
October 29-31, 2024
ALCF Hands-on HPC Workshop

The HDF5 Interface and File Format

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

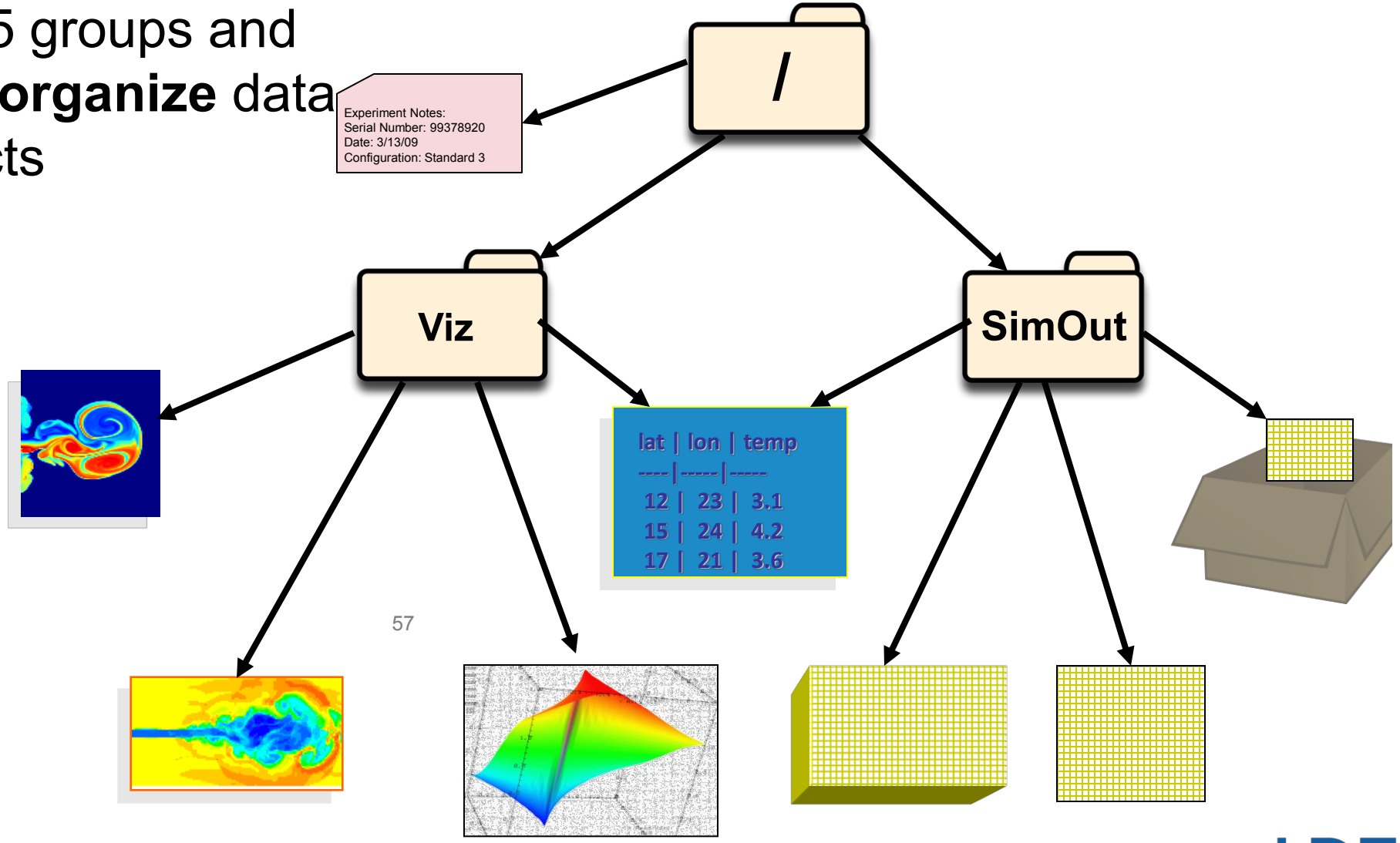
HDF5

- Hierarchical Data Format, from The HDF Group (formerly of NCSA)
  - https://www.hdfgroup.org/
- Data Model:
  - Hierarchical data organization in single file
  - Typed, multidimensional array storage
  - Attributes on any HDF5 "object" (dataset, data, groups)
- Features:
  - C, C++, Fortran, Java (JNI) interfaces
    - Community-supported Python, Lua, R
  - Portable data format
  - Optional compression (even in parallel I/O mode)
  - Chunking: efficient row or column oriented access
  - Noncontiguous I/O (memory and file) with hyperslabs
- Parallel HDF5 tutorial:
  - https://portal.hdfgroup.org/display/HDF5/Introduction+to+Parallel+HDF5

