# Optimizing Your I/O Workload: Techniques for Effective HDF5 Usage

March 12, 2024



**M. Scot Breitenfeld**
**Gerd Heber**

# Talk Outline

The HDF Group

- Foundations of HDF5
  - Brief introduction to
    - HDF5 data model, software, and architecture
    - HDF5 programming model
  - Overview of general best practices

- Overview of parallel HDF5
  - Introduction to HDF5 parallel I/O
  - New features -- Subfiling, GPU-VFD
  - General best practices and methods that affect parallel performance

- Supplemental Tools
  - `tar2h5`
  - HSDS
  - H5Web

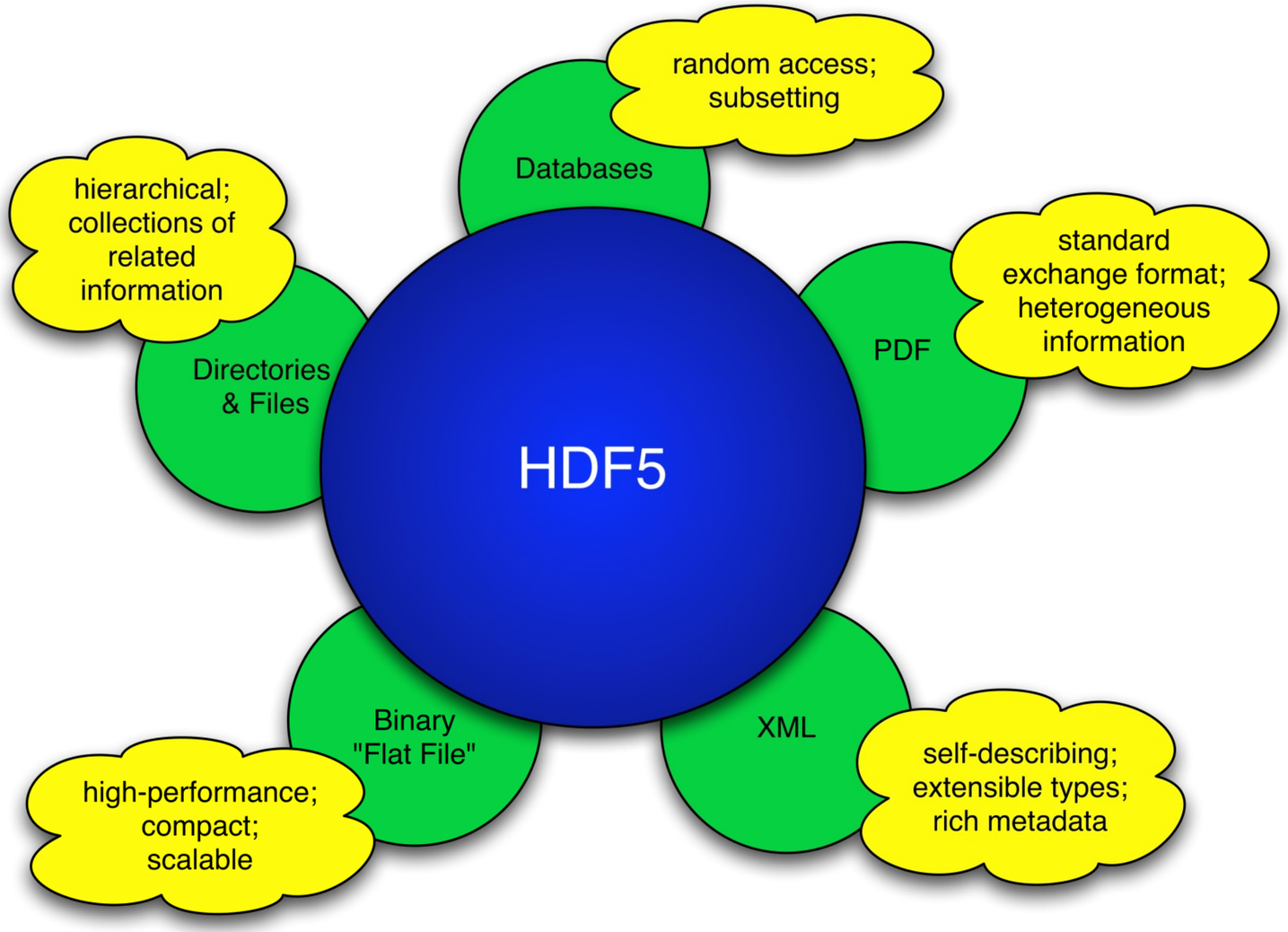
- Deep Learning I/O: Benchmark and Profiling - Hariharan Devarajan, Lawrence Livermore National Laboratory

# What is HDF5?

The HDF Group

- **H**ierarchical **D**ata **F**ormat version 5 (**HDF5**)
  1. An extensible **data model**
     - Uses structures for data organization and specification
  2. 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 backend storage: Object Stores (DAOS, AWS S3, AZURE, Ceph, etc.), memory hierarchies and other storage devices
  3. Open **file format** (POSIX storage only)

# HDF5 is like …


HDF5
Databases
random access; subsetting
Directories & Files
hierarchical; collections of related information
PDF
standard exchange format; heterogeneous information
Binary "Flat File"
high-performance; compact; scalable
XML
self-describing; extensible types; rich metadata

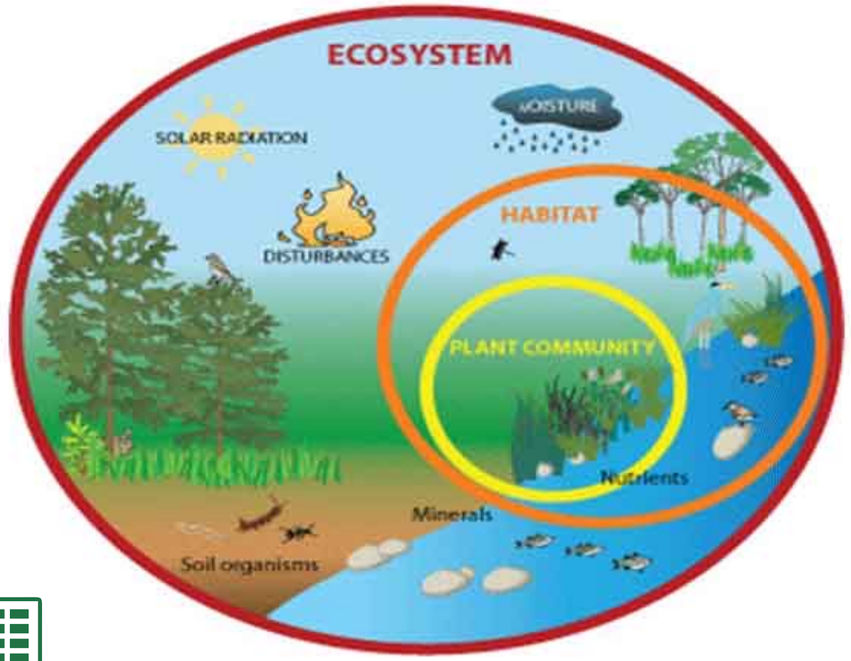
# HDF5 is designed for…

The HDF Group

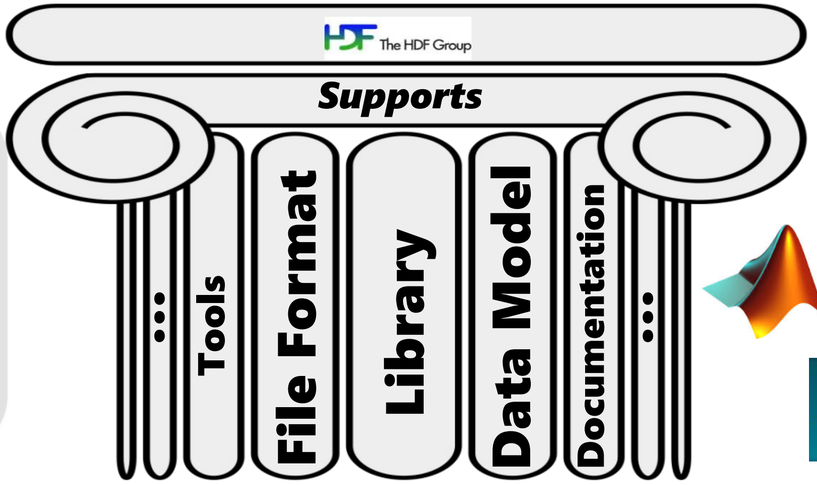
- High-volume and complex data
  - HDF5 files of GBs+ sizes are common
- Every size and type of system (portable)
  - Works on from embedded systems, desktops and laptops to exascale systems
- Flexible, efficient storage and I/O
  - Works for a variety of backing storage
- Enabling applications to evolve 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 in NASA archives now

