CSE 5449: Intermediate Studies in Scientific Data Management

Lecture 6: Parallel HDF5

Dr. Suren Byna

The Ohio State University

E-mail: byna.1@osu.edu

https://sbyna.github.io

01/26/2023

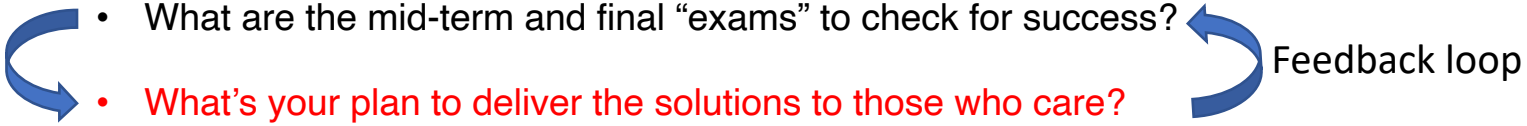
The Heilmeier Catechism

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you are successful, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

George H. Heilmeier, a former DARPA director from 1975 to 1977

The Heilmeier Catechism – My adjustments in performing research

- What's the problem? Articulate the problem using absolutely no jargon.
- Who cares? If you are successful, what difference will it make?
- How is it done today, and what are the limits / challenges of current practice?
- What is new in your approach, how different is it from existing work, and why do you think it will be successful?
- What are the risks and what's your risk mitigation plan?
- What's your execution plan?
- How long will it take?
- What are the mid-term and final "exams" to check for success?
- What's your plan to deliver the solutions to those who care?
- Retrospect!



George H. Heilmeier, a former DARPA director from 1975 to 1977

CS Research – Suren Byna's approach

- Problem
  - What's the problem
  - Who's complaining about it (Customers)
  - What's the benefit / improvement if the problem is solved
  - Get feedback from customers
- Gap analysis
  - What are the current approaches and gaps?
  - Define problem in detail and what are the challenges in solving it?
  - Get feedback from customers
- Solution
  - What's the solution?
  - What's the evaluation strategy to demonstrate benefit with the solution?
  - What's the design?
    - benchmarks
    - solution
    - real applications
  - How to deploy it?
  - Implement prototypes
  - Get feedback from customers
- Implement, Demonstrate, and Deploy
  - How to implement and resolve challenges
  - Test improvement with benchmarks
  - Demonstrate the benefit / improvement
  - Get feedback from customers
  - Deploy in real systems

Today's class

- Class project – Draft execution plan presentation
- More details about 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 Subfiling presentation:
    - https://www.hdfgroup.org/wp-content/uploads/2022/09/HDF5-Subfiling-VFD.pdf
    - https://www.youtube.com/watch?v=psl2iZmP2SY
  - PNetCDF subfiling
    - http://cucis.eecs.northwestern.edu/projects/PnetCDF/subfiling.html

