



CSE 5449: Intermediate Studies in Scientific Data Management

Lecture 5: I/O Software stack – HDF5

Dr. Suren Byna

The Ohio State University

E-mail: byna.1@osu.edu

<https://sbyna.github.io>

01/24/2023



Summary of the last class

- Parallel I/O software stack
- An intro to HDF5



Today's class

- More details about HDF5 API and the library

- Parallel HDF5



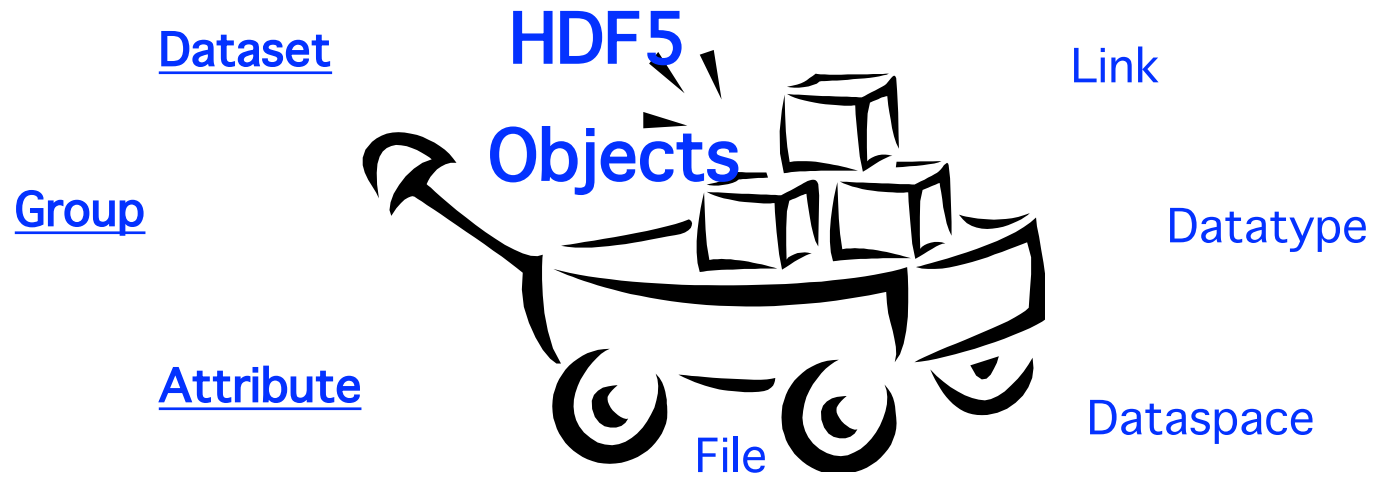
Class projects

5. Performance comparison of sub-filing in HDF5 and PnetCDF

- Background: Sub-filing is an approach to split a very large file into smaller files. However, there are pros / cons with the approach on how the data is organized.
- Question
 - Which of the HDF5 and PnetCDF sub-filing approaches are best?
 - What better strategies for sub-filing are there?
- Deliverable: A short paper describing
- Resources
 - Tuning HDF5 subfiling performance on parallel file systems
<https://escholarship.org/content/qt6fs7s3jb/qt6fs7s3jb.pdf>
 - Using Subfiling to Improve Programming Flexibility and Performance of Parallel Shared-file I/O
<https://ieeexplore.ieee.org/document/5362452>
 - Scalable Parallel I/O on a Blue Gene/Q Supercomputer Using Compression, Topology-Aware Data Aggregation, and Subfiling <https://ieeexplore.ieee.org/document/6787260>