# HDF5 Ecosystem

The HDF Group

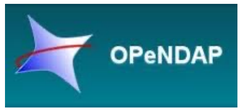








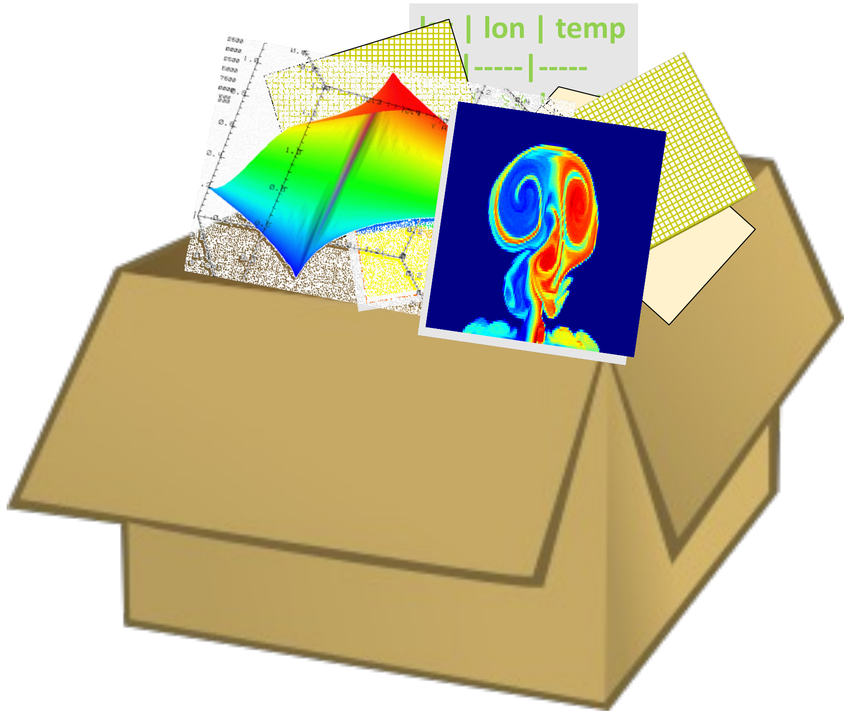




# HDF5 Data model

# HDF5 File

The HDF Group

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



# HDF5 Data Model

The HDF Group

**Dataset** –
Organize and contain data elements

**Dataspace** –
Describes logical layout of the data elements

**Attribute** –
*User-defined* metadata

HDF5 Objects

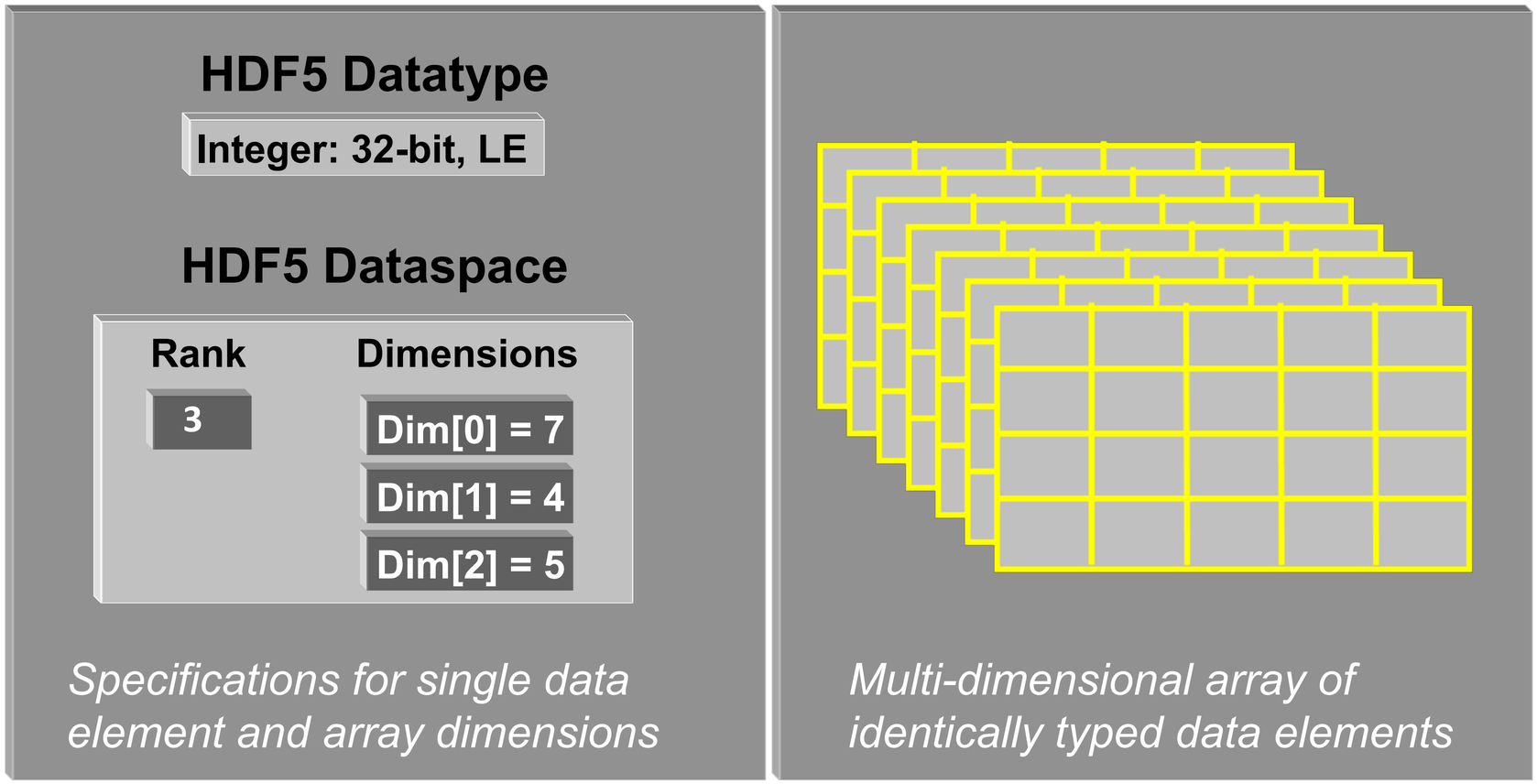
**File**

**Datatype** –
Describes individual data elements

**Link** –
Organize data objects

**Group** –
Organize data objects

# 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

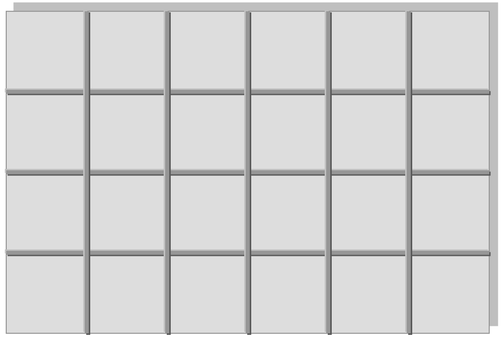
# HDF5 Dataspace

The HDF Group

Two roles:

(1) Spatial information for Datasets and Attributes

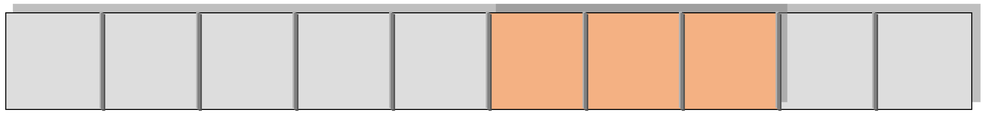
- Empty sets and scalar values
- Multidimensional arrays
  - Rank and dimensions
- A permanent part of object definition



Rank = 2

Dimensions = 4 x 6

(2) Partial I/O: Dataspace and subset describe the application's data buffer and data elements participating in I/O



Rank = 1

Dimension = 10

# How to describe a subset in HDF5?

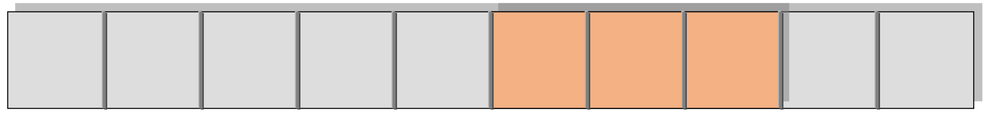
The HDF Group

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

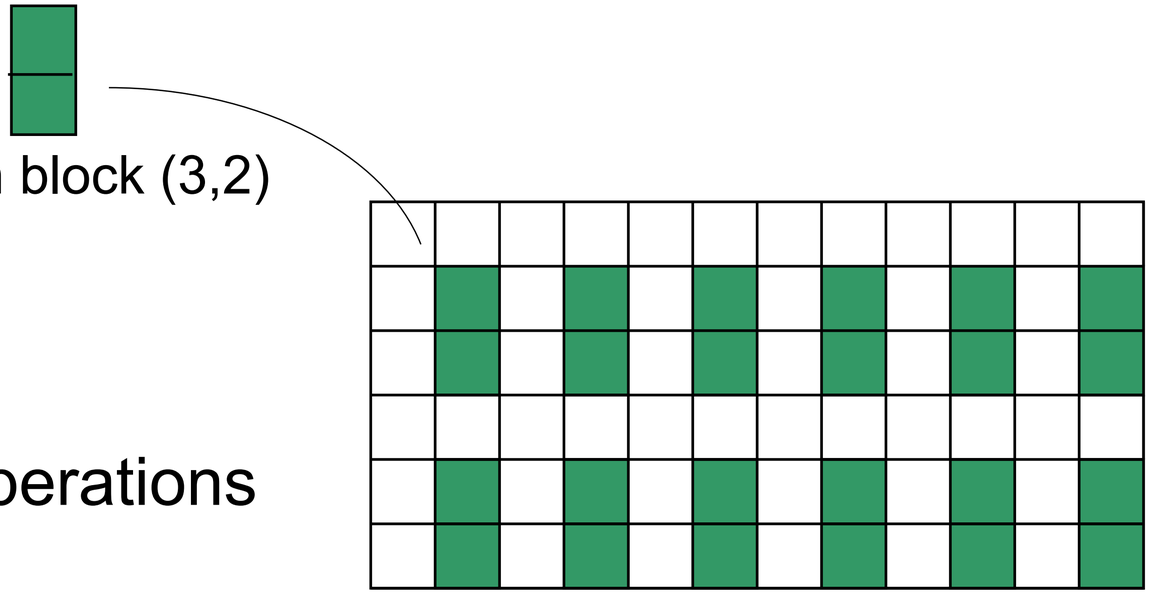
# Describing elements for I/O: HDF5 Hyperslab

The HDF Group

- *Everything is "measured" in the 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)
  - Block - block size (2,1)
  - Count - number of blocks (2,6)



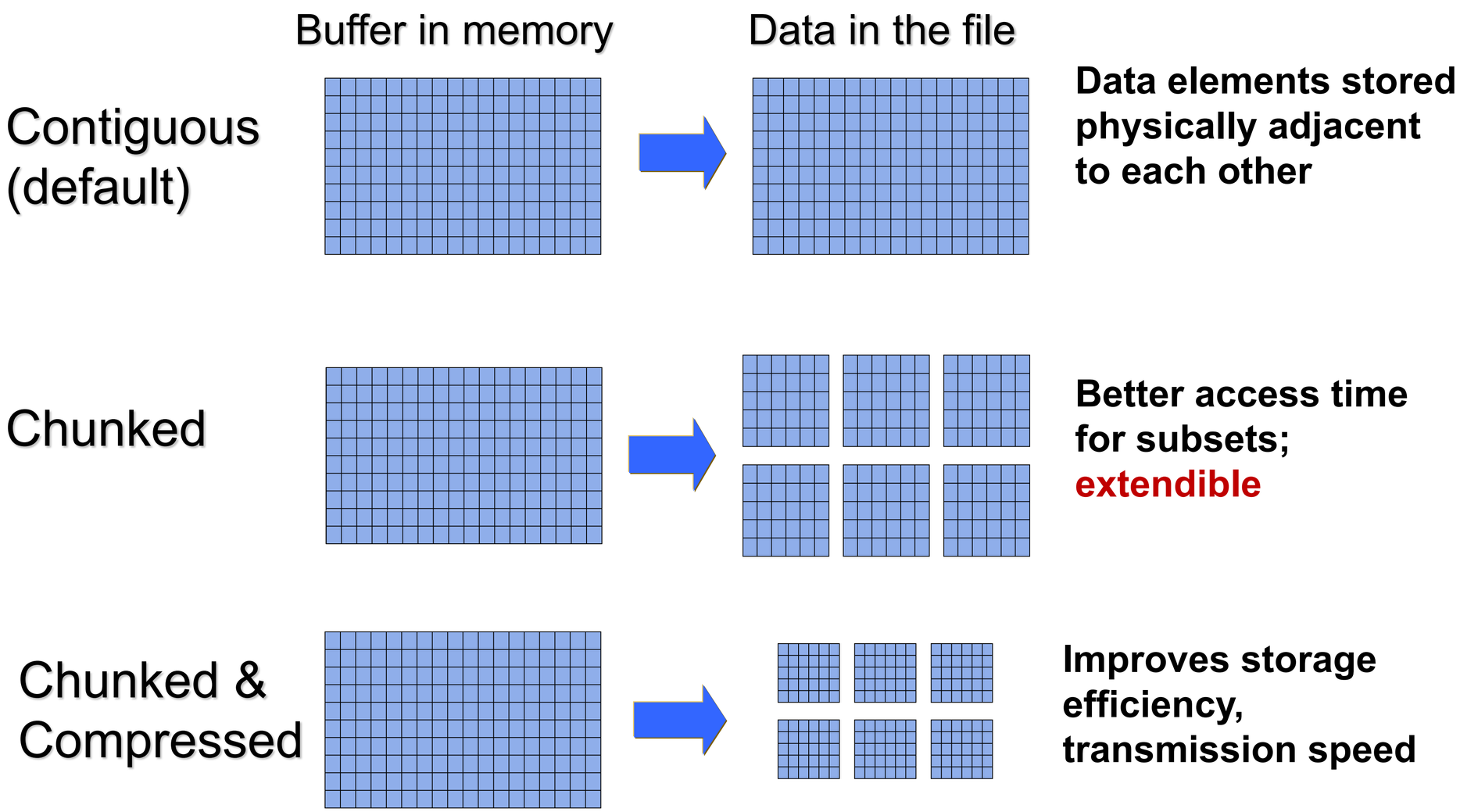
- All other selections are built using set operations

# HDF5 Datatypes

The HDF Group

- Describe individual data elements in an HDF5 dataset
- A 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's structures or Fortran's derived types)
  - Array (similar to matrix)
- HDF5 library provides predefined symbols to describe atomic datatypes