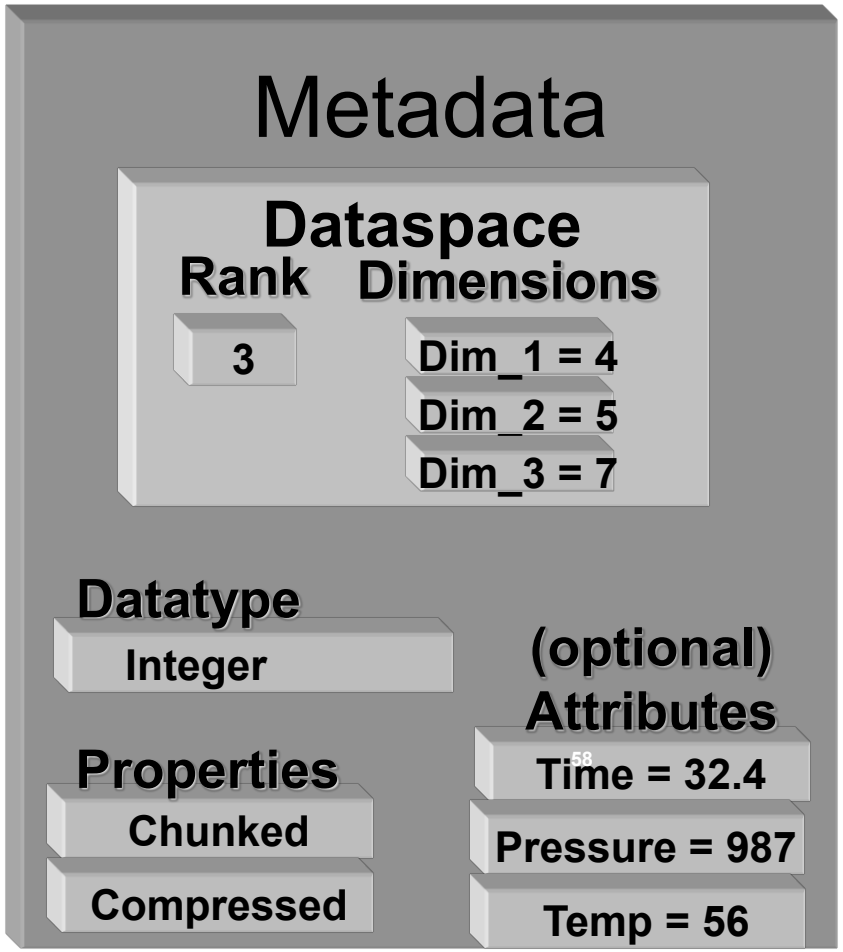
code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