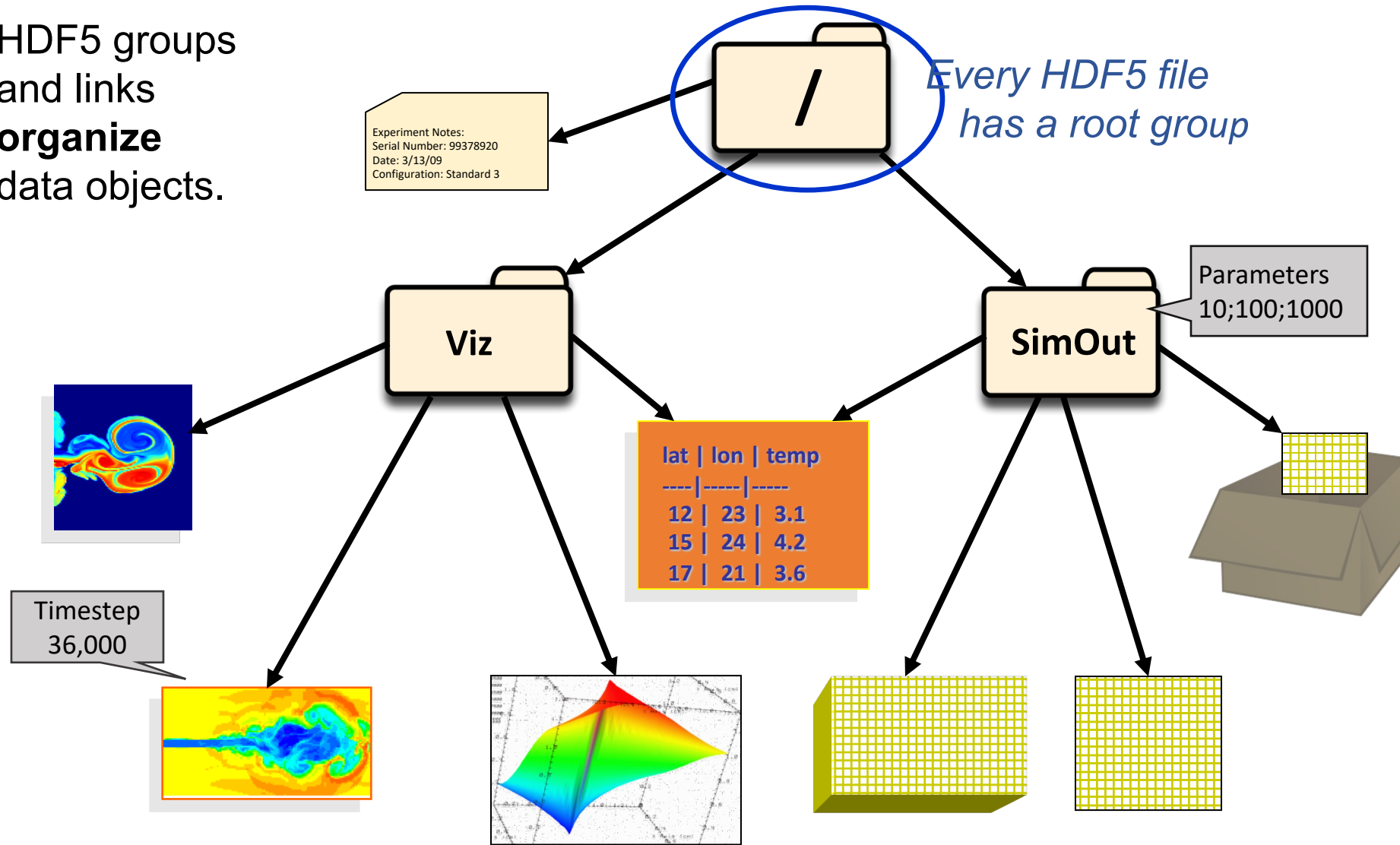
HDF5 Data Model





HDF5 Groups and Links

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





Homework

- Install HDF5 on your laptop or on OSC
- Go to https://docs.hdfgroup.org/hdf5/develop/_h_d_f5_examples.html
 - Run the [Examples from Learning the Basics](#) page
 - Report the observations in the next class



HDF5 Programming model and API



The General HDF5 API

- C, Fortran, Java, C++, and .NET bindings
 - Also: IDL, MATLAB, Python (H5Py, PyTables), Perl, ADA, Ruby, ...
- C routines begin with prefix: H5?
 - ? is a character corresponding to the type of object the function acts on

Example Functions:

H5D : Dataset interface *e.g.*, **H5Dread**
H5F : File interface *e.g.*, **H5Fopen**
H5S : dataSpace interface *e.g.*, **H5Sclose**



The HDF5 API

- For flexibility, the API is extensive
 - ✓ 300+ functions
- This can be daunting... but there is hope
 - ✓ A few functions can do a lot
 - ✓ Start simple
 - ✓ Build up knowledge as more features are needed



Victorinox
Swiss Army
Cybertool 34





General Programming Paradigm

- Object is opened or created
- Object is accessed, possibly many times
- Object is closed

- Properties of object are optionally defined
 - ✓ Creation properties (e.g., use chunking storage)
 - ✓ Access properties