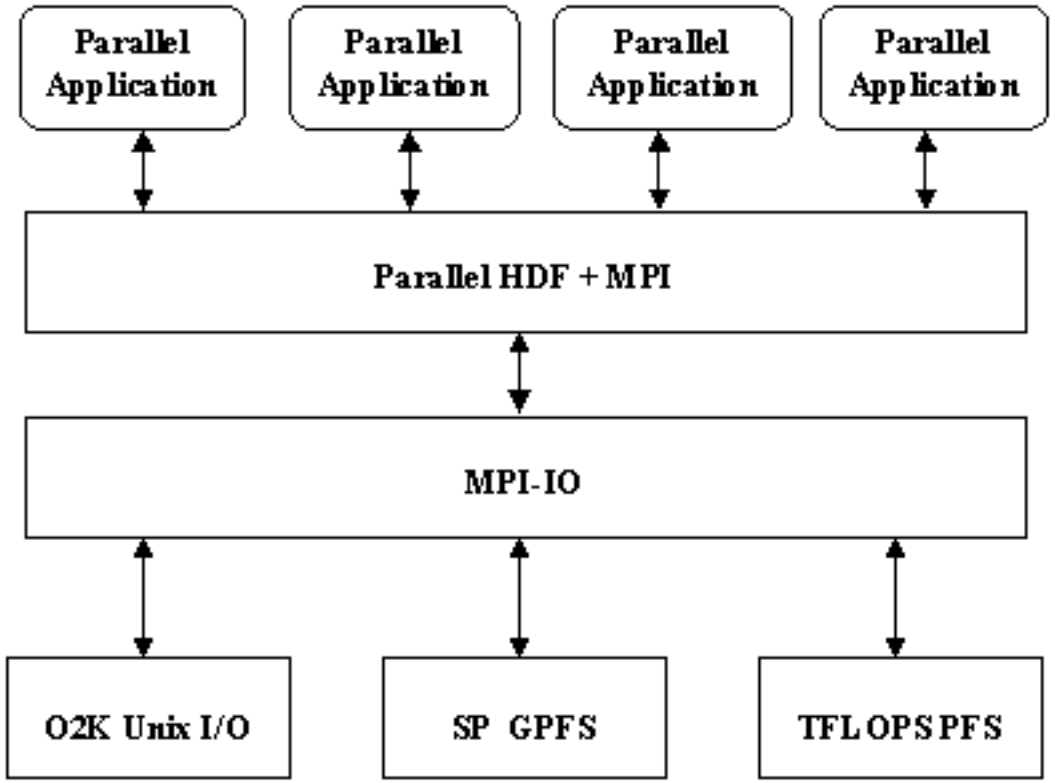
Homework – Parallel programming with MPI