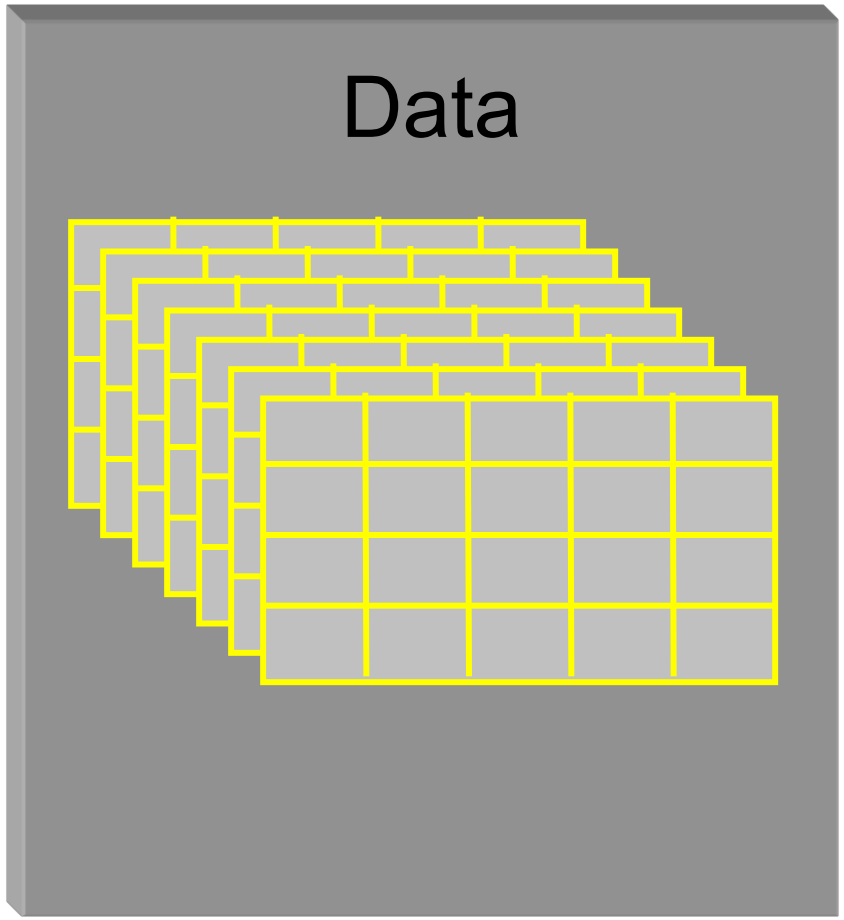
HDF5 Groups and Links

HDF5 groups and links **organize** data objects

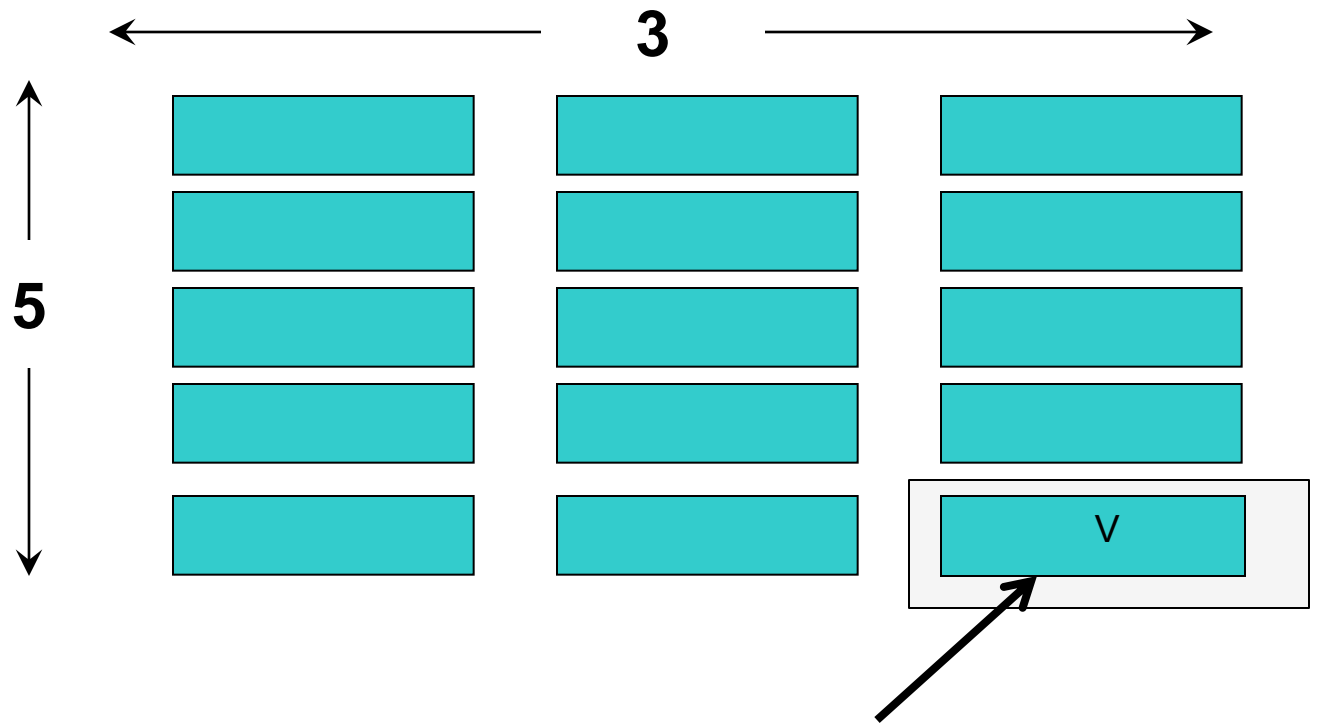


HDF5 Dataset





HDF5 Dataset



Datatype: 16-byte integer

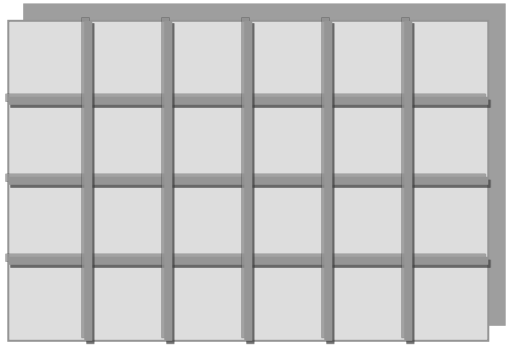
Dataspace: Rank = 2
Dimensions = 5 x 3

HDF5 Dataspaces

Two roles:

Dataspace contains spatial information (logical layout) about a dataset stored in a file

- Rank and dimensions
- Permanent part of dataset definition



Rank = 2

Dimensions = 4x6

Subsets: Dataspace describes application's data buffer and data elements participating in I/O

Rank = 1

Dimension = 10

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

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**S**select_hyperslab	*select subsections of data*
H5**D**read, H5**D**write	*access Dataset*
H5**D**close	*close Dataset*
H5**S**close	*close dataSpace*
H5**F**close	*close File*

NOTE: Order not strictly specified

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

“Hello World” HDF5 style

Cannot fit all in one slide: here are some highlights (see ‘hello-hdf5.c’ for full example)

```
file = H5Fcreate(argv[1], H5F_ACC_TRUNC, H5P_DEFAULT,
file_access_property_list);
```

- “property lists” used a lot in HDF5 (see next slide)
- Serial interface came first, with parallel features added later