Basic Functions

H5**F**create (H5**F**open)

create (open) File

H5**S**create_simple/H5**S**create

create dataSpace

H5**D**create (H5**D**open)

create (open) Dataset

H5**D**read, H5**D**write

access Dataset

H5**D**close

close Dataset

H5**S**close

close dataSpace


H5**F**close

close File



Other Common Functions

D ata S paces:	H5Sselect_hyperslab (Partial I/O) H5Sselect_elements (Partial I/O) H5Dget_space
D ata T ypes:	H5Tcreate, H5Tcommit, H5Tclose H5Tequal, H5Tget_native_type
G roups:	H5Gcreate, H5Gopen, H5Gclose
A tttributes:	H5Acreate, H5Aopen_name, H5Aclose H5Aread, H5Awrite
P roperty lists:	H5Pcreate, H5Pclose H5Pset_chunk, H5Pset_deflate



```
/*  
 * This example illustrates how to create a dataset that is a 4 x 6  
 * array. It is used in the HDF5 Tutorial.  
 */
```

```
#include "hdf5.h"  
#define FILE "dset.h5"
```

```
int  
main()  
{
```

```
    hid_t    file_id, dataset_id, dataspace_id; /* identifiers */  
    hsize_t  dims[2];  
    herr_t   status;
```

```
    /* Create a new file using default properties. */  
    file_id = H5Fcreate(FILE, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);
```

```
    /* Create the data space for the dataset. */  
    dims[0]    = 4;  
    dims[1]    = 6;  
    dataspace_id = H5Screate_simple(2, dims, NULL);
```

```
    /* Create the dataset. */  
    dataset_id =  
        H5Dcreate2(file_id, "/dset", H5T_STD_I32BE, dataspace_id, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
```


```
    /* End access to the dataset and release resources used by it. */  
    status = H5Dclose(dataset_id);
```

```
    /* Terminate access to the data space. */  
    status = H5Sclose(dataspace_id);
```

```
    /* Close the file. */  
    status = H5Fclose(file_id);
```

```
}
```

https://raw.githubusercontent.com/HDFGroup/hdf5/hdf5_1_10/examples/h5_crtdat.c



```
/*
 * This example illustrates how to write and read data in an existing
 * dataset. It is used in the HDF5 Tutorial.
 */
```

```
#include "hdf5.h"
#define FILE "dset.h5"
```

```
int
main()
{
```

```
    hid_t file_id, dataset_id; /* identifiers */
    herr_t status;
    int i, j, dset_data[4][6];
```

```
    /* Initialize the dataset. */
    for (i = 0; i < 4; i++)
        for (j = 0; j < 6; j++)
            dset_data[i][j] = i * 6 + j + 1;
```

```
    /* Open an existing file. */
    file_id = H5Fopen(FILE, H5F_ACC_RDWR, H5P_DEFAULT);
```

```
    /* Open an existing dataset. */
    dataset_id = H5Dopen2(file_id, "/dset", H5P_DEFAULT);
```

```
    /* Write the dataset. */
    status = H5Dwrite(dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