# How are data elements stored? (1/2)



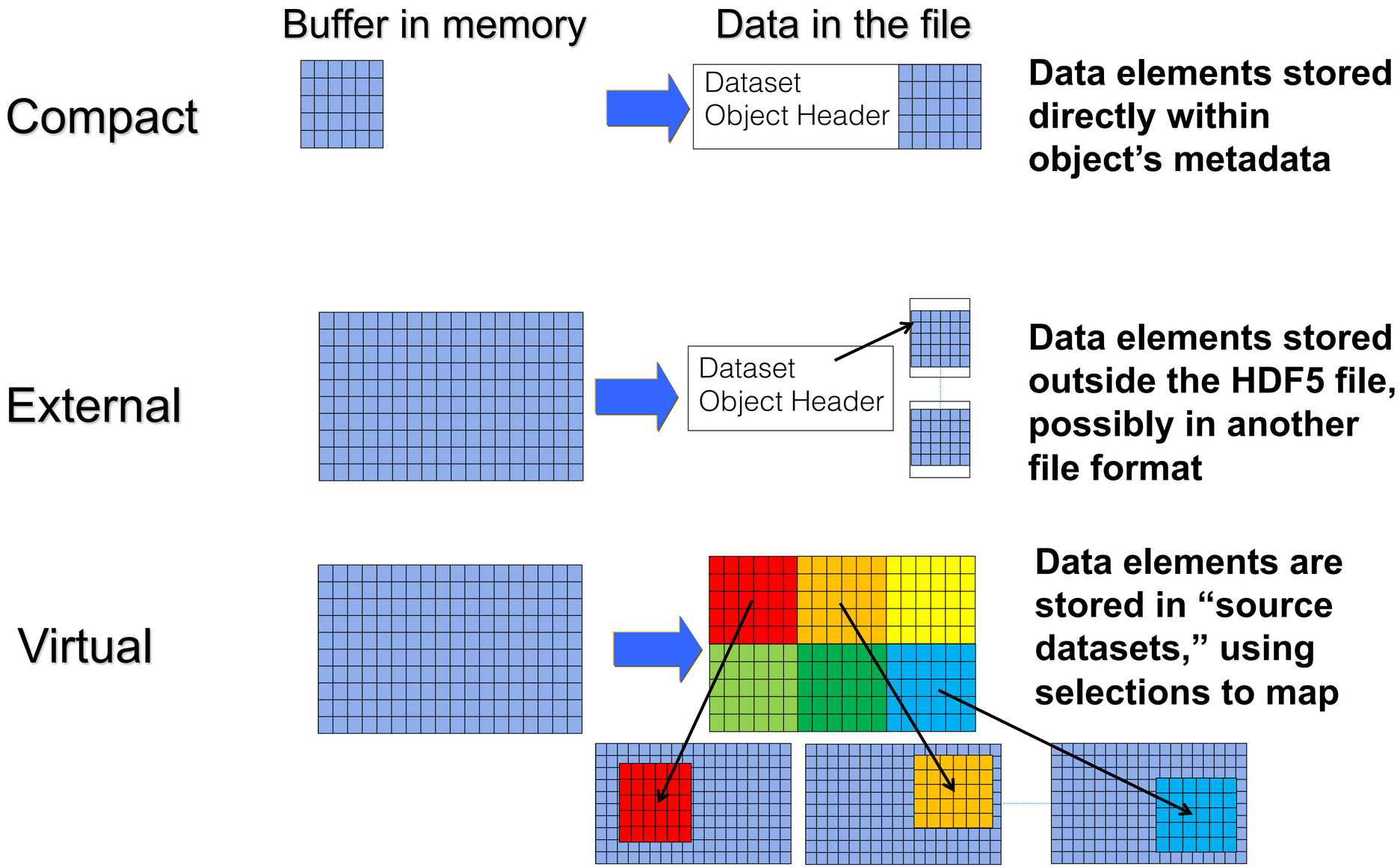
# Compression and filters in HDF5

The HDF Group

- 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/documentation/hdf5-docs/registered_filter_plugins.html
  - BZIP2, JPEG, LZF, BLOSC, MAFISC, LZ4, Bitshuffle, SZ and ZFP,  etc.
    - The listed above are available as dynamically loaded plugins

- Filters:
  - Fletcher32 (checksum)
  - Shuffle
  - Scale+offset
  - n-bit

# How are data elements stored? (2/2)



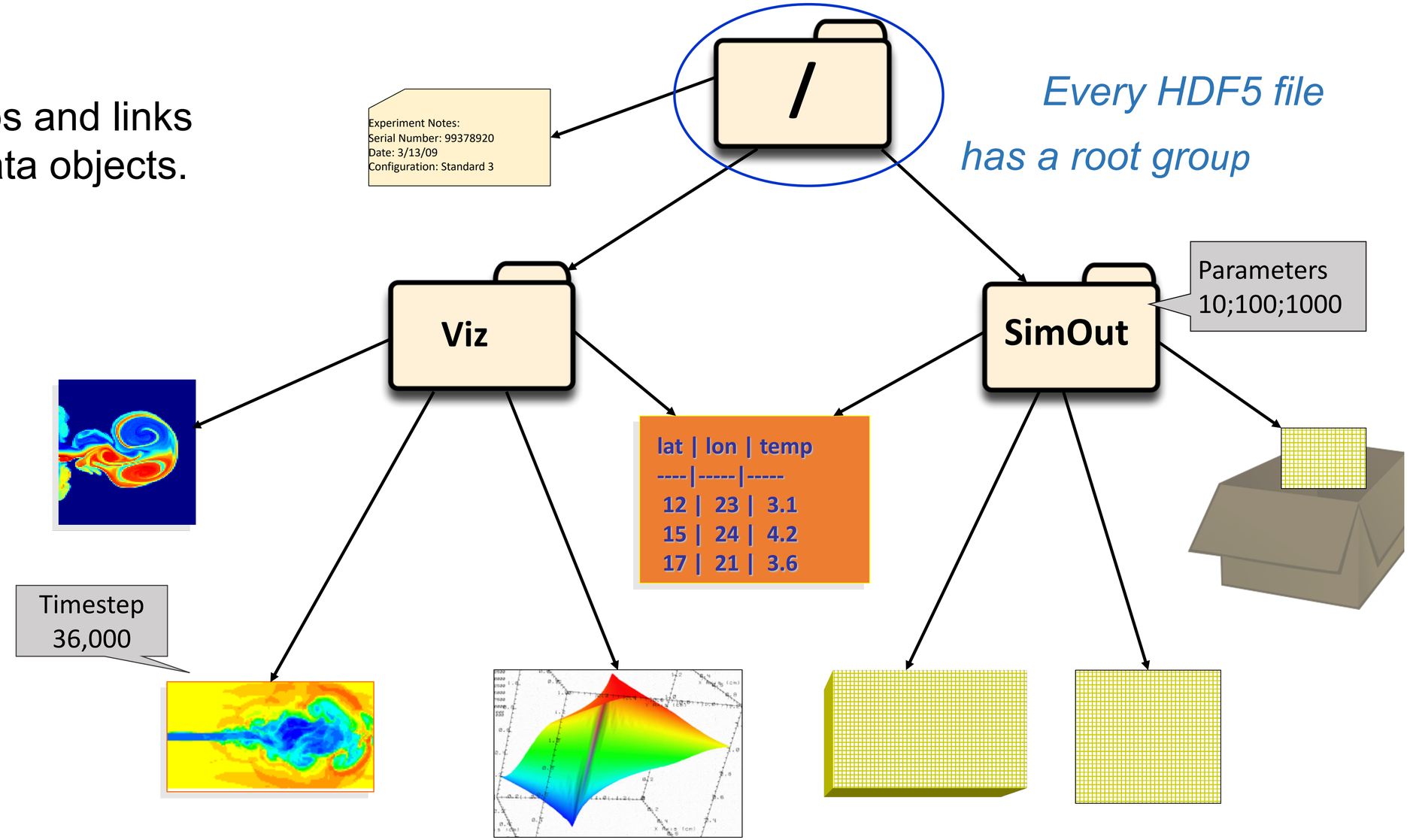
# HDF5 Attributes

The HDF Group

- 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

# HDF5 Groups and Links

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



# HDF5 software and architecture

# HDF5 Software

The HDF Group

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

- Latest releases: HDF5 1.14.3

HDF5 source code:

- Available on GitHub: https://github.com/HDFGroup/hdf5
- 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:

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

3rd party software:

- h5py (Python, pip install h5py)
- HDF5.jl (Julia)
- h5cpp[1], h5cpp[2] (Contemporary C++ including support for MPI I/O )

# Useful Tools For New Users

The HDF Group

**h5dump**

Tool to “dump” or display contents of HDF5 files

**Scripts to compile applications:**

h5cc, h5c++, h5fc (*h5pcc, h5pfc – parallel variants*)

**HDFView**:

Java browser to view HDF5 file

https://www.hdfgroup.org/downloads/hdfview/

**HDF5 Examples** (C, Fortran, Java, Python, Matlab, ...)

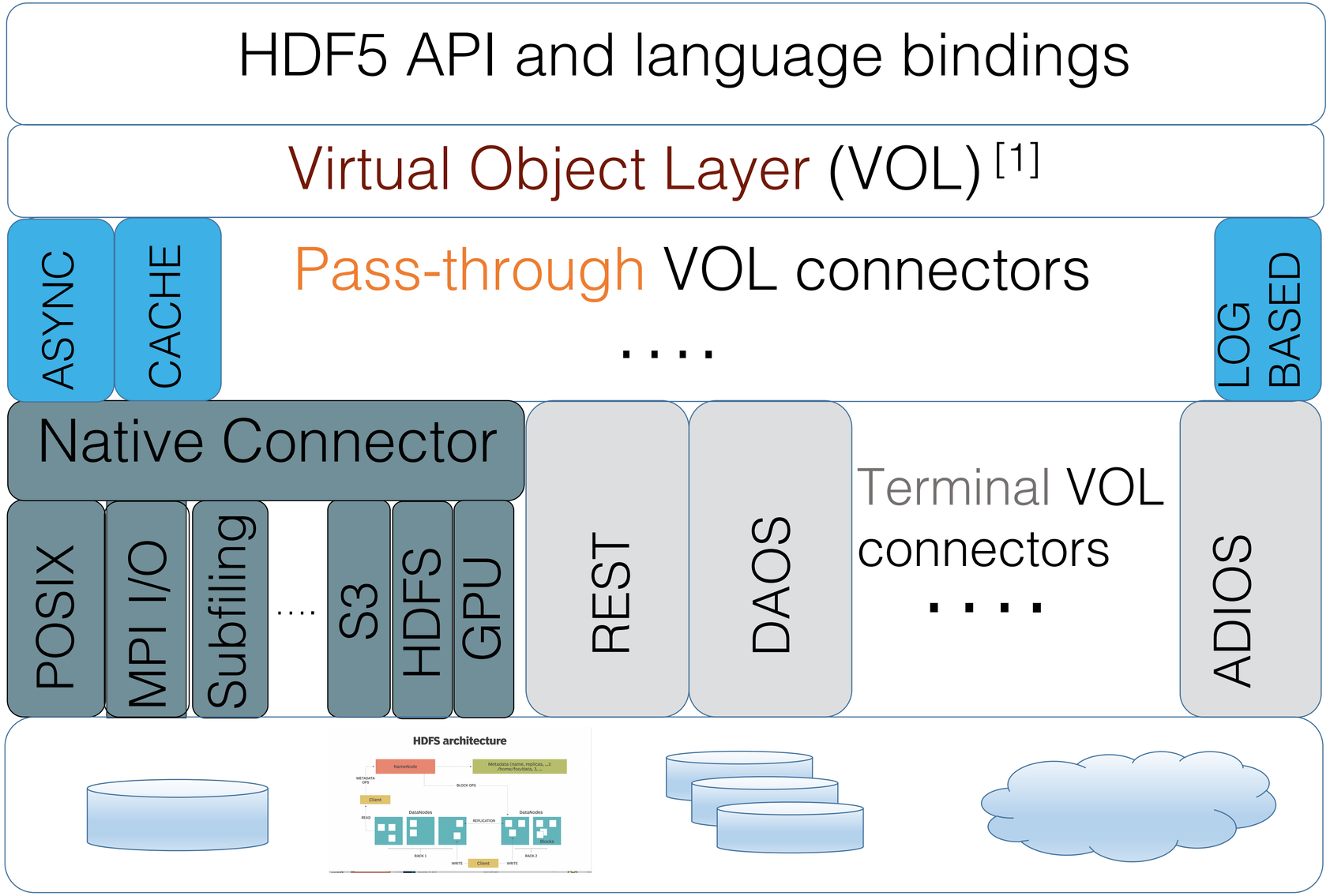
https://docs.hdfgroup.org/hdf5/develop/_h_d_f5_examples.html

# HDF5 Library Architecture (1.12.0 +)

The HDF Group

HDF5 Core Library

VFDs




[1] https://portal.hdfgroup.org/documentation/hdf5-docs/registered_vol_connectors.html

# HDF5 Programming model and API

# The General HDF5 API

The HDF Group

- C, FORTRAN, Java, and C++
- Routines begin with the 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**