```
/* in this simple example everyone writes their string to a
 1-d dataset; HDF5 supports variable length arrays ("ragged
arrays") but these datatypes have odd interactions with parallel
i/o */

    /* like writing to a plain file, we'll create one big
variable and everyone can write their string to the right (non-
overlapping) place in the file */
    hid_t dataset, datatpye, file_space;
    hsize_t size=varlen;

    file_space = H5Screate_simple(1, &size, NULL);
/* remember we got 'offset' from the MPI_Exscan above */
    hsize_t start=offset, count=len;
    status = H5Sselect_hyperslab(file_space, H5S_SELECT_SET,
&start, NULL, &count, NULL);
```

- Lots of flexibility in how memory, file regions described
- Lots more we could say about “hyperslab”

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

HDF5 example: opening with MPI-IO

```
/* Initialize MPI */
MPI_Init(&argc, &argv);
…
/* Create an HDF5 file access property list */
fapl_id = H5Pcreate (H5P_FILE_ACCESS);

/* Set file access property list to use the MPI-IO file driver */
ret = H5Pset_fapl_mpio(fapl_id, MPI_COMM_WORLD, MPI_INFO_NULL);

/* Create the file collectively */
file_id = H5Fcreate(argv[1], H5F_ACC_TRUNC, H5P_DEFAULT, fapl_id);

/* Release file access property list */
ret = H5Pclose(fapl_id);
```

