSA BW



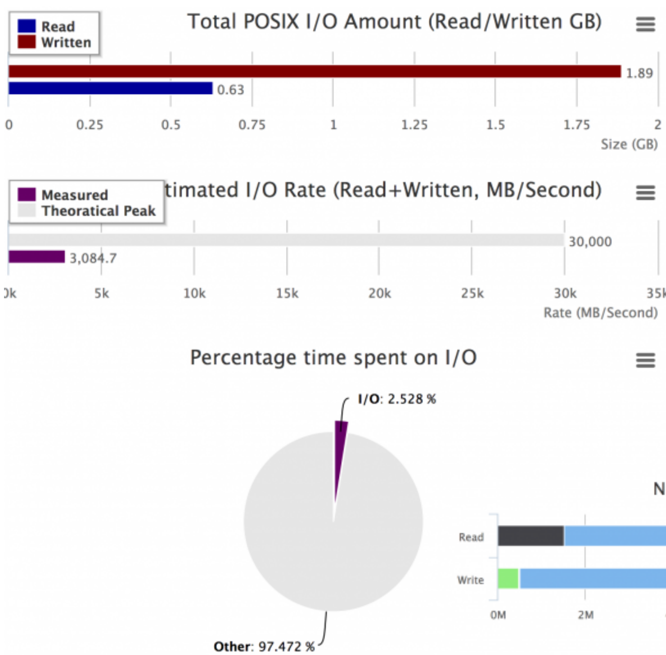
https://tinyurl.com/uoxkwaq

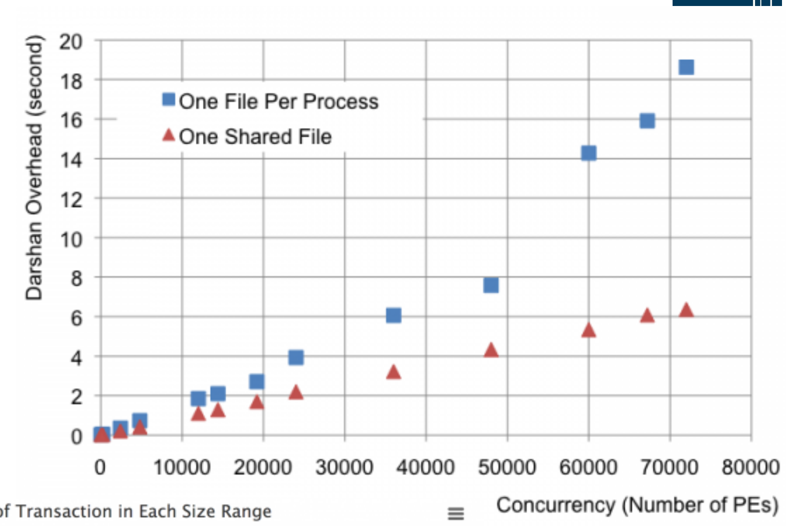
Darshan (ECP DataLib team)

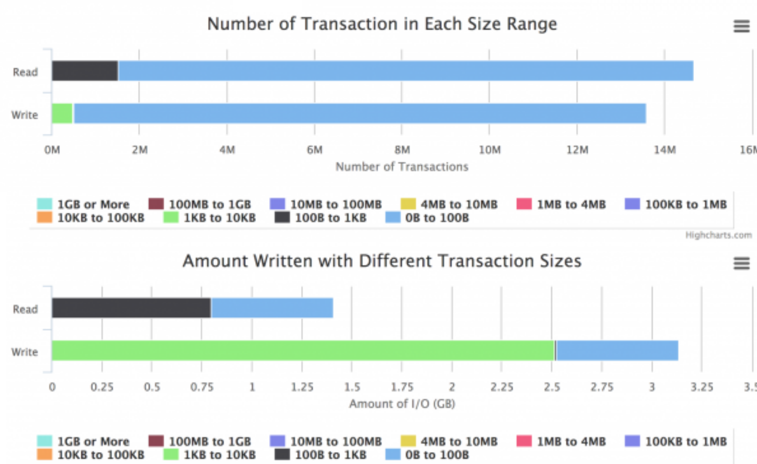
- Design goals:
  - Transparent integration with user environment
  - Negligible impact on application performance
- Provides aggregate figures for:
  - Operation counts (POSIX, MPI-IO, HDF5, PnetCDF)
  - Datatypes and hint usage
  - Access patterns: alignments, sequentiality, access size
  - Cumulative I/O time, intervals of I/O activity
- Does not provide I/O behavior over time
- Excellent starting point, maybe not your final stop

https://tinyurl.com/uoxkwaq

Darshan Sample Output



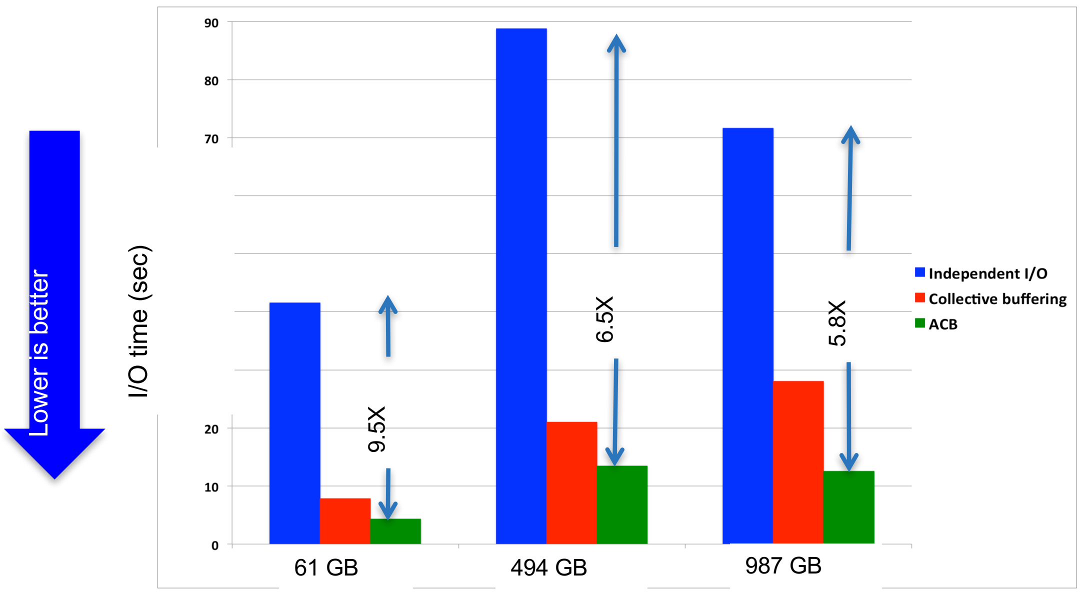




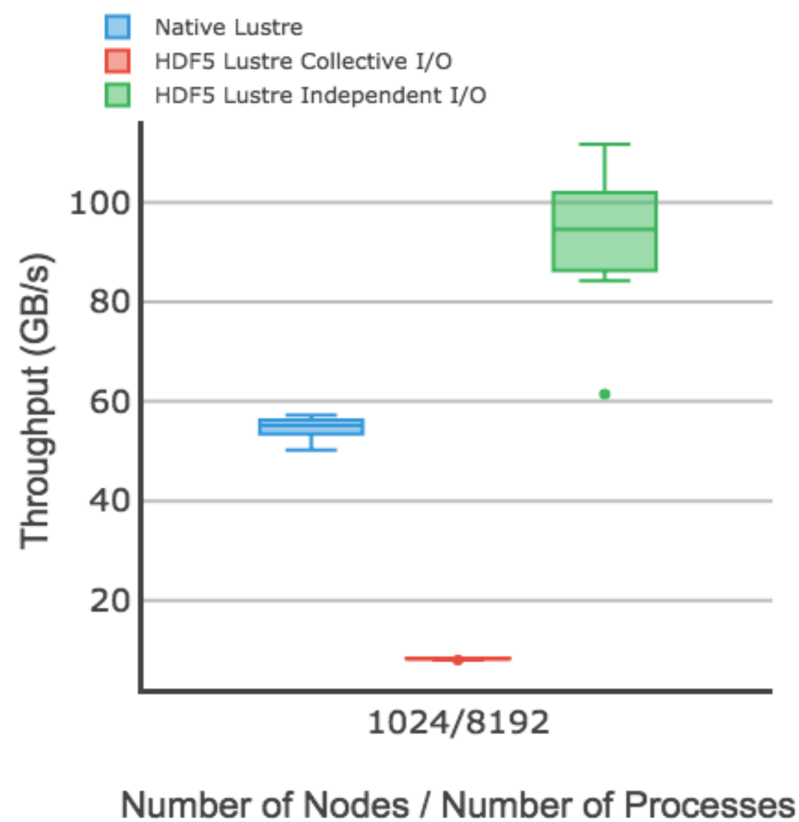
Source: NERSC

https://tinyurl.com/uoxkwaq

Chombo I/O (collective vs. independent)



AMRex I/O (collective vs. independent)



https://tinyurl.com/uoxkwaq

EXASCALE COMPUTING PROJECT

ECP EXAIO - HDF5 PROJECT

NEW FEATURES AND APPLICATION SUPPORT

Suren Byna

ECP HDF5 project mission

- Work with ECP applications and facilities to meet their needs
- Productize HDF5 features
- Support, maintain, package, and release HDF5
- R&D toward future architectures and incoming requests from ECP teams

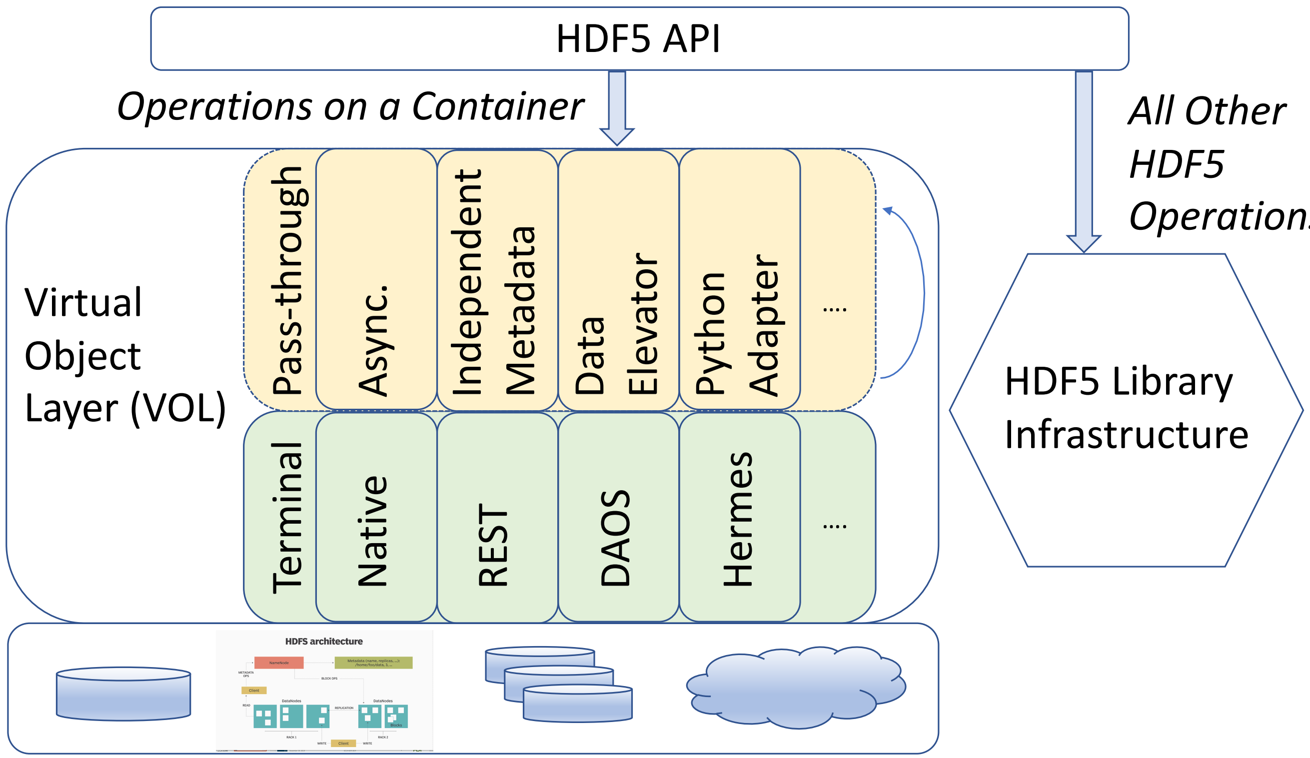
https://tinyurl.com/uoxkwaq

Features: Virtual Object Layer (VOL)

- Abstraction Layer within HDF5 Library
  - Redirect I/O operations into VOL “connector”, immediately after an API routine is invoked
- VOL Connectors
  - Implement “storage” for HDF5 objects, and “methods” on those objects
    - Dataset create, write / read selection, query metadata, close, …
  - Able to be transparently invoked from a dynamically loaded shared library, without modifying application source code (or even rebuilding the app binary)
  - Can be stacked, to allow many types of connector to be invoked
    - “Pass-through” and “Terminal” connector types

https://tinyurl.com/uoxkwaq

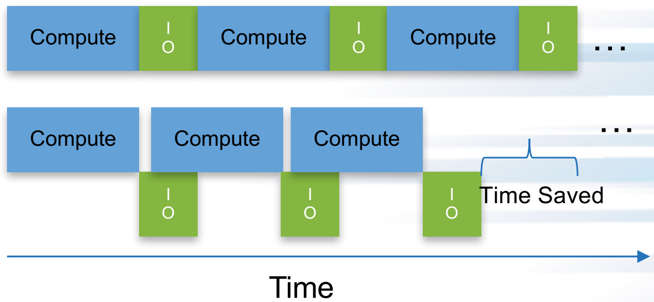
VOL overview and connectors



https://tinyurl.com/uoxkwaq

Features: Asynchronous I/O

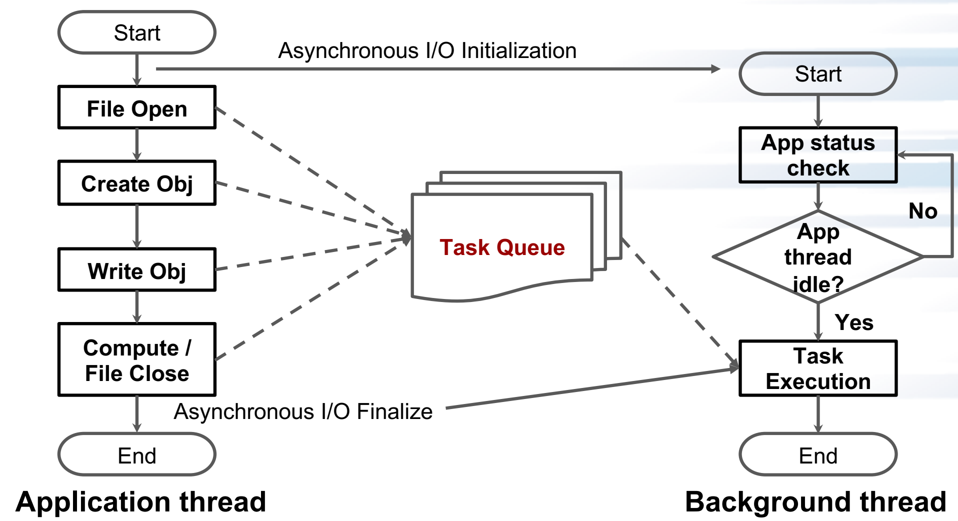
- Allows asynchronous operations for HDF5 applications:
  - Implicit: For unmodified applications; transparently invoked by setting environment variable; conservative asynchronous behavior
  - Explicit: For applications that want more control of async operations; can extract more performance using async operations that return “request tokens” to app



https://tinyurl.com/uoxkwaq

Asynchronous HDF5 Operations VOL Connector

- Implemented as a pass-through VOL connector w/background threads, using Argobots
- Transparent from the application, no major code changes
- Execute I/O operations in the background thread
- Lightweight and low overhead for all I/O operations
- No need to launch and maintain extra server processes



https://bitbucket.hdfgroup.org/projects/HDF5VOL/repos/async/

  - More details in PDSW Paper

    https://sc19.supercomputing.org/proceedings/workshops/workshop_files/ws_pdsw109s2-file1.pdf

    https://tinyurl.com/uoxkwaq

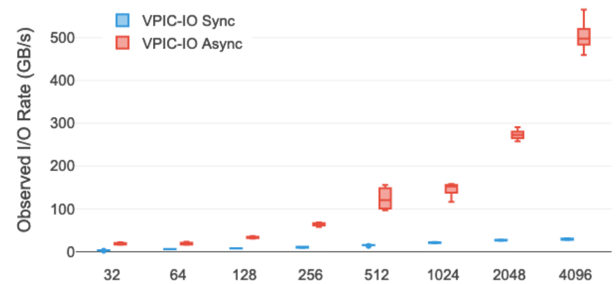
Asynchronous HDF5 Operations VOL Connector

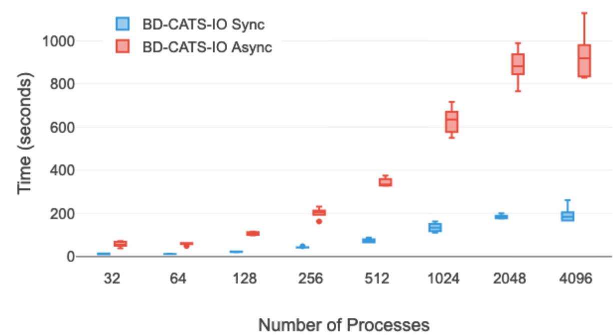
- Implemented as a pass-through VOL connector w/background threads, using Argobots
- Transparent from the application, no major code changes
- Execute I/O operations in the background thread
- Lightweight and low overhead for all I/O operations
- No need to launch and maintain extra server processes

https://bitbucket.hdfgroup.org/projects/HDF5VOL/repos/async/

- More details in PDSW Paper:
  - https://sc19.supercomputing.org/proceedings/workshops/workshop_files/ws_pdsw109s2-file1.pdf
    https://tinyurl.com/uoxkwaq

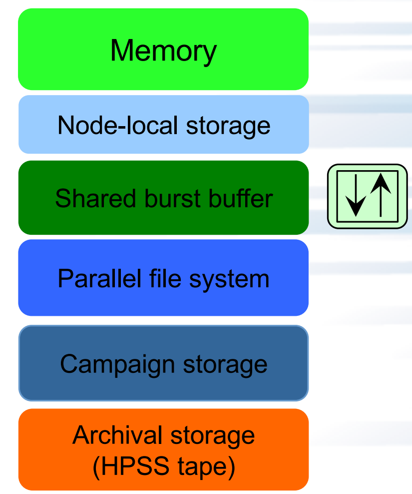
On Summit

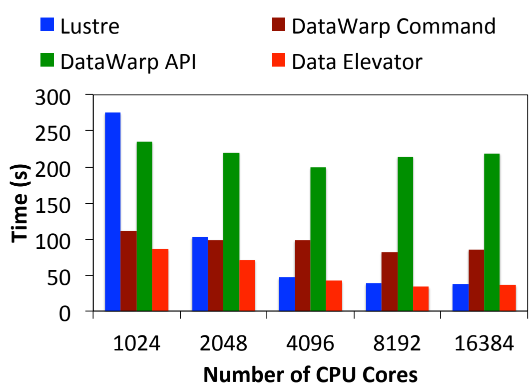


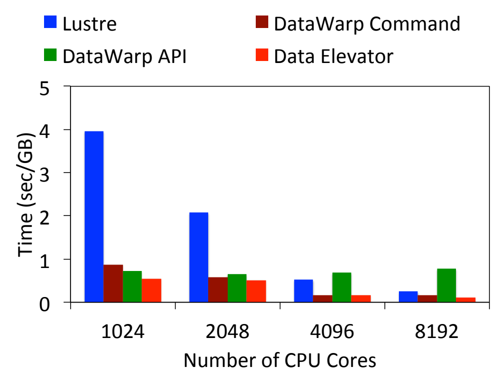


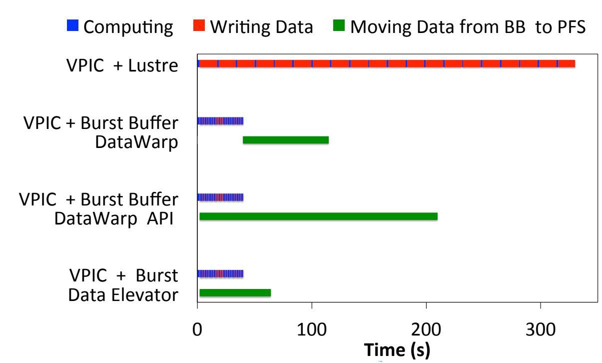
Features: Data Elevator for using shared burst buffers - Write

- Data Elevator write caching using burst buffers
  - Transparent data movement in storage hierarchy
  - In situ data analysis capability using burst buffers
- Tested with a PIC code and Chombo-IO benchmark
- Applications evaluating Data Elevator
  - E3SM-MMF and Sandia ATDM project is evaluating performance
  - Other candidates: EQSim, AMReX
- Installed on NERSC's Cori system (*module load data-elevator*)



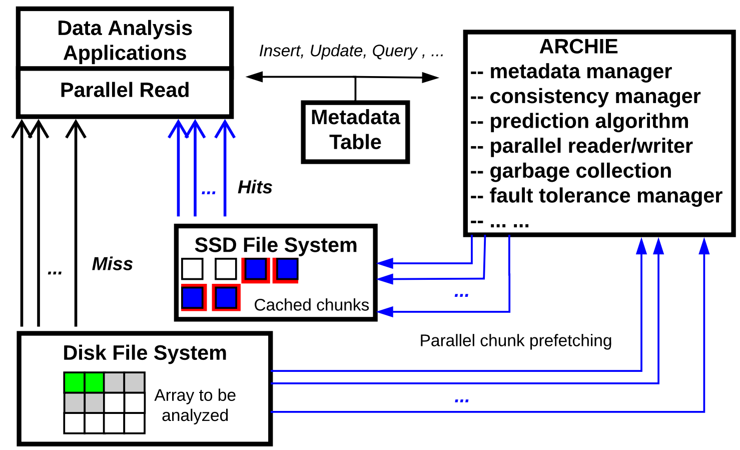


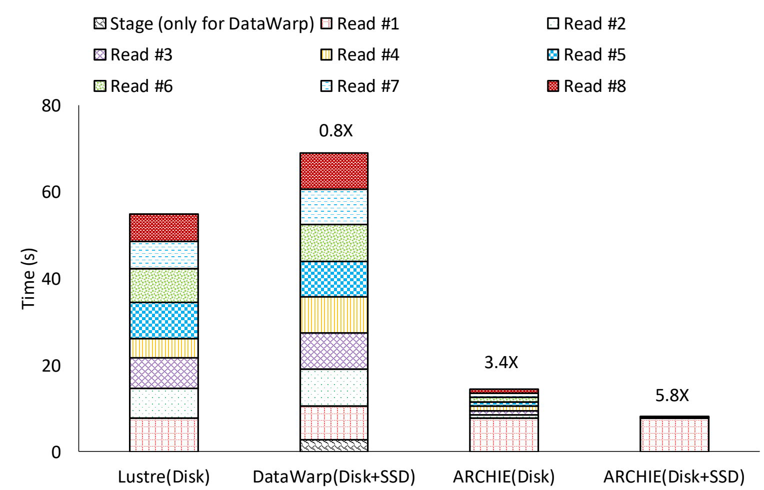




Features: Data Elevator for using shared burst buffers - Read

- ARCHIE - Array caching in hierarchical storage
- ARCHIE predicts data accesses based on a history of accesses, and prefetches them in faster storage layers for future use
- Automatic conversion of expensive non-contiguous accesses to storage devices into faster contiguous data accesses
- ARCHIE supports HDF5 I/O library and is part of the Data Elevator Virtual Object Layer (VOL) connector

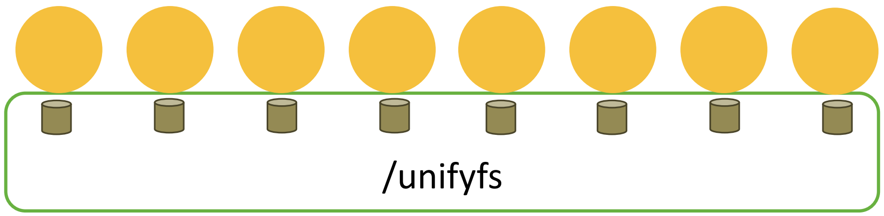




https://bitbucket.hdfgroup.org/projects/HDF5VOL/repos/dataelevator

https://tinyurl.com/uoxkwaq

UnifyFS for node-local storage (Project collaboration)

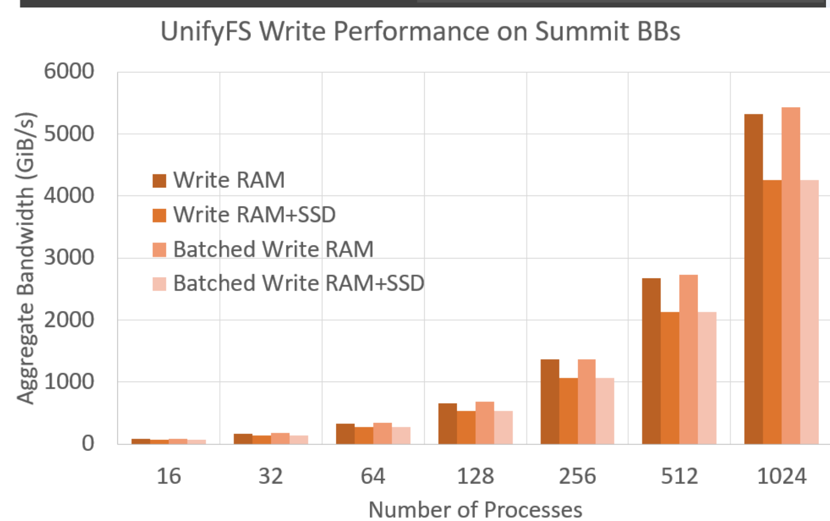


- A file system for node-local burst buffers
  – Developed by LLNL, ORNL, and NCSA team
- Goal: make using burst buffers on exascale systems *easy* and *fast*
- Results on Summit show scalable write performance for UnifyFS with shared files on burst buffers
- Designing Data Elevator to use UnifyFS as a single node-local burst buffer namespace for caching
- UnifyFS is designing an API for supporting HDF5, ADIOS, netCDF, etc.

```
int main(int argc, char **argv) {
 MPI_Init(argc, argv);

 for (t = 0; t < TIMESTEPS; t++) {

  /* do work ... */

  checkpoint();
 }

 MPI_Finalize();

 return 0;
}
```

```
void checkpoint(void) {
 int rank;

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 // file = "/pfs/shared.chpt";
 file = "/unifyfs/shared.ckpt";

 File *fs = fopen(file, "w");

 if (rank == 0)
  fwrite(header, ..., fs);

 long offset = header_size +
               rank*state_size;
 fseek(fs, offset, SEEK_SET);
 fwrite(state, ..., fs);
 fclose(fs);
}
```

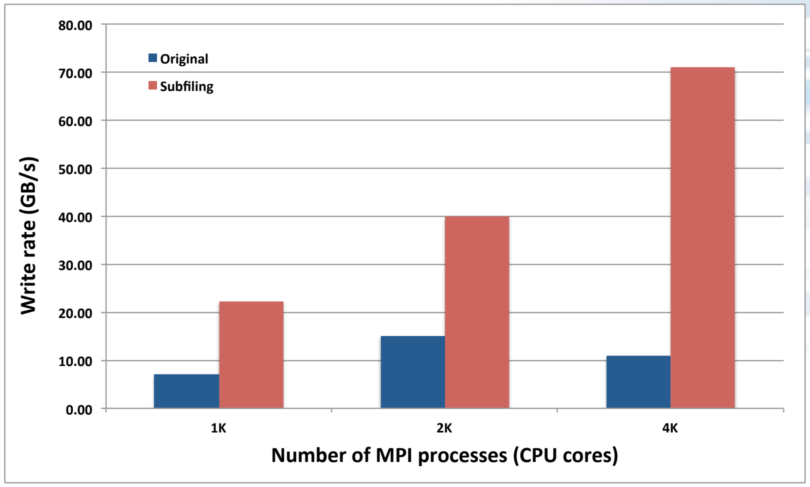
The only required change is to use `/unifyfs` instead of `/pfs`



https://tinyurl.com/uoxkwaq

Features: Sub-filing

- Writing to single shared file is slow due to:
  - Locking contention
- A solution: Sub-filing
  - Multiple small files
  - A metadata file stitching the small files together
- Benefits
  - Better use of parallel I/O subsystem
  - Reduced locking and contention issues improve performance
- Designing production quality implementation of sub-filing in HDF5 using Virtual File Driver (VFD)
  - Will use node-local storage for caching



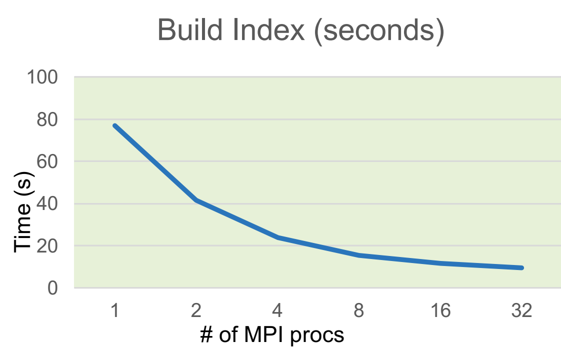
Performance on Cori with prototype implementation to show the potential of sub-filing

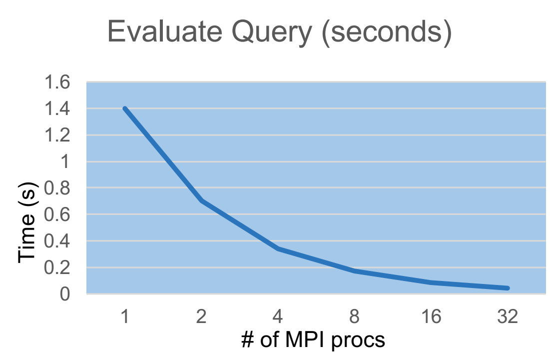
https://tinyurl.com/uoxkwaq

Features: Querying datasets

- HDF5 *index* objects and API routines allow the creation of indexes on the contents of HDF5 containers, to improve query performance
- HDF5 *query* objects and API routines enable the construction of query requests for execution on HDF5 containers
  - H5Qcreate
  - H5Qcombine
  - H5Qapply
  - H5Qclose

- HDF5 Bitbucket repo containing the "topic-parallel-indexing" source code: https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5



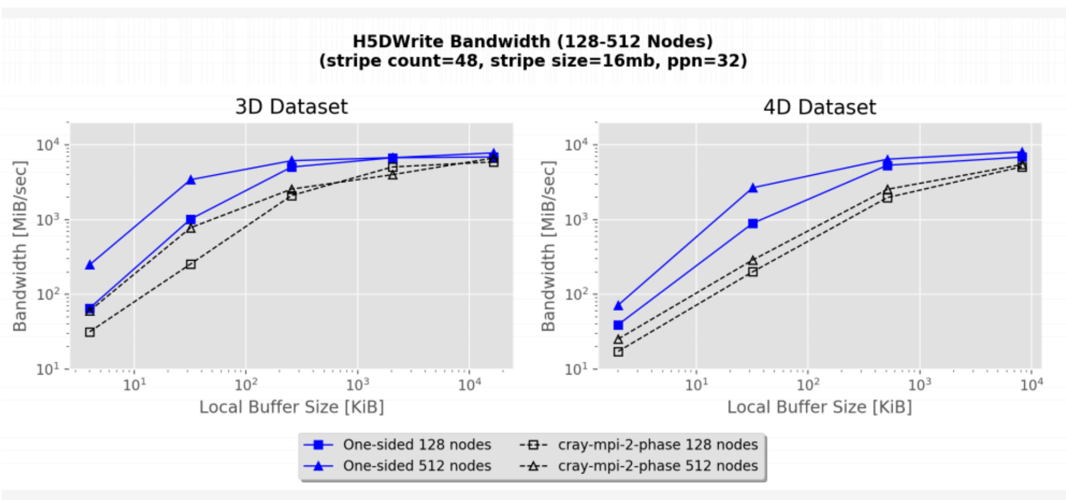


- Parallel scaling of index generation and query resolution is evidenced even for small-scale experiments:

https://tinyurl.com/uoxkwaq

Features: System topology-aware VFD

- Taking advantage of the topology of compute and I/O nodes and network among them improves overall I/O performance
- Developing topology-aware data-movement algorithms and collective I/O optimizations within a new HDF5 virtual file driver (VFD)



Performance comparison of the new HDF5 VFD, using one-sided aggregation, with the default binding to Cray MPICH MPI-IO. Data was collected on Theta using an I/O benchmarking tool (the HDF5 Exerciser),

Prototype implementation: **CCIO** branch
https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5/

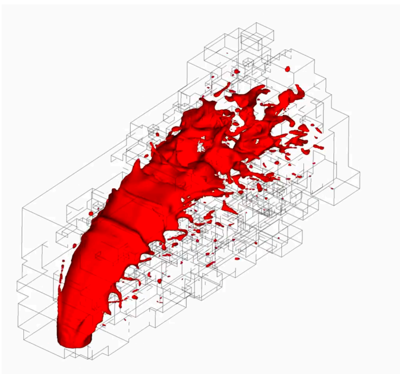
Features: Independent metadata updates

- Allows HDF5 metadata operations to be performed independently
  - Currently, all HDF5 metadata modification operations must be collective
    - Dataset / group creation & deletion, attribute create, write, etc.
  - Each metadata modification is “voted on” by other HDF5 processes writing to that file using non-blocking communication channels, then committed to the file
- IMM is a pass-through VOL connector
  - Allows IMM operations for *any* underlying HDF5 VOL connector
- Connector is extendible to multiple comm. channels: MPI, ZeroMQ, POSIX, etc.
- Async and IMM connectors demonstrate the power of pass-through VOL connectors to modify behavior of HDF5 library

https://tinyurl.com/uoxkwaq

Applications: AMReX-based applications

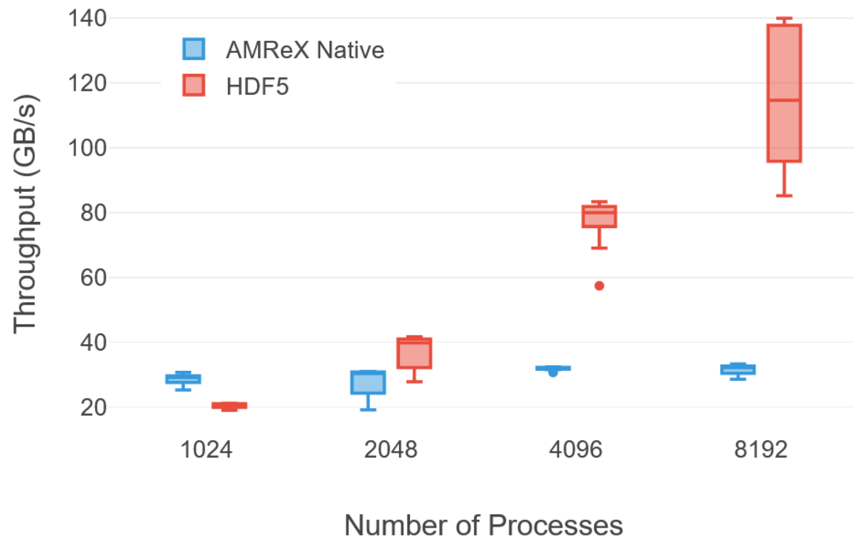
- AMReX - SW framework for building massively parallel block- structured adaptive mesh refinement (AMR) applications
  - Combustion, accelerator physics, carbon capture, cosmology apps from ECP use this framework
- HDF5: Integrated HDF5-based I/O functions for reading and writing plot files and particle data



Liquid jet in supersonic flow

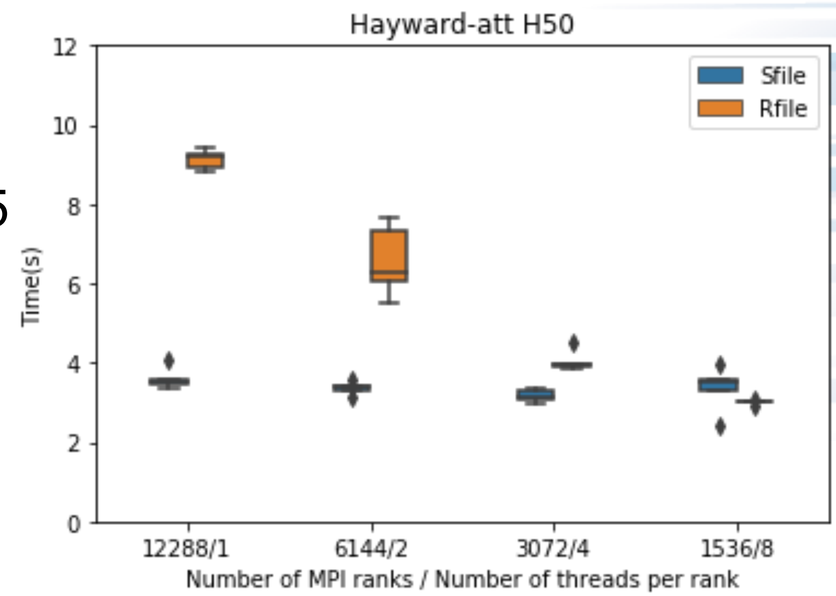
https://tinyurl.com/uoxkwaq

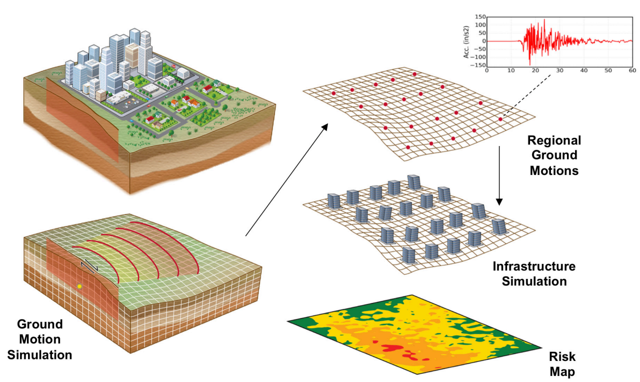
On Cori at NERSC



Applications: EQSIM

- EQSIM is a high performance, multidisciplinary simulation for regional-scale earthquake hazard and risk assessments
- Integrating various I/O functionality using HDF5 file format - for portability and for performance
  - Converted reading HDF5 formatted file
  - Implemented checkpointing data to HDF5
  - Implementing SW4 image output to HDF5

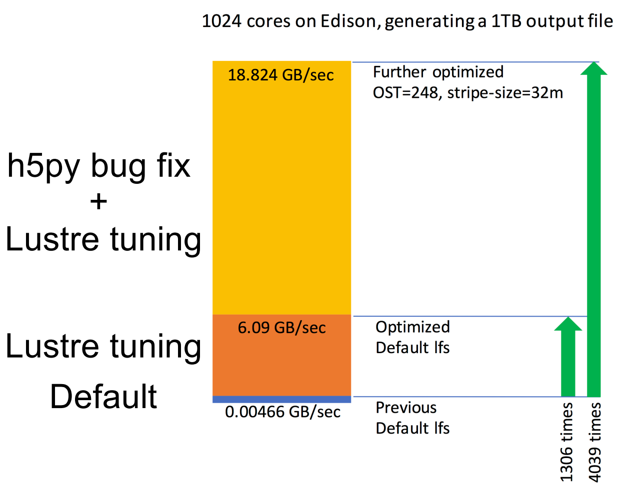




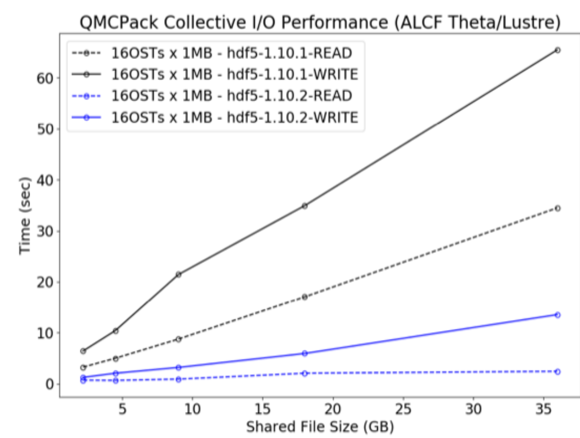
Applications: WarpX and QMCPACK

- WarpX is an advanced electromagnetic Particle-In-Cell code
- Applied file system and MPI-IO level optimizations to achieve good HDF5 I/O performance (uses h5py)

- QMCPACK, is a modern high-performance open-source Quantum Monte Carlo (QMC) simulation code
- HDF5 optimizations in file close and fixing a bug improved I/O performance



Warp-IO

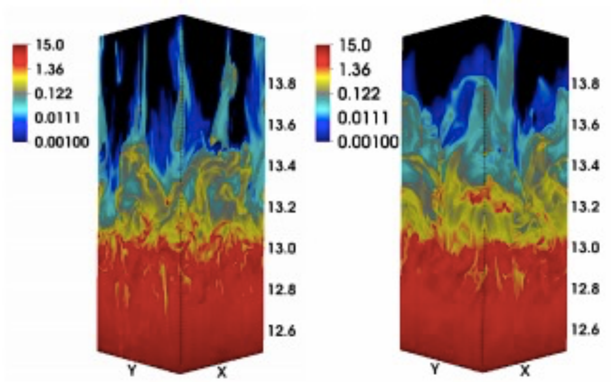


QMCPACK

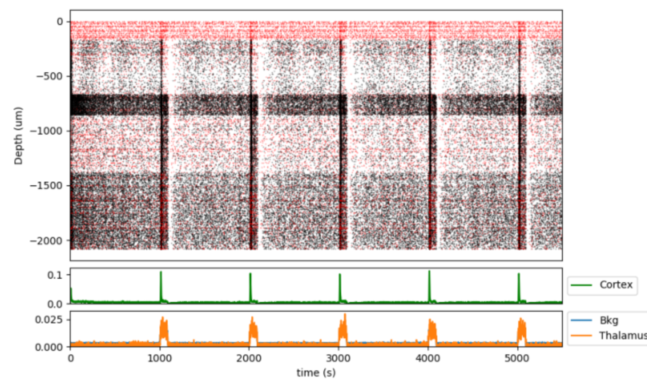
https://tinyurl.com/uoxkwaq

Facilities: Astrophysics and Neuroscience codes

- Supporting any I/O issue related tickets at facilities
- The following are astrophysics and neurological disorder pipelines that experienced high I/O overhead
- Used performance introspection interfaces of HDF5 to identify bottlenecks



Athena astrophysics code
40% of execution time in I/O, using HDF5 profiling tools identified a large number of concurrent writes; with collective I/O, reduced I/O portion to less than 1% of the execution time.

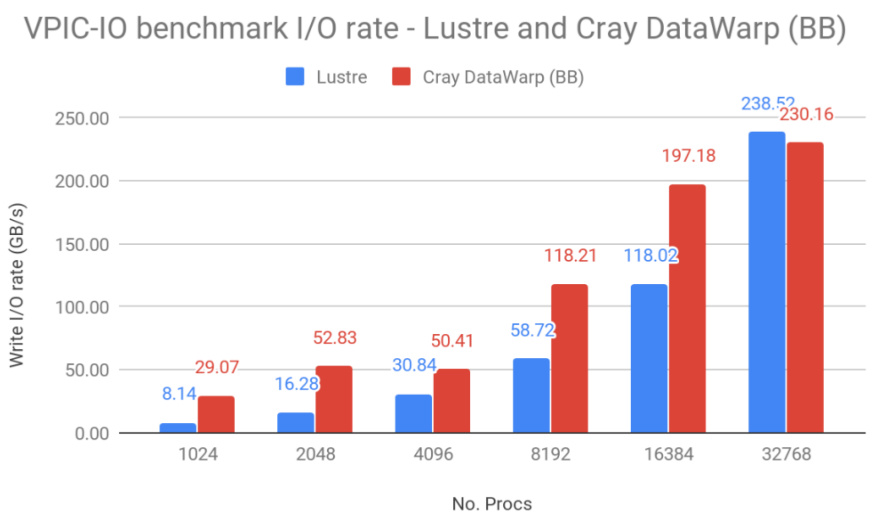


Neurological Disorder I/O Pipeline
Identified that h5py interface was prefilling HDF5 dataset buffers unnecessarily and avoiding that improved performance by 20X (from 40 min to 2 min)

https://tinyurl.com/uoxkwaq

Facilities: HDF5 benchmarking

- Benchmarking HDF5 on Cori, Theta, and Summit each quarter
- Benchmarks:
  - VPIC-IO: a simple I/O benchmark that writes particles from a PIC code
  - BD-CATS-IO: an I/O kernel from a clustering code



https://tinyurl.com/uoxkwaq

Need help?

- HDF-FORUM https://forum.hdfgroup.org/
- HDF Helpdesk help@hdfgroup.org
  - Indicate that you are with ECP project
- For ECP teams:

Contact the ExaIO POCs for existing collaborations and the PIs for new collaborations.

https://tinyurl.com/uoxkwaq

Thank you!

https://tinyurl.com/uoxkwaq