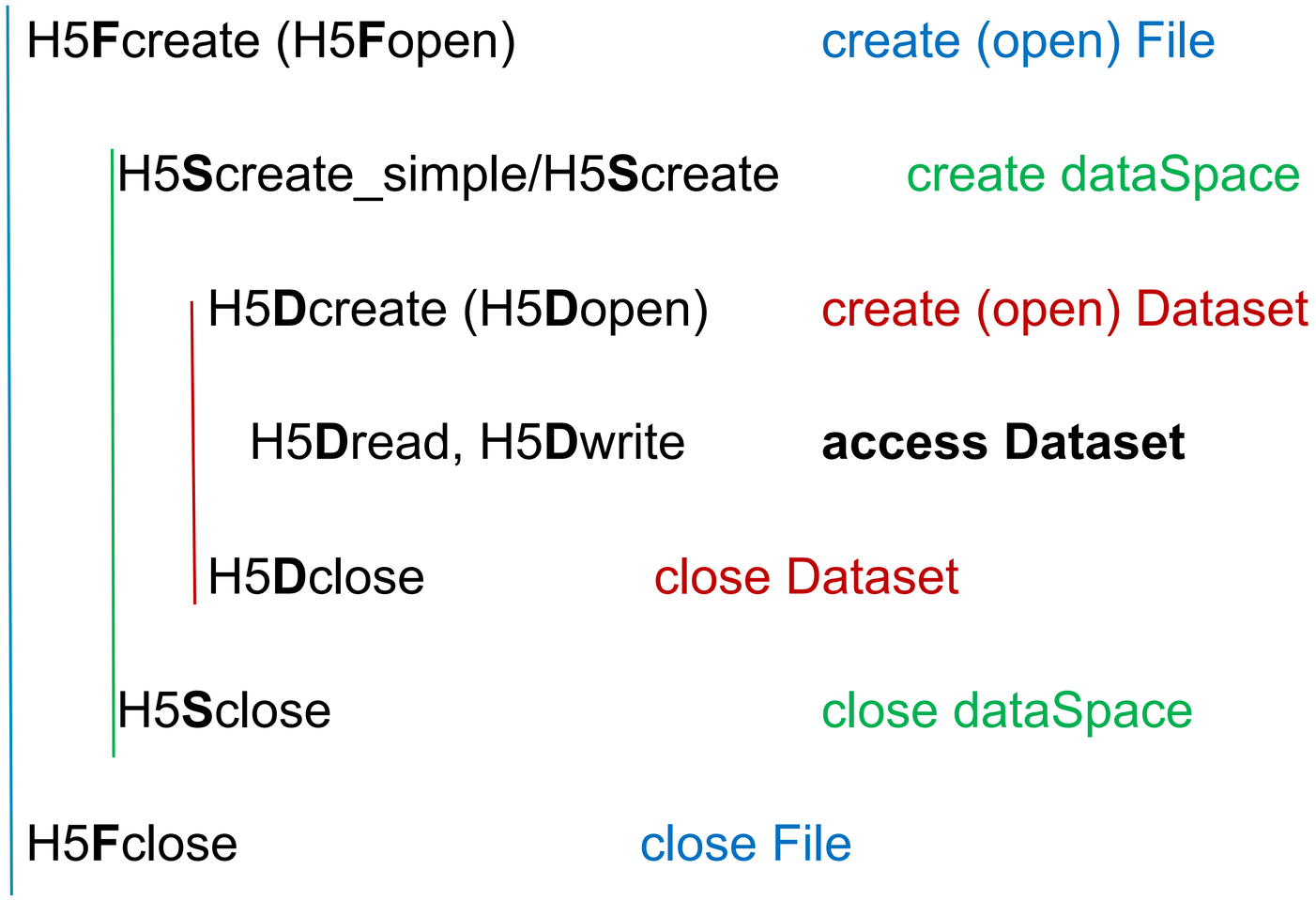
**H5S : data S**pace interface *e.g.,* **H5Sclose**

- The language wrappers follow the same trend

- There are more than 300 APIs – but one can start with less than 50

# General Programming Paradigm

- 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
- Properties (H5**P**) of an object are



# General best practices

# HDF5 Dataset I/O

The HDF Group

- Issue large I/O requests
  - At least as large as the file system block size
- Avoid **datatype conversion** ℹ️
  - Use the same data type in the file as in memory
  - If conversion is necessary, increase datatype conversion buffer size (default 1MB) with *H5Pset_buffer()*
- Avoid **dataspace conversion**
  - One dimensional buffer in memory to two-dimensional array in the file

ℹ️ Can break collective operations; check what mode was used H5Pget_mpio_actual_io_mode, and why H5Pget_mpio_no_collective_cause

# HDF5 Dataset - Storage

The HDF Group

- Use **contiguous storage** if no data will be added and compression is not used
  - HDF5 will not cache data
- Use **compact** storage when working with small data (<64K)
  - Data becomes part of HDF5 internal metadata and is cached (metadata cache)
- 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

# Terminology

The HDF Group

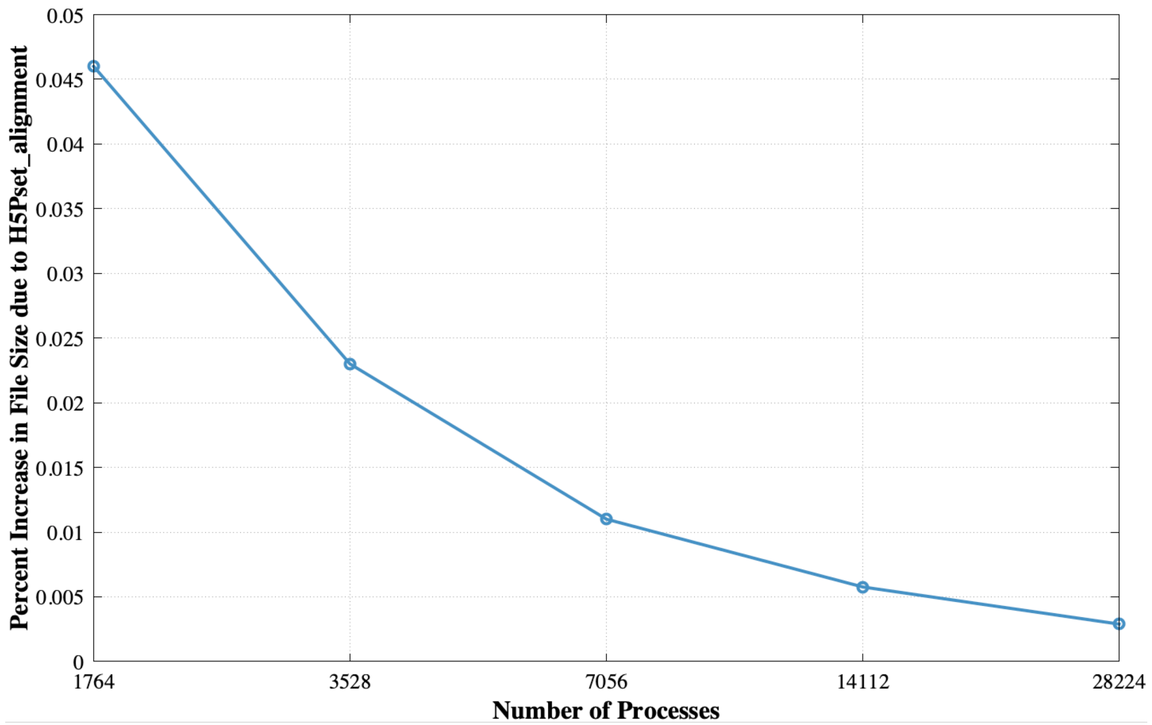
- DATA – “problem-size” data, e.g., large arrays
- METADATA – is an overloaded term
- In HDF5-centric context:

  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.

# General HDF5 Efficiency

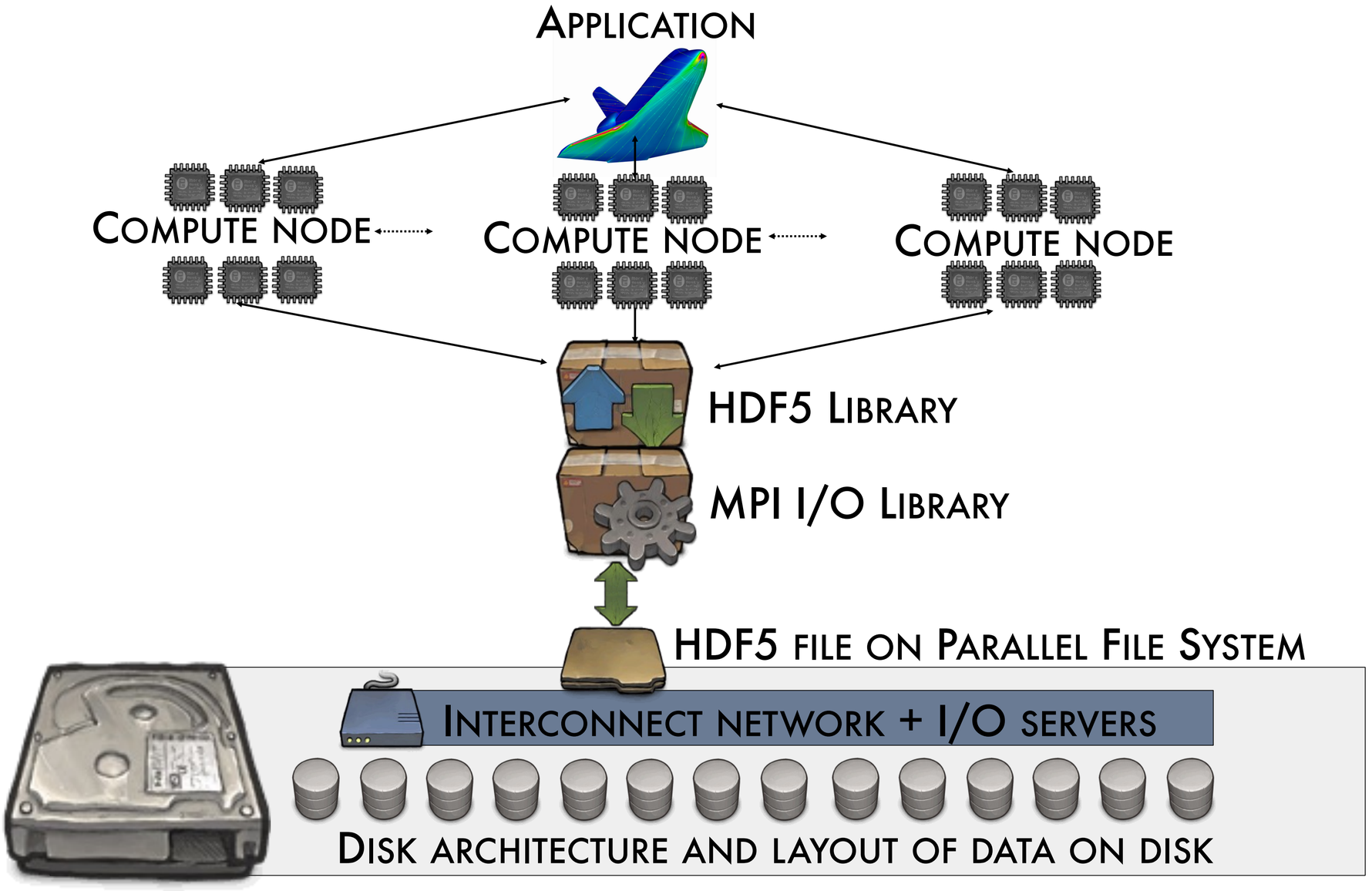
The HDF Group

- Faster HDF5 Performance: **Metadata**
  - Use the “latest” file format features
    - *H5Pset_libver_bounds()*
  - Increase the size of metadata data structures
    - *H5Pset_istore_k(), H5Pset_sym_k(), etc.*
  - Aggregate metadata into larger blocks
    - *H5Pset_meta_block_size()*
  - Align objects in the file
    - *H5Pset_alignment()*
  - Control metadata cache
  - Paged allocation and page buffering
    - Aggregate and align metadata and small data, perform I/O in aligned pages
    - See File Space Management Documentation https://portal.hdfgroup.org/display/HDF5/File+Space+Management

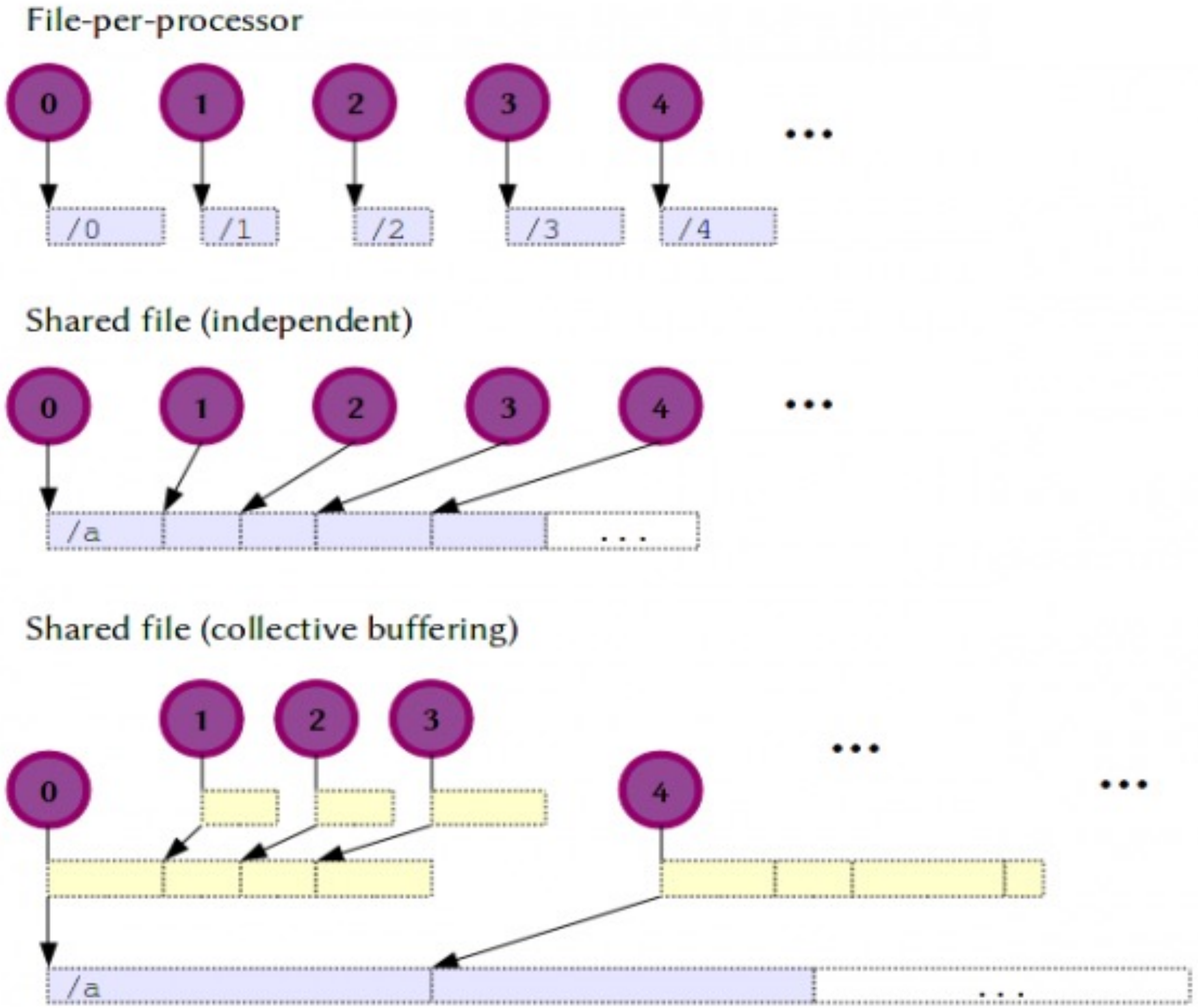


# Parallel I/O with HDF5

# PHDF5 implementation layers



# Types of Application I/O to Parallel File Systems



# Why Parallel HDF5?

The HDF Group

- 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

- Maintained code base, performance and data portability
  - Rely on HDF5 to optimize for the underlying storage system

# Parallel HDF5 (PHDF5) vs. Serial HDF5

The HDF Group

- PHDF5 allows multiple MPI processes in an MPI application to perform I/O to a single HDF5 file
- PHDF5 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 “schema”

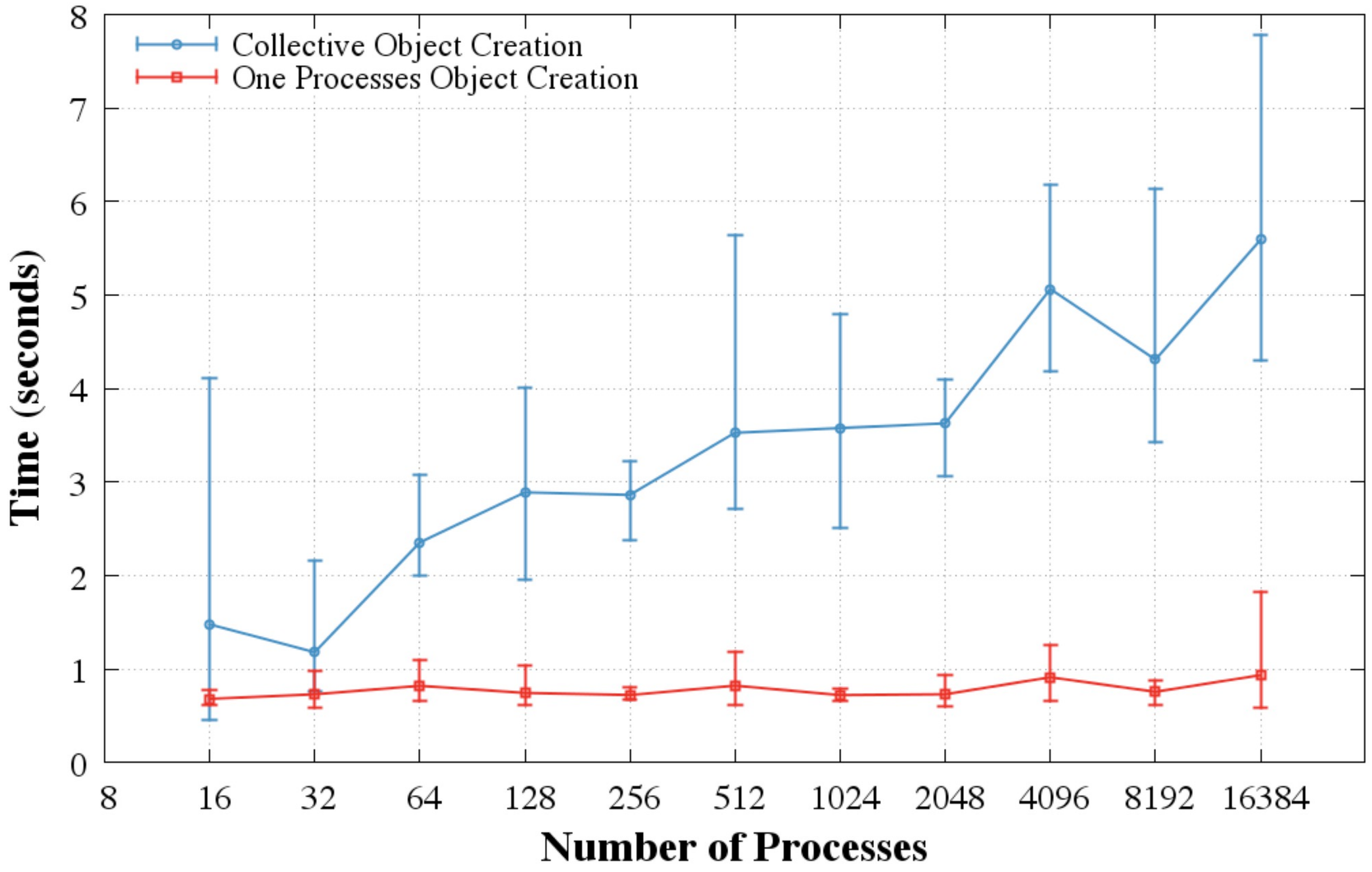
# Parallel HDF5 Schema

The HDF Group

- 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.
  - Raw data operations can either be collective or independent
    - For collective, all processes must participate, but they don't need to read/write data.

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

# Object Creation (Collective vs. Single Process)



# Collective vs. Independent Operations

The HDF Group

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

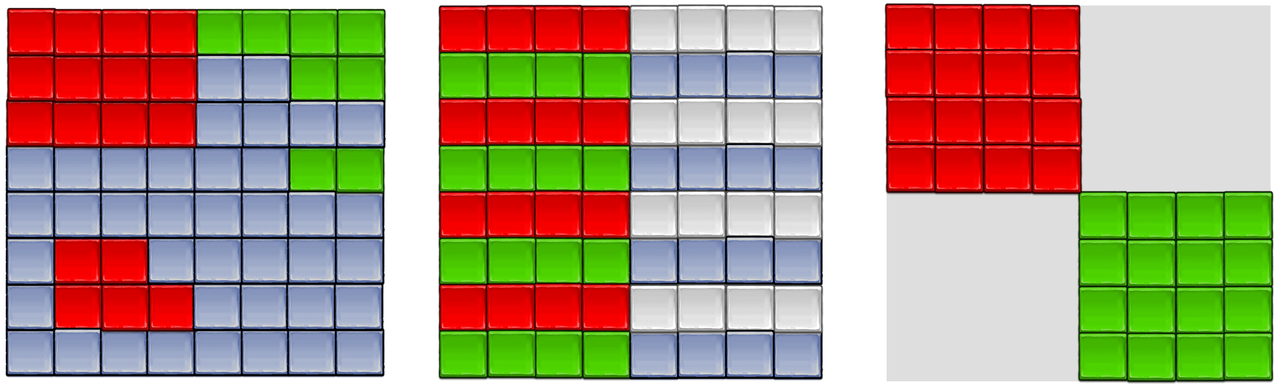
| | Process1 | Process2 | |
|---|---|---|---|
|  | **call A(); call B();** | **call A(); call B();** | **…CORRECT** |
|  | **call A(); call B();** | **call B(); call A();** | **…WRONG** |

- Collective I/O attempts to combine multiple smaller independent I/O ops into fewer larger ops; **neither mode is preferable *a priori***

# General HDF5 Programming Parallel Model for raw data I/O

The HDF Group

- Distributed memory model: data is split among processes
- Each process defines selections in memory and in file (aka HDF5 hyperslabs) using `H5Sselect_hyperslab`
- The hyperslab parameters define the portion of the dataset to write to
  - Contiguous hyperslab, Regularly spaced data (column or row), Pattern, or Blocks



- Each process executes a write/read call using selections, which can be either collective or independent

# General HDF5 Best Practices and Case Studies for Parallel Performance

# PHDF5 Fundamentals – A Simple Problem

The HDF Group

- Writing multiple 2D array variables over time:

**ACROSS P** processes arranged in a **R x C** process grid
  **FOREACH** step 1 .. **S**
   **FOREACH** count 1 .. **A**
    **CREATE** a double **ARRAY** of size **[X,Y]** | **[R*X,C*Y]** (Strong | Weak)
    (**WRITE** | **READ**) the **ARRAY** (**to** | **from**) an HDF5 file

# Fundamentals – Missing Information

The HDF Group

- How are the array variables represented in HDF5?
  - 2D, 3D, 4D datasets
  - Are the extents known a priori?
  - How are the dimensions ordered?
  - Groups?
- What order is the data written, and is the data read the same way?
- What's the storage layout?
  - How many physical files?
  - Contiguous or chunked, etc.
  - Is the data compressible?
- What's the file system or data store?
- Collective vs. independent MPI-IO

# One Kind of Performance Hurdle

The HDF Group

- HDF5 has a complex-looking interface
  - Complexity does not necessarily mean difficult to use
  - Users may require such complexity to achieve their goals
    - **Goal**: Self-describing share-friendly data layout
      - Tuning performance and efficiency with the constraint of using a standardized file format (netCDF, CGNS, etc.)
    - **Goal**: Fastest I/O possible
      - Tuning for check-points by minimizing metadata, large write blocks.
  - The complexity of the HDF5 workflow and underlying hardware may make the HDF5 tasks unavoidably complex.

# Other Sources of Performance Variability

The HDF Group

Hardware

System configuration and activity of other users

**HDF5 property (H5P) lists**

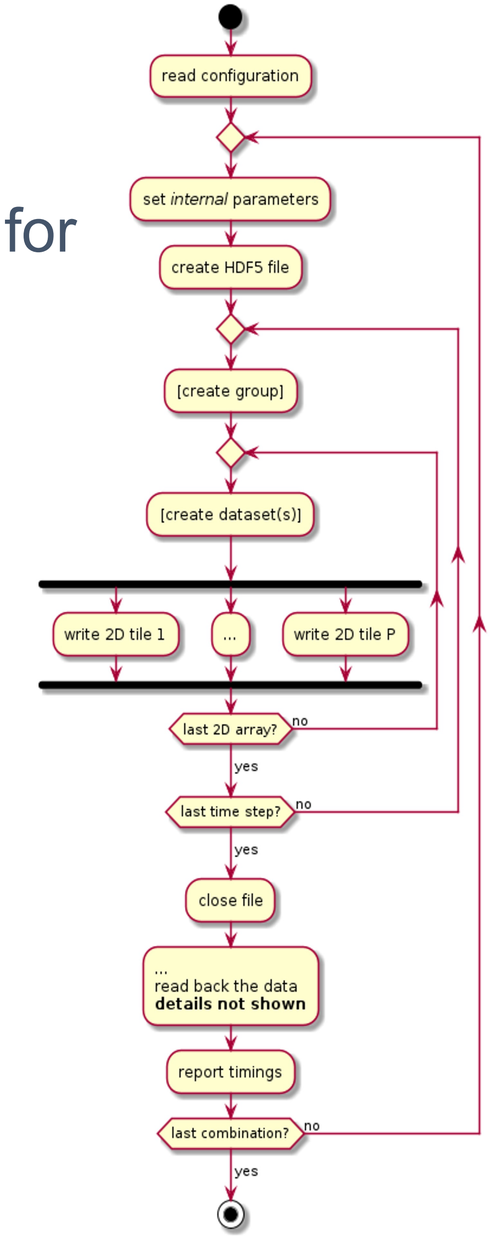
- Nearly 180 APIs
- Controls storage properties for HDF5 objects
- Controls in-flight HDF5 behavior
- About 100 *H5Pset_** functions
  - ≤ $p_1$ * … * $p_{100}$ combinations!
  - How many are tested?
- What does *H5P_DEFAULT* mean?
- What is the effect of using H5P_DEFAULT?

HDF5 User's Guide: File Property Lists

# Back to earlier example – Application Model

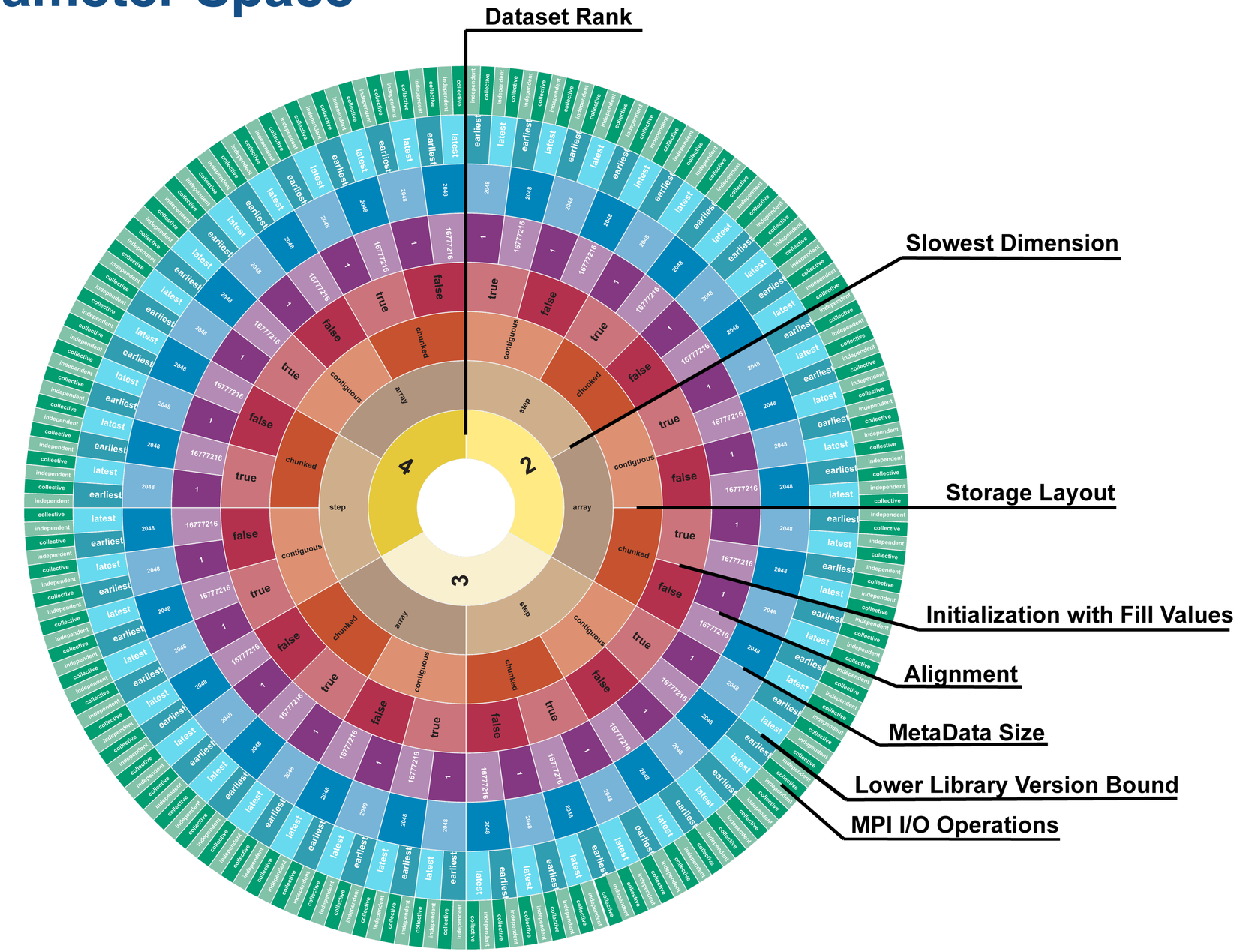
Good or bad news:

- There are *several* different ways to handle the data in HDF5, for example:
  - Many 2D datasets or attributes
  - A few 3D datasets
  - A 4D dataset
- There are many ways to use HDF5 properties
  - Chunking
  - Data alignment
  - Metadata block size
  - Collective/Independent I/O
- Ideally, performance would be more or less the same
- **HDF5 I/O**[1] test explores the HDF5 parameter space



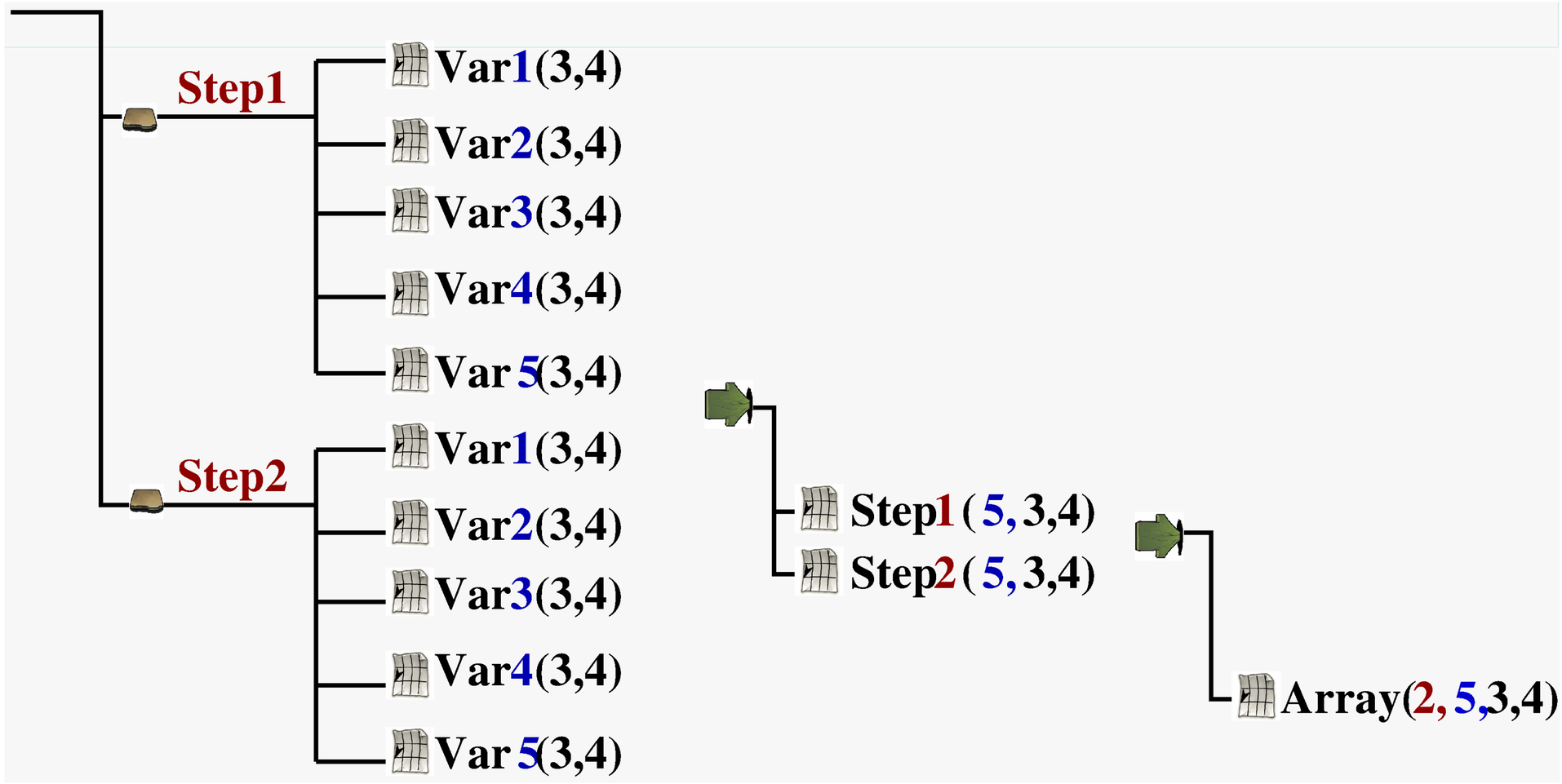

1 https://github.com/HDFGroup/hdf5-iotest

# HDF5 Parameter Space



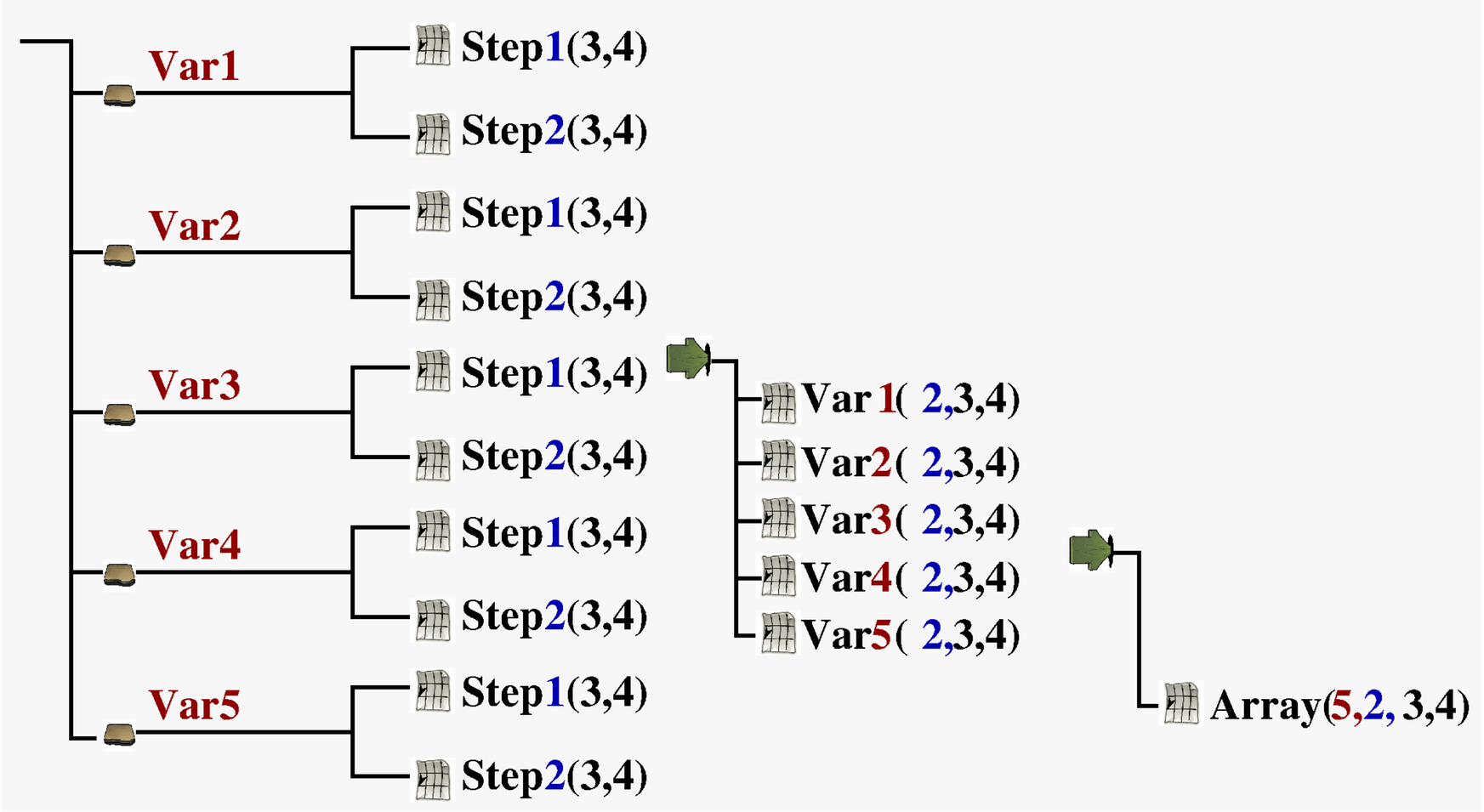
# IO Pattern Model

**Step** based IO Pattern

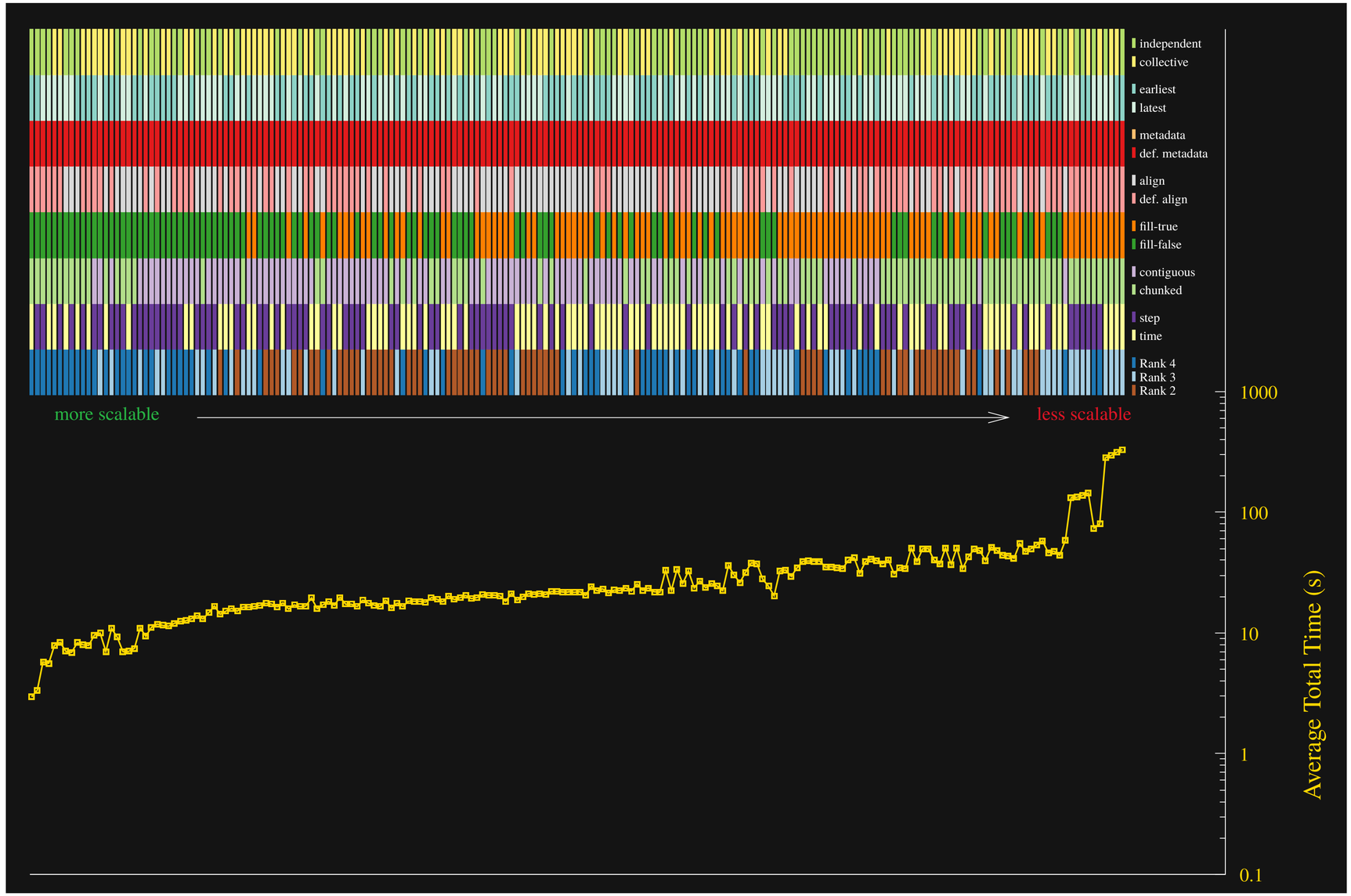


# IO Pattern Model

**Array** based IO Pattern



# Performance as a function of HDF5 parameter space




- Summit, weak scaling ( 42 to 2688)
- Best had:
  - four rank array (layout)
  - chunked
  - no fill values
  - default alignment
  - independent I/O

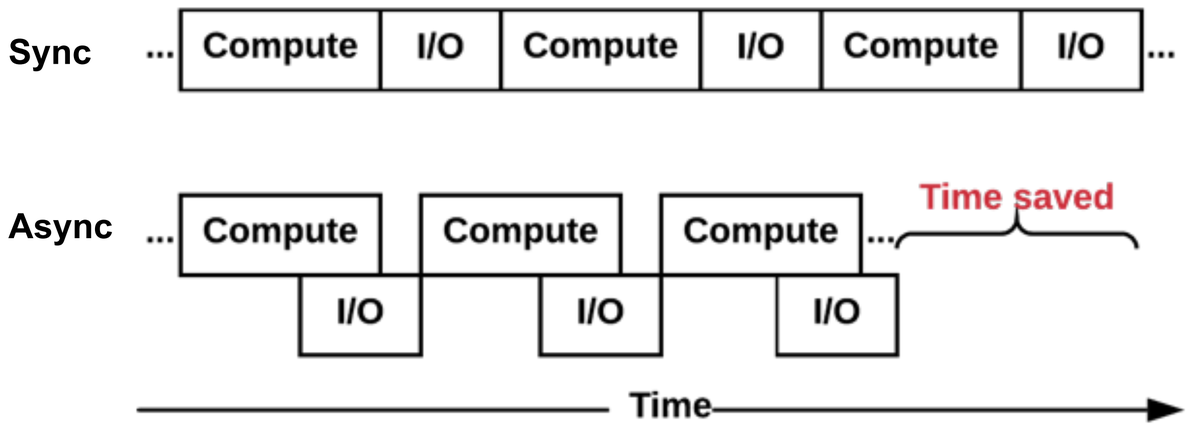
# Strongly Recommended Options

The HDF Group

- 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
  - When set, MPI rank 0 will issue the read for a metadata entry to the file system and broadcast to all other ranks

- Set HDF5 to never fill chunks (H5Pset_fill_time with H5D_FILL_TIME_NEVER)

# Features: Asynchronous I/O

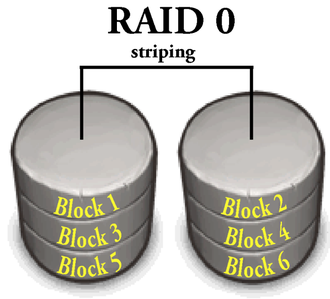



- Allows asynchronous operations for HDF5 applications:
  - Applications use the *_async* versions for the **H5** APIs
  - Return “request tokens” to applications to track I/O tasks.
- Requires a VOL (async or DAOS) which supports asynchronous I/O; otherwise, defaults to synchronous I/O.

# Subfiling

The HDF Group

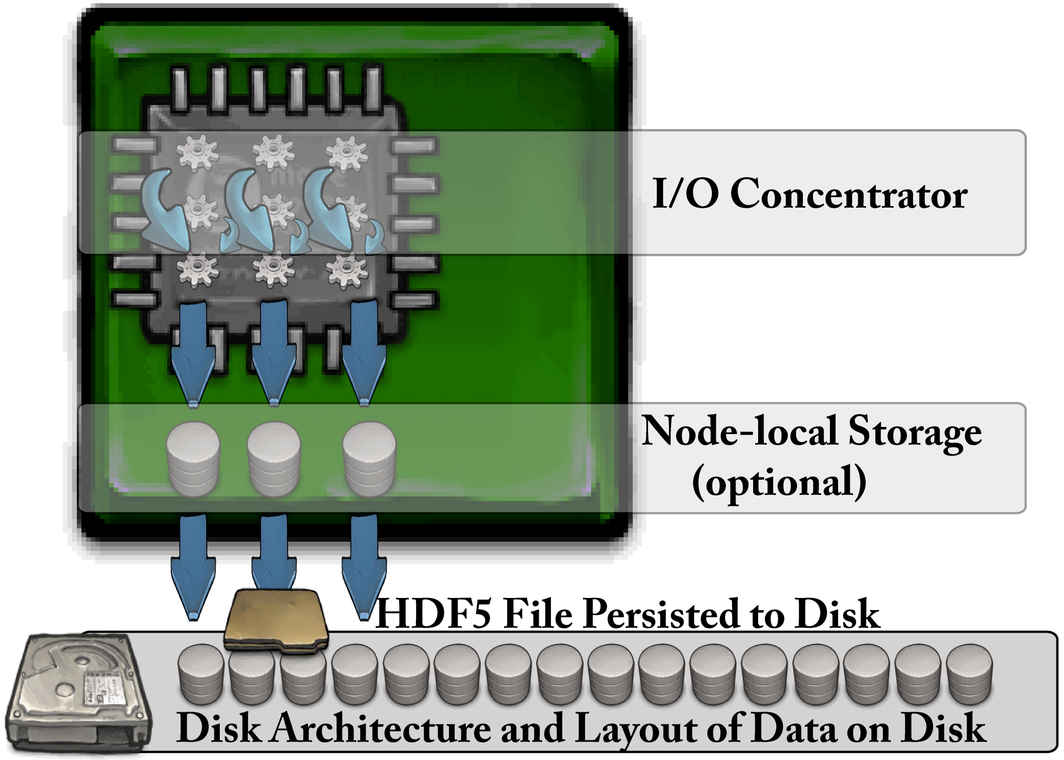
- Subfiling is a compromise between file-per-process (*fpp*) and a single shared file (*ssf*)
  - Use the Subfiling VFD, ***H5Pset_fapl_subfiling***(…),
    - use environment variables to control parameters




  - Multiple files organized as a Software RAID-0 Implementation
    i. Configurable “stripe-depth” and “stripe-set size”
    ii. A default “stripe-set” is created by using 1 file per node
    iii. A default “stripe-depth” is 32MB
      i. **export H5FD_SUBFILING_STRIPE_SIZE=$(( 16*1024*1024 ))**
    iv. Using Node-local storage
      i. **export H5FD_SUBFILING_CONFIG_FILE_PREFIX=<*Global File System Location*>**
      ii. **export H5FD_SUBFILING_SUBFILE_PREFIX="/tmp"**

    v. The resulting collection can be read using Subfiling, or fused together using *h5fuse* into a single HDF5 file.
- Benefits
  - Better use of parallel I/O subsystem (node-local storage)
  - Reduces the complexity of *fpp*
  - Reduced locking and contention issues to improve performance at larger processor counts over *ssf*
  - Available in *HDF5 1.14.0*

# Subfiling

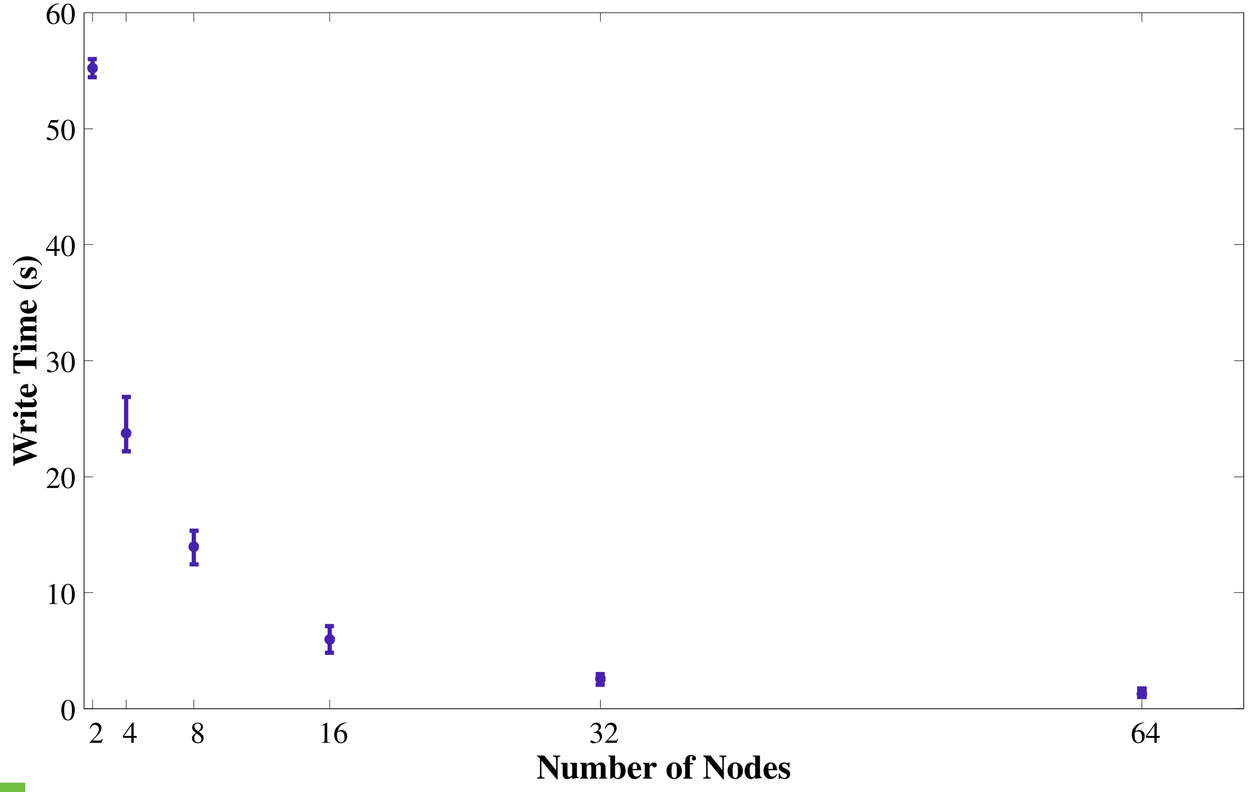


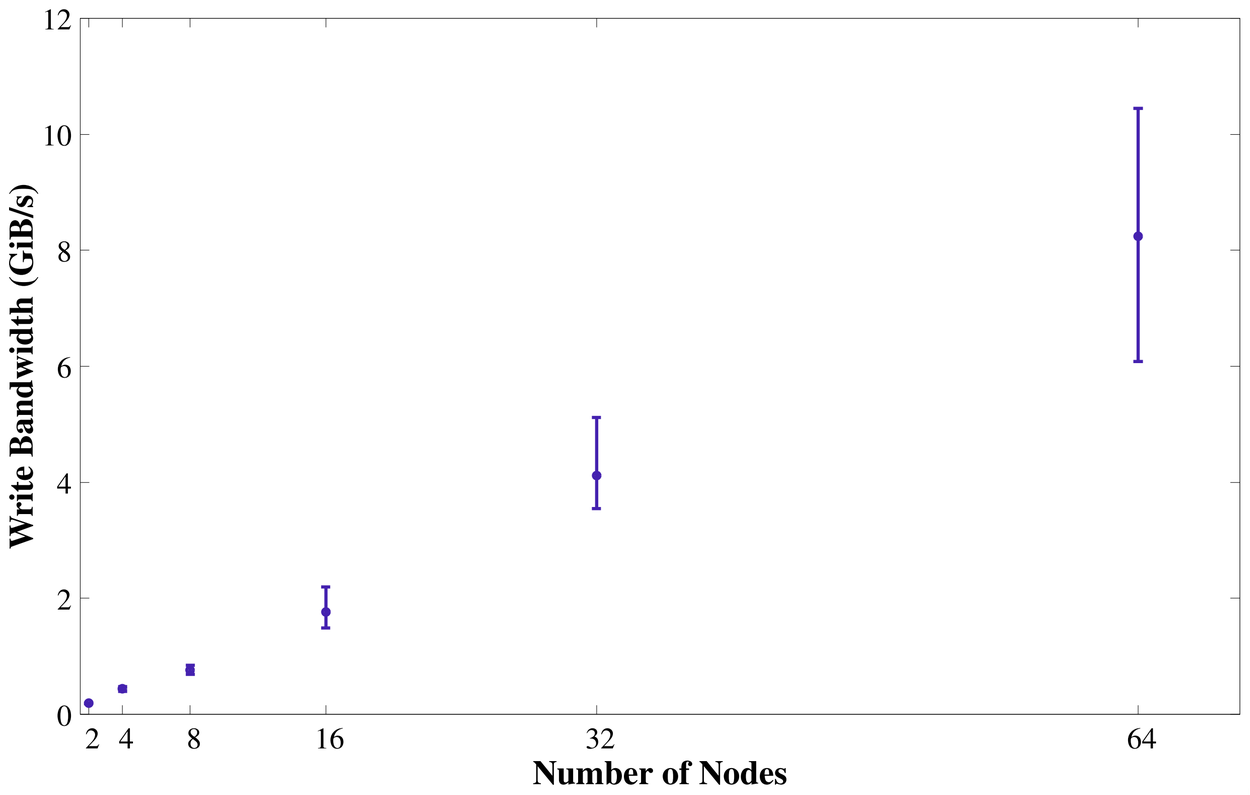

a. I/O Concentrators are implemented as independent threads attached to a normal HDF5 process.
b. MPI is utilized for communicating between HDF5 processes and the set of I/O Concentrators.
c. Because of (b), applications need to use *MPI_Init_thread* to initialize the MPI library.
d. Currently does not support collective I/O

# Subfiling

The HDF Group

- (CGNS[1] benchmark_hdf5)
  - One subfile per node, 16MiB striping, to node-local storage.
  - Fixed-size problem, 108 GiB file.






[1] CGNS = Computational Fluid Dynamics (CFD) General Notation System, cgns.org

# GPU VFD

The HDF Group

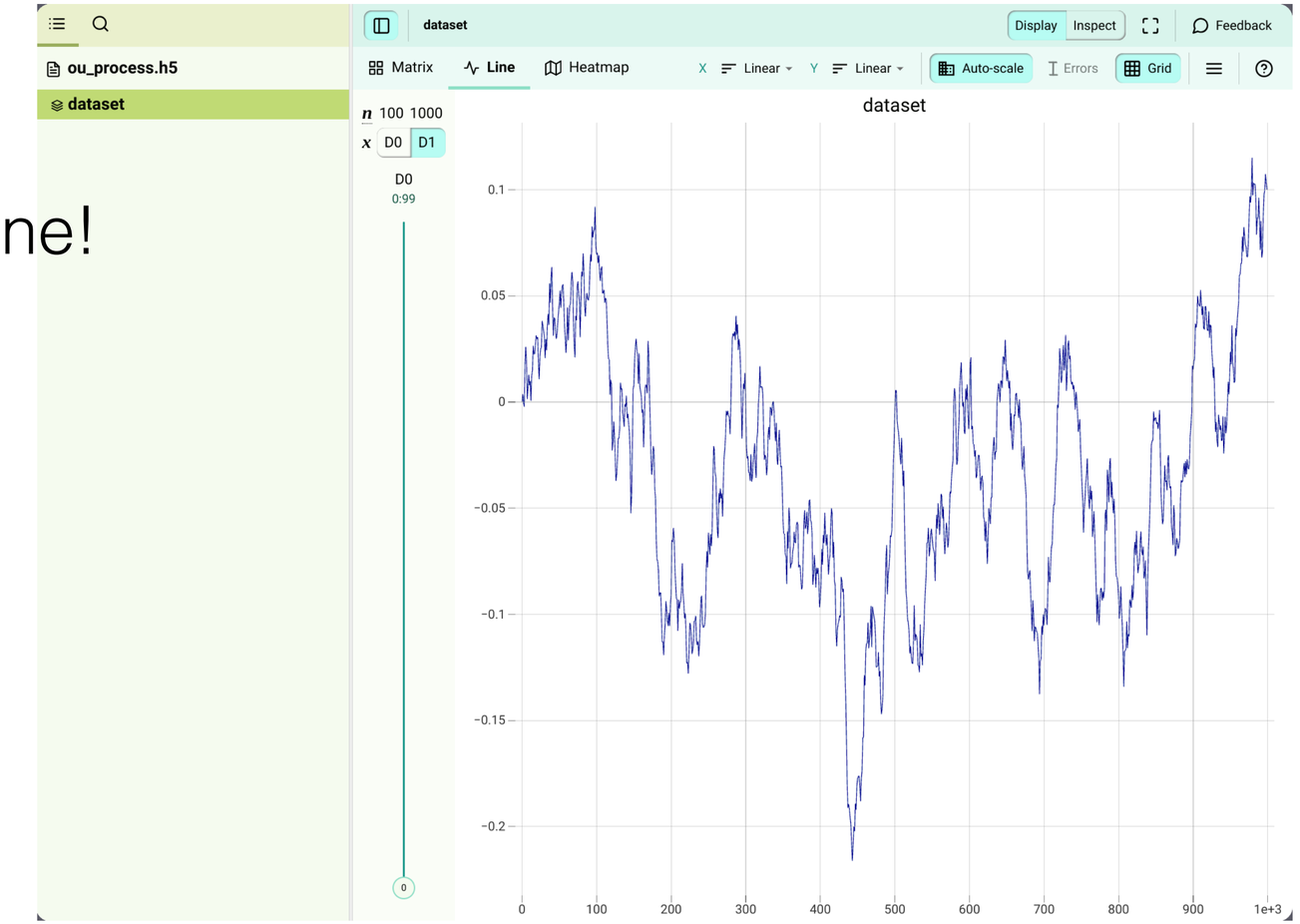
- The HDF5 GPUDirect Storage VFD is used to interface with Nvidia's GPUDirect Storage (GDS) API.
  - Available at https://github.com/hpc-io/vfd-gds
  - Your application should make use of CUDA memory management API routines (e.g., cudaMalloc) to allocate memory on the device.
  - The device-allocated memory buffer should be passed to H5Dread/H5Dwrite

Limitations

- The VFD should still be considered a prototype with new features under development.
- Contact us if you wish to be informed of new features that may fit your use case.
- The VFD does not support MPI parallel-based applications.
- **Delta**: Tested with openmpi/5.0.1+cuda and gcc/11.4.0 modules
  - Testing with the Nvidia module is being done.

# Viewing HDF5 Data in a Web Browser

- Point your browser at https://myhdf5.hdfgroup.org/
- Open from URL: ou_process.h5 from the HDF5 tutorial
- Kudos
  - H5Web team (ESRF)
  - h5wasm (NIST)
- Your data never leaves your machine!
  - Depends on browser file system
  - Keep that in mind (size)
- Demo time!



# A Tool for Dealing with Many Small Objects - `tar2h5`

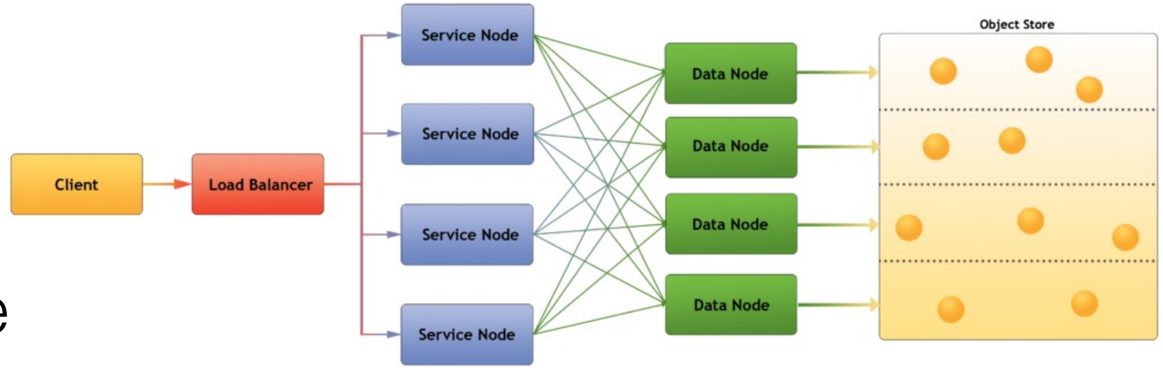
The HDF Group

- Original presentation by Dawei Mu and Volodymyr Kindratenko
- Large collections of small objects (images, a/v clips, etc.)
- Tremendous pressure on the parallel file system's MD server
- **Not a bad idea:** Collect them in a TAR archive
  - Zillions of files -> one file
- **But:** No random or parallel access
- **A better idea:** Store them in a single HDF5 file
  - Compression
  - Deduplication
- GitHub
- Demo time!

# HDF5 as a Highly Scalable Data Service (HSDS)

The HDF Group

- HDF5 is about data sharing
- Yes, you can share files, but …
  - File size could be unwieldy
  - You need just a small part of a file
  - Share w/ multiple people
    - Access control
- Certain use cases don't fit well w/ the HDF5 library
  - Can't call a native binary
  - Cloud-based infrastructure
  - Multiple writers and readers
- No problem: The HDF5 data model can be implemented in other ways
- Demo time!




HSDS can be run on most container management systems:


Using different supported storage systems:


POSIX
Filesystem

# HDF Resources

The HDF Group

- Google Scholar - about 26,500 citations since 1998
- YouTube channel
- HDF clinic - Tuesday's 12:20 PM Central Time
- HDF Forum
- HDF Users Group (HUG)
- Help desk
- HDF5 tutorial

Contact: help@hdfgroup.org

# THANK YOU!

Questions & Comments?