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

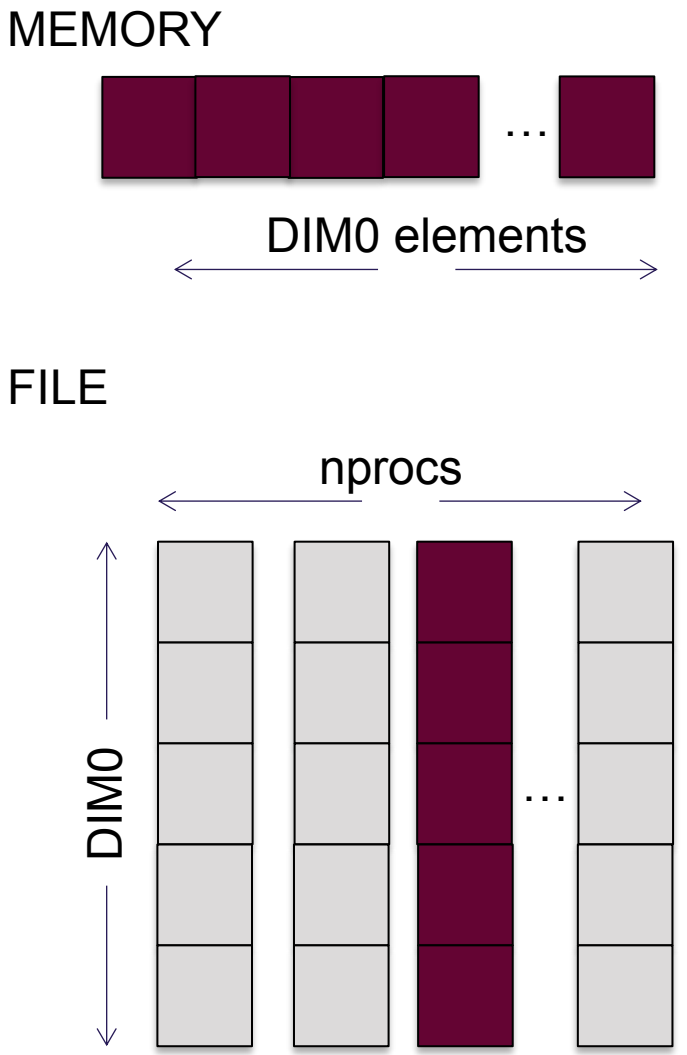
HDF5 example: setting up data transfer

```
/* Select column of elements in the file dataset */
file_start[0] = 0;          file_start[1] = mpi_rank;
file_count[0] = DIM0;       file_count[1] = 1;
ret = H5Sselect_hyperslab(file_space_id, H5S_SELECT_SET,
        file_start, NULL, file_count, NULL);

mem_start[0] = 0;           mem_count[0] = DIM0;
ret = H5Sselect_hyperslab(mem_space_id, H5S_SELECT_SET,
        mem_start, NULL, mem_count, NULL);

/* Set up the collective transfer properties list */
dxpl_id = H5Pcreate(H5P_DATASET_XFER);
ret = H5Pset_dxpl_mpio(dxpl_id, H5FD_MPIO_COLLECTIVE);

/* Write data (one column of doubles) collectively */
ret = H5Dwrite(dset_id, H5T_NATIVE_DOUBLE, mem_space_id,
        file_space_id, dxpl_id, write_buf);
```



code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

Effect of HDF5 Tuning

- HDF5 property lists can have big impact on internal operations
- Collective I/O vs. Independent I/O
  - Huge reduction in operation count
  - Implies all processes hit I/O at same time
- Collective metadata (new in 1.10.2)
  - Further reduction in op count, especially reads (reading HDF5 internal layout information)
  - Big implications for performance at scale

Operation counts	Independent	Coll. I/O	Coll. MD
POSIX Write	3680007	9	9
MPI-IO Indep write	3680007	7	0
MPI IO Collective Write	0	16	48
POSIX Read	3680113	115	10
MPI-IO indep read	3680113	113	8
MPI-IO collective read	0	16	16