```
    status = H5Dread(dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

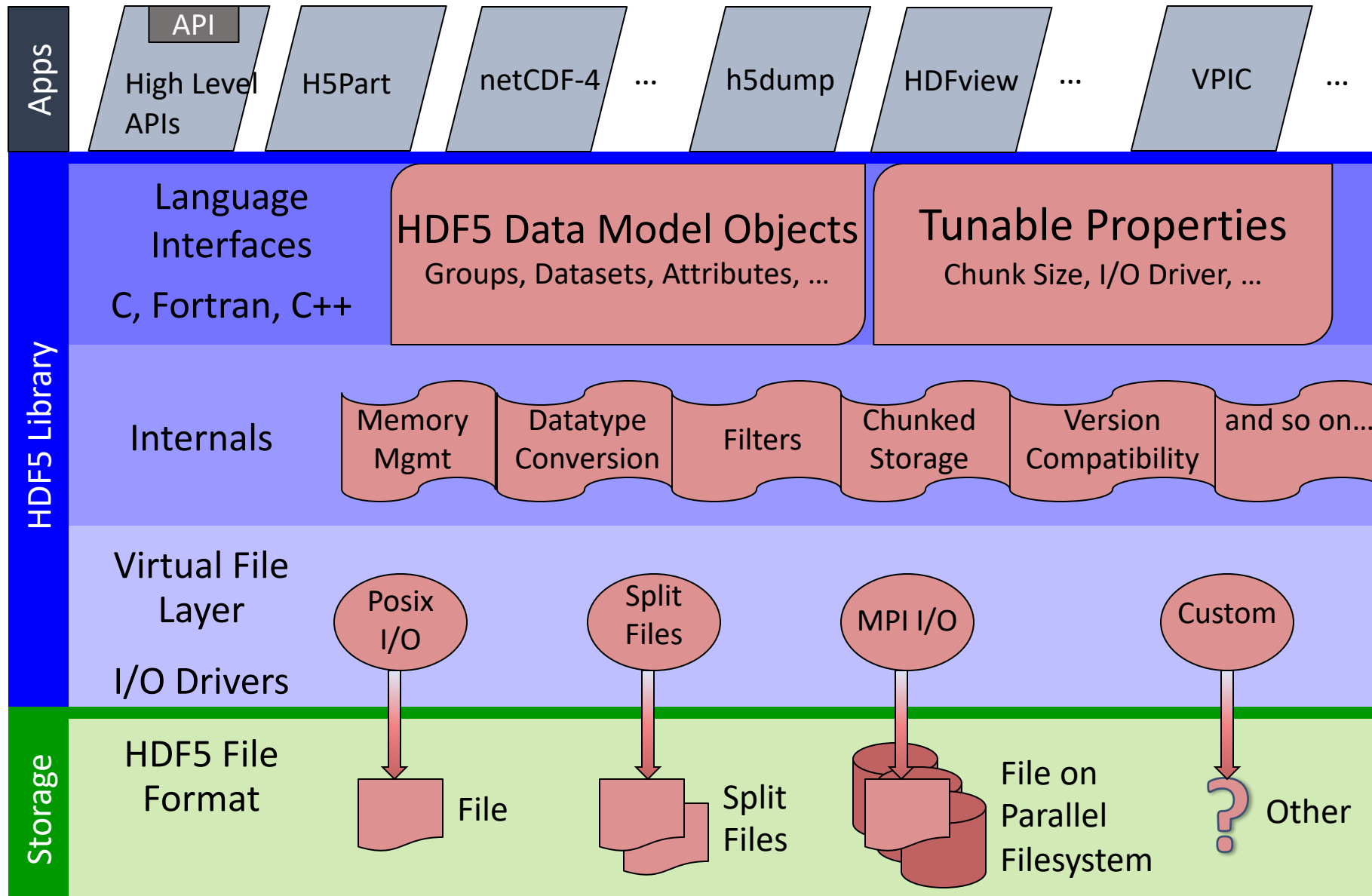
```
    /* Close the dataset. */
    status = H5Dclose(dataset_id);
```

```
    /* Close the file. */
    status = H5Fclose(file_id);
```

```
}
```

https://raw.githubusercontent.com/HDFGroup/hdf5/hdf5_1_10/examples/h5_rdwt.c

HDF5 Software Layers & Storage





Homework – Parallel programming with MPI

- On OSC, run MPI programming examples
 - https://www.osc.edu/resources/available_software/software_list/mvapich2
- <https://mpitutorial.com/tutorials/>
 - Run MPI Send and Receive
 - <https://mpitutorial.com/tutorials/mpi-send-and-receive/>
 - Run collective communication codes
 - <https://mpitutorial.com/tutorials/mpi-broadcast-and-collective-communication/>



Parallel HDF5



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 likely many associated pieces of HDF5 metadata
 - There are also other sources of HDF5 metadata



Why Parallel HDF5?

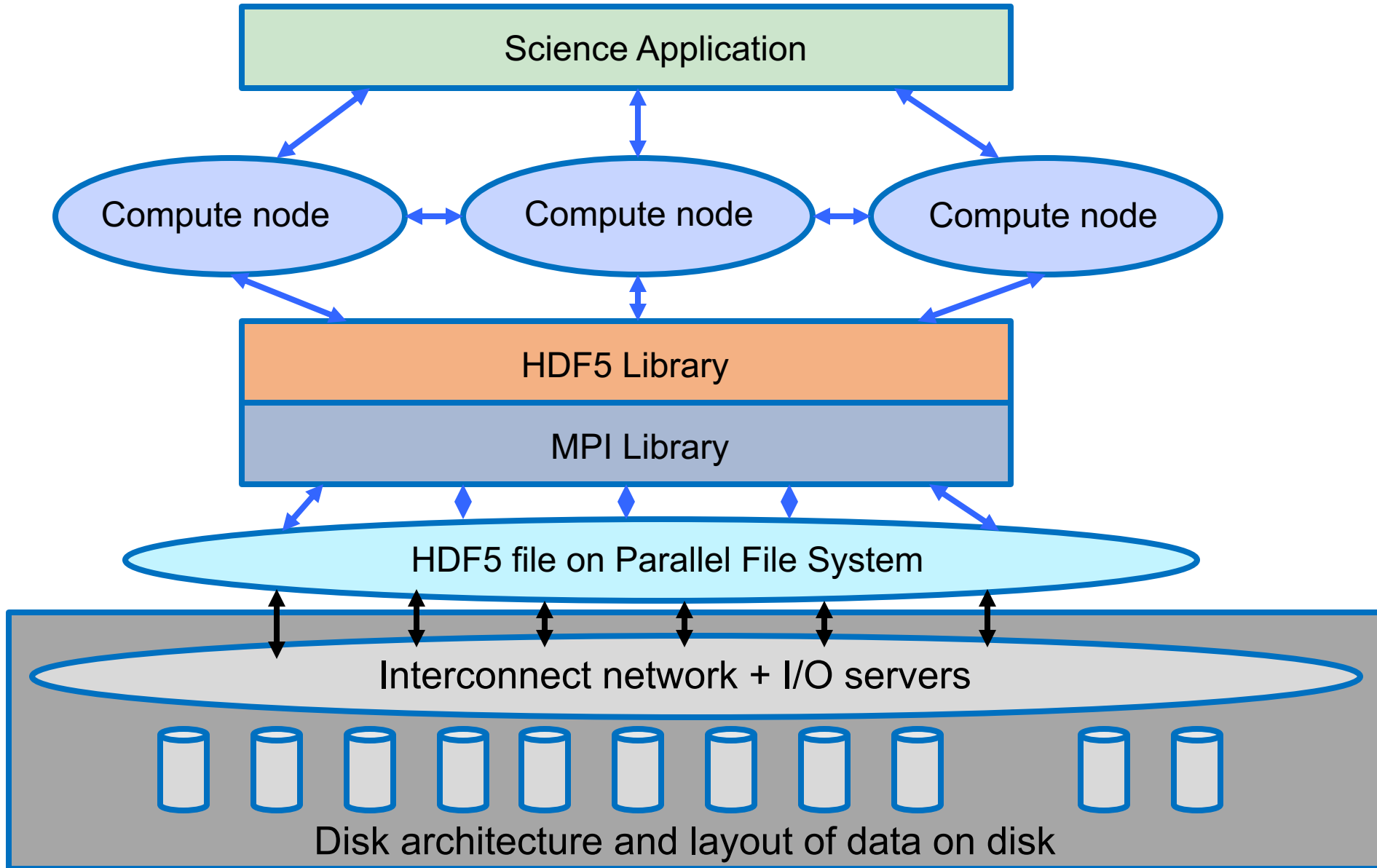
- Take advantage of high-performance parallel I/O while reducing complexity
 - Add a well-defined layer to the I/O stack
 - Produce a single or a few shared files
 - *“Friends don’t let friends use file-per-process”*
- Make performance portable



(MPI-)Parallel vs. Serial HDF5

- PHDF5 allows multiple MPI processes in an MPI communicator 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
 - <http://support.hdfgroup.org/HDF5/doc/H5.format.html>
- The PHDF5 API consists of:
 - The standard HDF5 API
 - A few extra knobs and calls
 - A parallel “etiquette”
- Bottom line: PHDF5 is as user-friendly as HDF5

PHDF5 Implementation Layers





Collective vs. Independent I/O

- MPI definition of collective calls: (not just I/O calls)
 - All processes of a communicator must participate in calls in the right order:

Process1

A(); → B();

A(); → B();

Process2

A(); → B();

B(); → A();

****right****

****wrong****

- Collective I/O is not necessarily synchronous, nor must it require communication
- Neither mode is preferable *a priori*.

Collective I/O: Attempts to combine multiple smaller independent I/O ops into fewer larger ops.



PHDF5 Etiquette

- PHDF5 opens a shared file with an MPI communicator
 - Returns a file handle
 - All future access to the file via that file handle
- All processes must participate in collective PHDF5 APIs
- Different files can be opened via different communicators
- **All** HDF5 APIs that modify file structure are collective!
 - Object create / delete, attribute and group changes, etc.
 - <http://support.hdfgroup.org/HDF5/doc/RM/CollectiveCalls.html>



Summary of today's class

- HDF5 API
- Parallel HDF5
- Homework
 - Try MPI programming examples
- Class projects
 - Present an initial project execution plan

After the class, slides are uploaded to: <https://osu.instructure.com/courses/141406/files>

Also available at: <https://sbyna.github.io/teaching/5449-sdm.html>