- 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

HDF5 Parallel I/O Stack



User Applications

HDF library

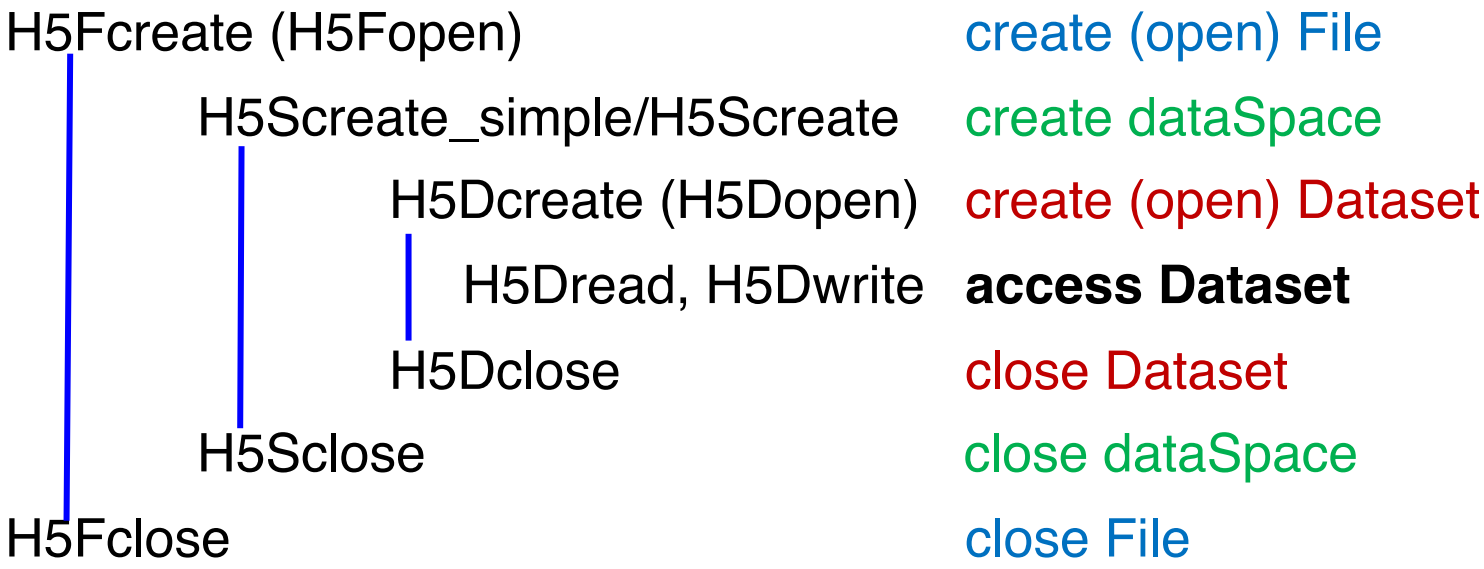
Parallel I/O layer

Parallel Filesystems

Important parallel HDF5 APIs

- Set file access property list (FAPL) to use MPI communicator
  - H5Pset_fapl_mpio(hid_t fapl_id, MPI_Comm comm, MPI_Info info);
  - *MPI_Comm* → MPI communicator
    - If all processes will access the file, use MPI_COMM_WORLD
  - *MPI_Info* → MPI Info object for passing hints about I/O to the MPI-IO layer
    - E.g., buffer sizes, MPI-IO concurrency, contiguity, etc.
- Set data transfer mode
  - H5Pset_dxpl_mpio(hid_t dxpl_id, H5FD_mpio_xfer_t xfer_mode)
  - *H5FD_mpio_xfer_t*
    - H5FD_MPIO_INDEPENDENT → Use independent I/O access (default).
    - H5FD_MPIO_COLLECTIVE → Use collective I/O access.

Standard HDF5 “Skeleton”



Example of a PHDF5 C Program

A parallel HDF5 program has a few extra calls

```
…

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, …);
…
```

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();
```

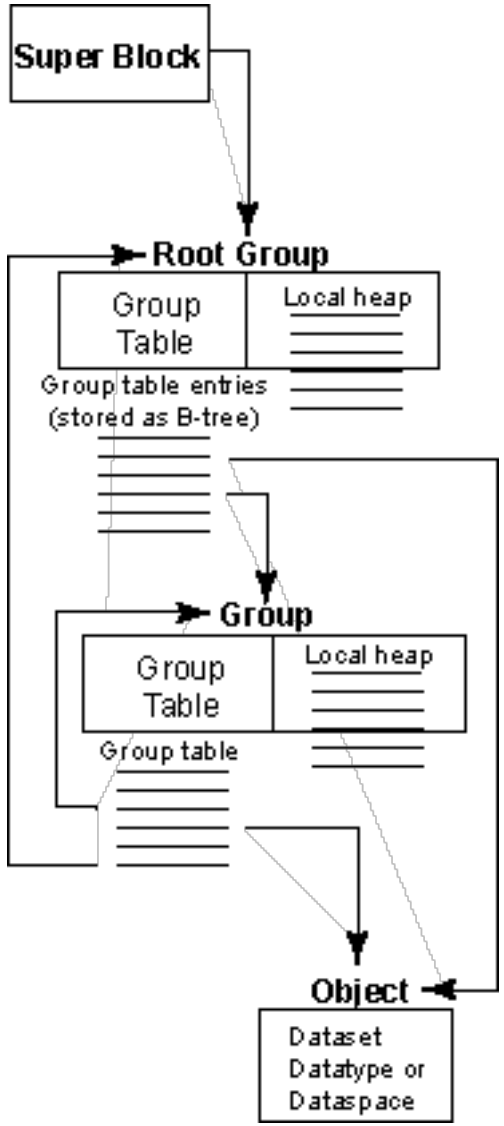
Parallel HDF5 – reading resources

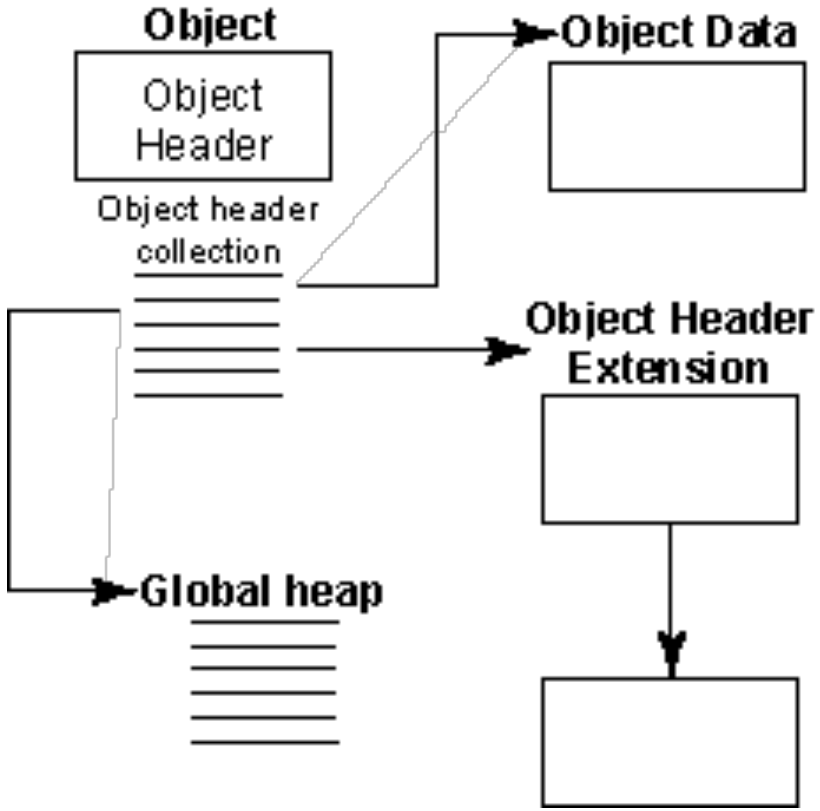
- Blog posts
  - https://www.hdfgroup.org/2015/04/parallel-io-why-how-and-where-to-hdf5/
  - https://www.hdfgroup.org/2015/08/parallel-io-with-hdf5
- For examples how to write different data patterns see:

http://support.hdfgroup.org/HDF5/Tutor/parallel.html

HDF5 file format

- Current specification is version 3
- HDF5 file is made up of
  - A superblock
  - B-tree nodes
  - Heap blocks
  - Object headers
  - Object data
  - Free space





https://docs.hdfgroup.org/hdf5/develop/_f_m_t3.html

Superblock

The superblock is composed of the format signature, followed by a superblock version number and information that is specific to each version of the superblock.

Layout: Superblock (Versions 2 and 3)

byte	byte	byte	byte
Format Signature *(8 bytes)*			
Version # of Superblock	Size of Offsets	Size of Lengths	File Consistency Flags
Base Address^{O}			
Superblock Extension Address^{O}			
End of File Address^{O}			
Root Group Object Header Address^{O}			
Superblock Checksum			

(Items marked with an 'O' in the above table are of the size specified in the Size of Offsets field in the superblock.)

B-trees and B-tree nodes

- Allow flexible storage for objects which tend to grow in ways that cause the object to be stored non-contiguously

Layout: Version 2 B-tree Header

byte	byte	byte	byte
Signature			
Version	Type	*This space inserted only to align table nicely*	
Node Size			
Record Size		Depth	
Split Percent	Merge Percent	*This space inserted only to align table nicely*	
Root Node Address^{O}			
Number of Records in Root Node		*This space inserted only to align table nicely*	
Total Number of Records in B-tree^{L}			
Checksum			

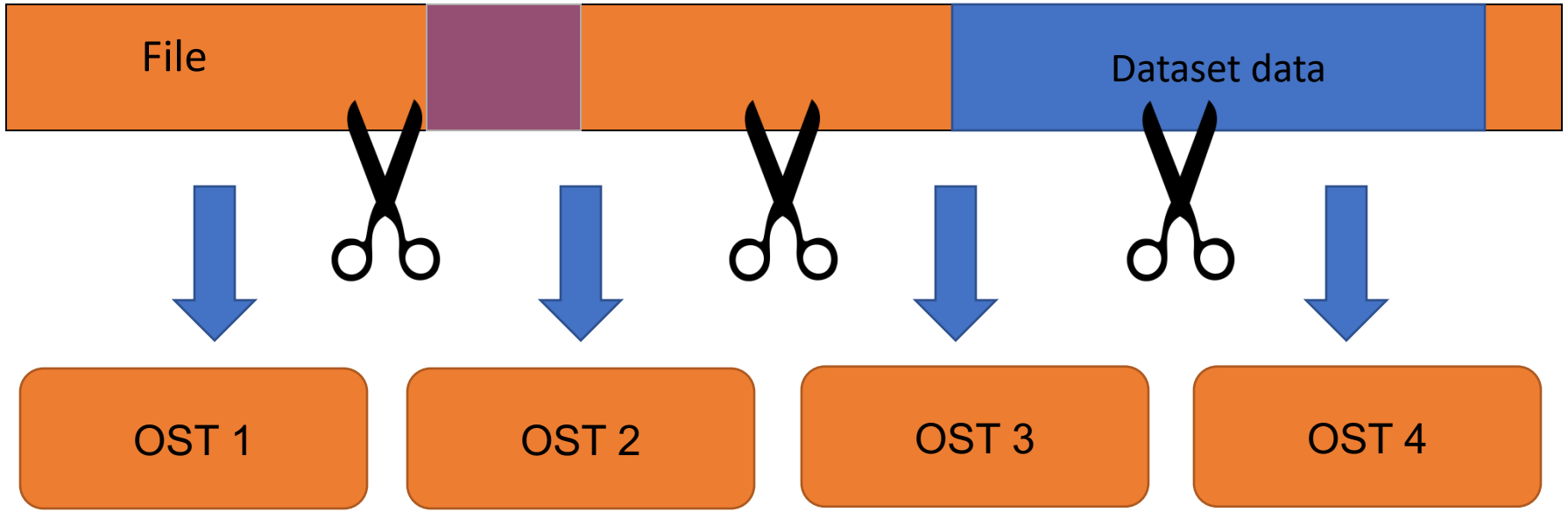
https://docs.hdfgroup.org/hdf5/develop/_f_m_t3.html

Other file format blocks

Layout: Version 2 B-tree Internal Node

byte	byte	byte	byte
Signature			
Version	Type	Records 0, 1, 2...N-1 *(variable size)*	
Child Node Pointer 0^O			
Number of Records N_0 for Child Node 0 *(variable size)*			
Total Number of Records for Child Node 0 *(optional, variable size)*			
Child Node Pointer 1^O			
Number of Records N_1 for Child Node 1 *(variable size)*			
Total Number of Records for Child Node 1 *(optional, variable size)*			
...			
Child Node Pointer N^O			
Number of Records N_n for Child Node N *(variable size)*			
Total Number of Records for Child Node N *(optional, variable size)*			
Checksum			

In a Parallel File System

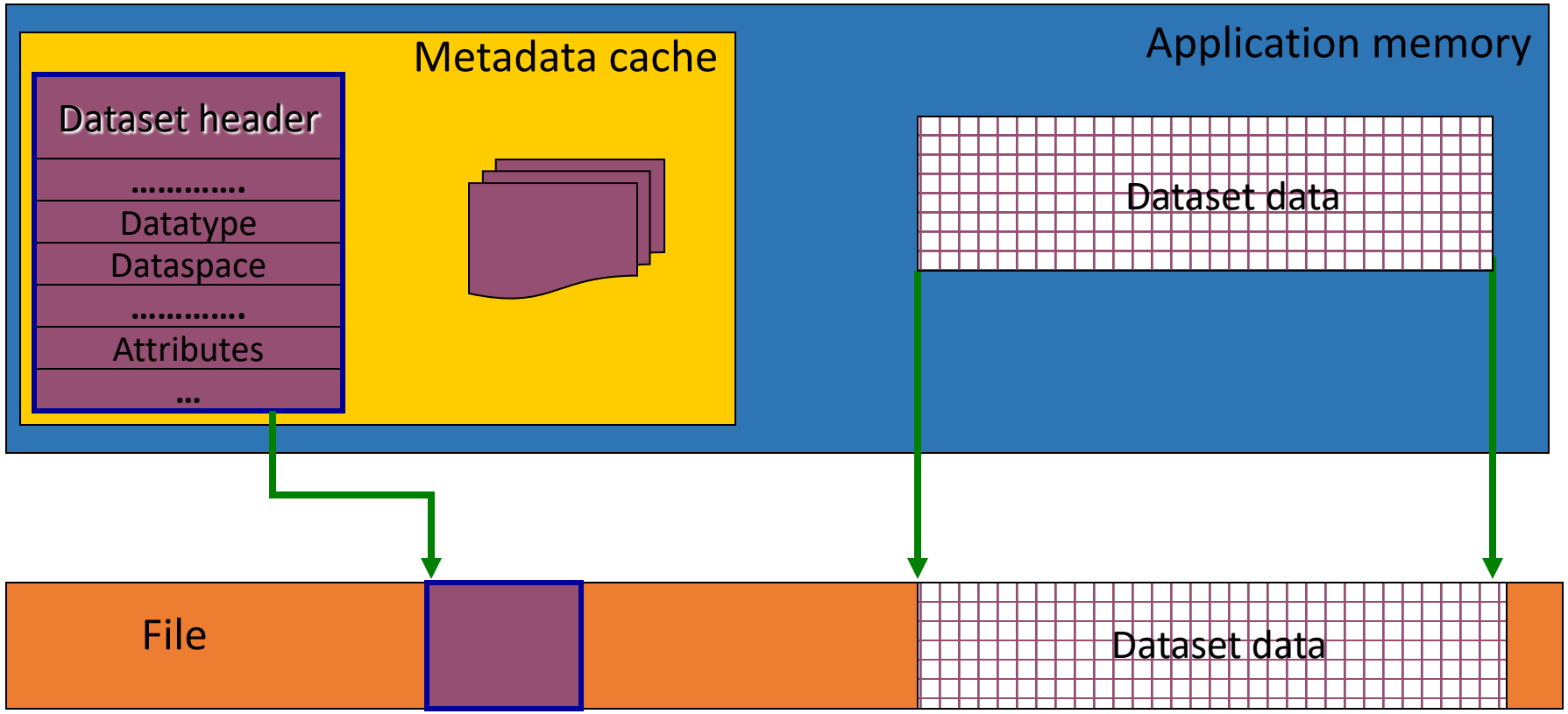


The file is striped over multiple “disks” (e.g., Lustre OSTs) depending on the stripe size and stripe count with which the file was created.

And it gets worse before it gets better…

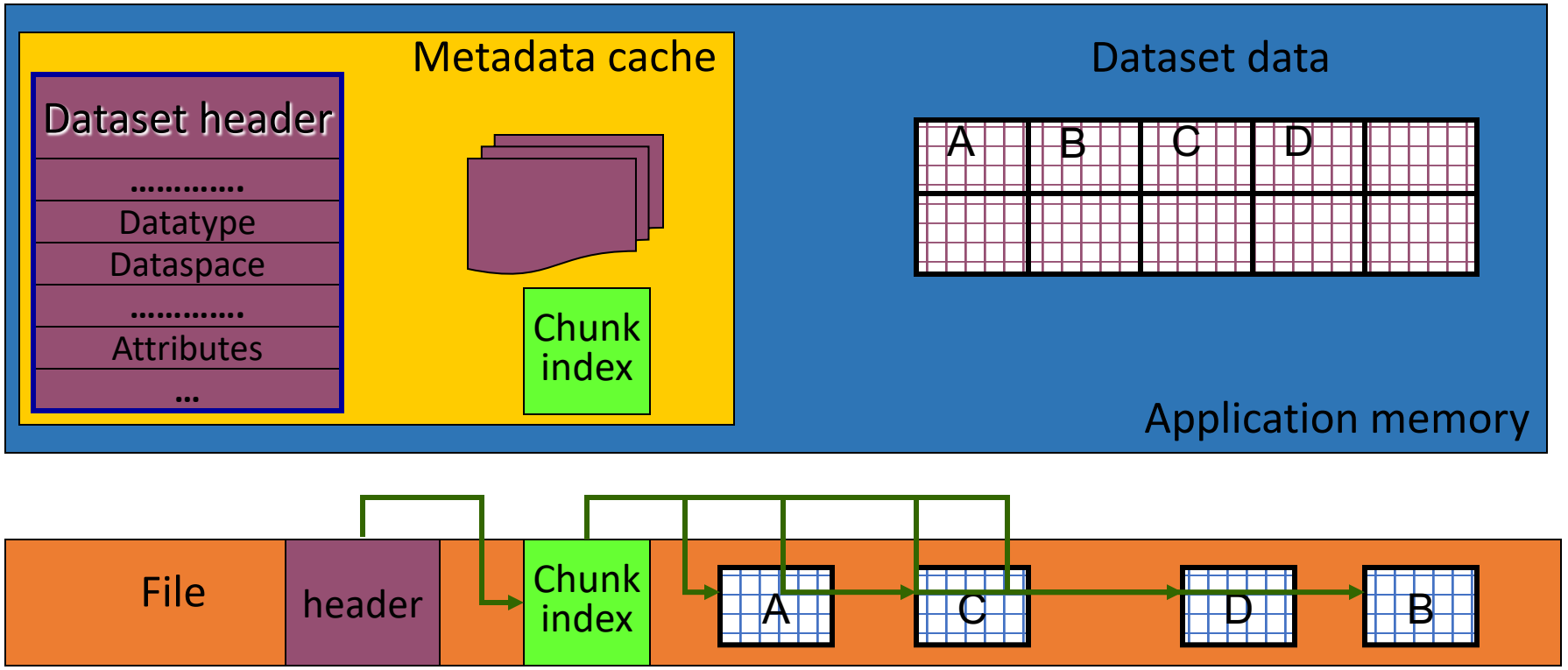
Contiguous Storage

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

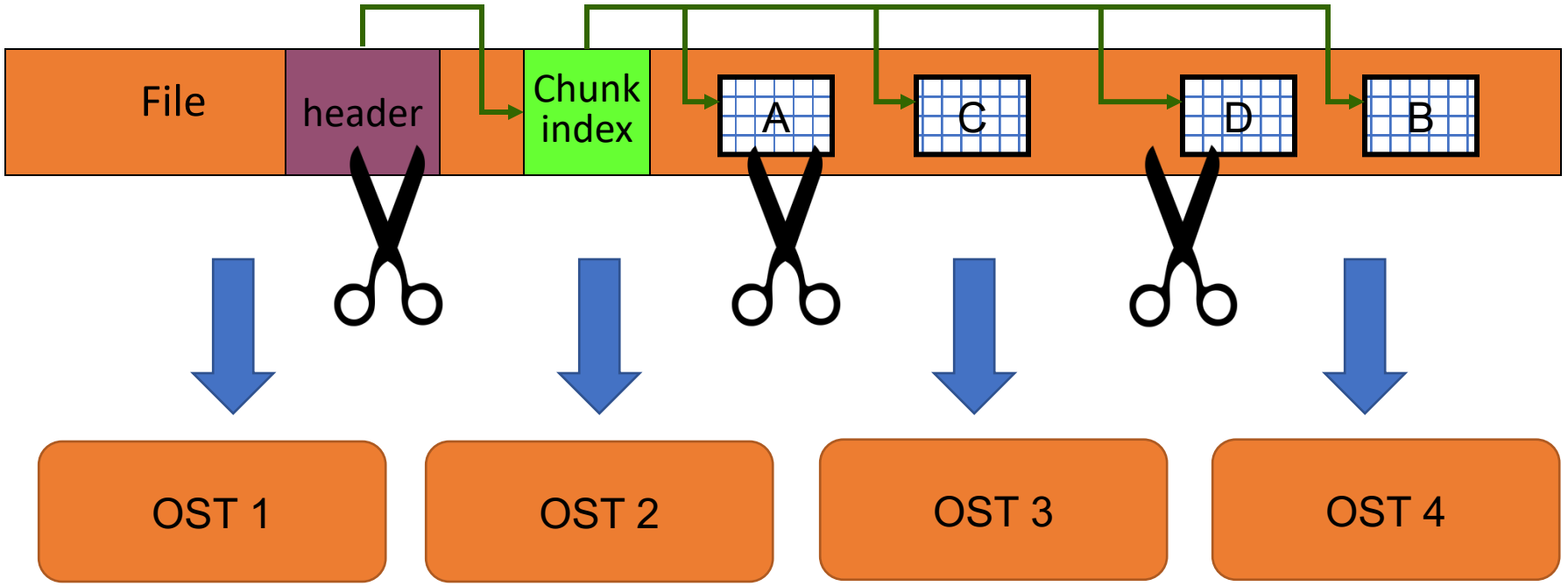


Chunked Storage

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



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.

Homework

- Run h5bench (write and read benchmarks) on a supercomputing system
  - https://github.com/hpc-io/h5bench
  - https://h5bench.readthedocs.io/en/latest/index.html
- Install PnetCDF and run examples

Summary of today's class

- Class project
- Parallel HDF5 and HDF5 internals
- Next Class –
  - Brief intro to PnetCDF and ADIOS
  - Parallel I/O performance topics