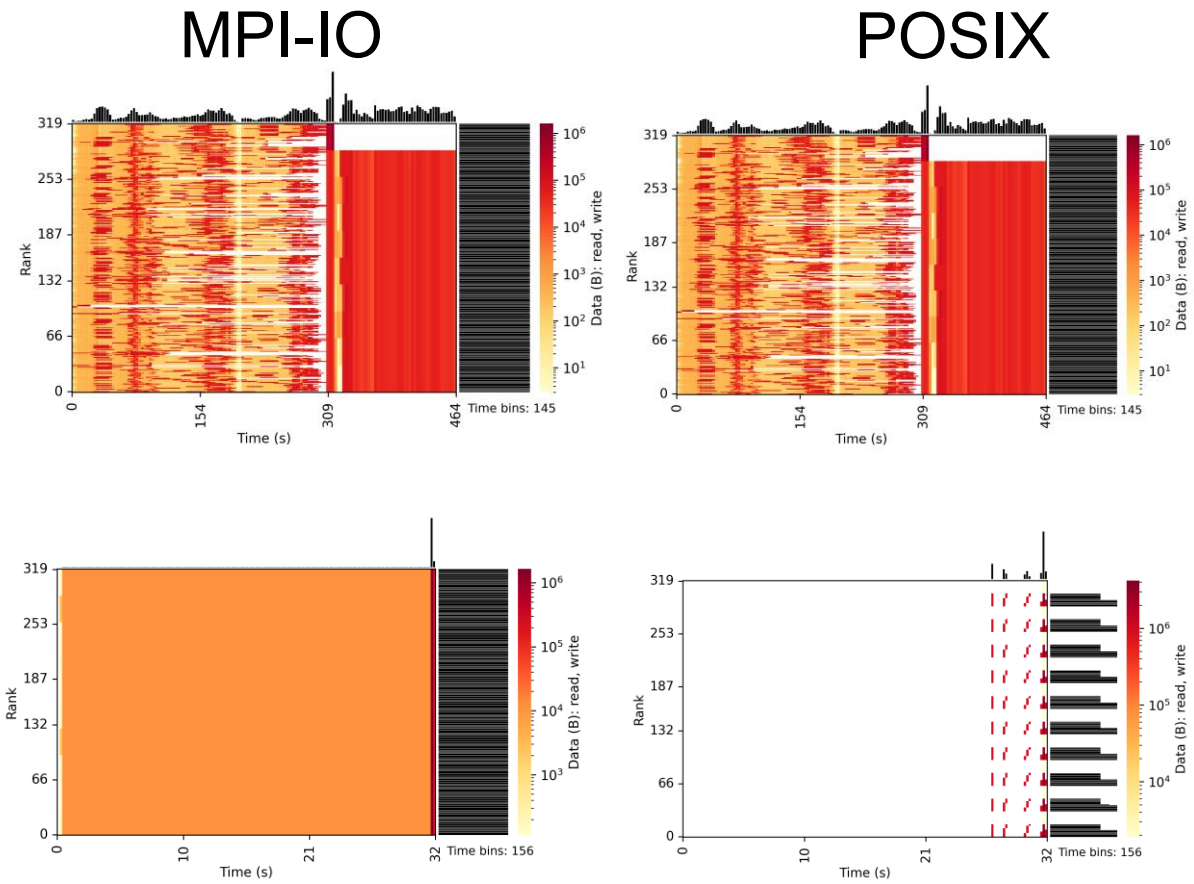
Selected Darshan statistics for 16 MPI processes writing 230 K doubles to HDF dataset, reading back same.
visualization_io/mpiio-hdf5/hands-on/hdf5/h5par-comparison.c

Effect of HDF5 Tuning

- HDF5 property lists can have big impact on internal operations

- Collective I/O vs. Independent I/O
  - Huge reduction in operation count
  - Implies all processes hit I/O at same time

- Collective metadata (new in 1.10.2)
  - Further reduction in op count, especially reads (reading HDF5 internal layout information)
  - Big implications for performance at scale



visualization_io/mpiio-hdf5/io-sleuthing/examples/hdf5

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

HDF5 in other languages

- Python:
  - H5py: http://www.h5py.org/
    - closely coupled with mpi4py and numpy;
    - some collective tuning not exposed at python level
- C++:
  - Highfive: https://github.com/BlueBrain/HighFive
    - header-only interface to HDF5 C API

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

New HDF5 features:

- New in HDF5-1.14.0
  - Async operations
    - Potential for background progress
  - Multi-dataset I/O
    - Similar to pnetcdf “operation combining”

Data Model I/O libraries

- Parallel-NetCDF: http://www.mcs.anl.gov/pnetcdf
- HDF5: http://www.hdfgroup.org/HDF5/
- NetCDF-4: http://www.unidata.ucar.edu/software/netcdf/netcdf-4/
  - netCDF API with HDF5 back-end
- ADIOS: http://adiosapi.org
  - Configurable (xml) I/O approaches
- SILO: https://wci.llnl.gov/codes/silo/
  - A mesh and field library on top of HDF5 (and others)
- H5part: http://vis.lbl.gov/Research/AcceleratorSAPP/
  - simplified HDF5 API for particle simulations
- GIO: https://svn.pnl.gov/gcrm
  - Targeting geodesic grids as part of GCRM
- PIO:
  - climate-oriented I/O library; supports raw binary, parallel-netcdf, or serial-netcdf (from master)
- … Many more: consider existing libs before deciding to make your own.
- Note absence of a “machine learning” library – research opportunity for someone!

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop

Wrap-up

- Lots of activity, history making I/O better… Still a lot to do!
  - Workflow, task-oriented, AI/ML
- ALCF consultants, research community eager to help

code etc: https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop