Parallel I/O with HDF5
(the slides are available under “Presentation Materials” in the above URL)
Date: March 13, 2019
Presented by: Quincey Koziol (LBNL/NERSC)

In addition to Quincey Koziol’s answers during the webinar, Elena Pourmal, Scot Breitenfeld, and Suren Byna contributed to the answers.

Q. Is there a more introductory material you can recommend for somebody starting in parallel I/O ?

A. There are several tutorials online. A few links below.
   a. This is a good starting point: https://portal.hdfgroup.org/display/HDF5/Introduction+to+Parallel+HDF5
   b. A blog post: https://www.hdfgroup.org/2015/08/parallel-io-with-hdf5/
   c. Links to training materials on HDF5: https://sdm.lbl.gov/exahdf5/training.html

Q. Will there ever be HDF6 and will it be compatible with HDF5?

A. We don’t anticipate moving to an HDF6. We upgrade the format periodically but we want to access every HDF5 file written, ever.

Q. Out of curiosity, are there alternative implementations of HDF5 specifications?

A. Yes - Unidata has pure Java (limited) reader directly from HDF5 file, along with [another] Java package that only creates / writes HDF5 files. Both of these directly parse / write the file format, without using the HDF5 C library.

Q. (Slide 13) What’s the upside of specifying a max size instead of leaving data spaces as unlimited? What optimization does that buy you?

A. When you fix the dimensionality of an array you can store elements more efficiently on disk. Can do at once. Unlimited requires you to break it up into chunks to be extended easily. Otherwise you need to read in the whole dataset to extend its size, which would be terrible. Slight performance penalty for

unlimited. Will talk more about chunking later on. Straight-forward applications should use fixed dimensions for writing all at once.

Q. (Slide 15): Is there built-in support for complex numbers?

A. We've talked about it and the C standards cover complex types. We have discussed implementing it, but have not standardized it yet. Several community approaches for various science domains, but we are lacking a standard one currently.

Q. (slide 21) Why can't you link parts of datasets? Like one column in one dataset to another row in a different dataset.

A. Could create a dataset like longitude which was just a column that could be shared between datagroups to indicate they all access that component of it.

Q. Is there a more introductory material you can recommend for somebody starting in parallel I/O ?

A. There are several tutorials online. A few links below.
   a. This is a good starting point: https://portal.hdfgroup.org/display/HDF5/Introduction+to+Parallel+HDF5
   b. A blog post: https://www.hdfgroup.org/2015/08/parallel-io-with-hdf5/
   c. Links to training materials on HDF5: https://sdm.lbl.gov/exahdf5/training.html

Q. How is HDF5 development and maintenance funded and do you expect any changes/challenges to future funding?

A: HDF5 is maintained and developed by the The HDF Group. We get funding from DOE, NASA, commercial entities. The HDF Group is part of ECP project and maintenance and some feature development like sub-filing, query indexing, multiple performance improvements are funded by ECP.

Q. (Slide 25) Do you support reviewing bindings for other programming languages?

A: No, The HDF Group accepts patches for bindings THG supports - Java and Fortran and C++. No one contacted us with such request, but we do review code if people ask for help.

Q. Are there Julia wrappers?

A. Yes there are Julia wrappers. Might be this: https://github.com/JuliaIO/HDF5.jl

Q. For writing lots of unit tests can one mock HDF5 files and objects without writing to disk?

A: Yes. Use core driver see https://portal.hdfgroup.org/display/HDF5/H5P_SET_FAPL_CORE

Q. Is a parallel filesystem required for parallel I/O of files?

A. Strongly recommended. Debugging story time! MPICH supports accessing files on NFS through MPI. You can write what you think is parallel I/O to NFS mounted file, but very simple test that takes a few fractions of a second on a real parallel filesystem, but on NFS was about 2 minutes. About 1000x worse performance as a cautionary tale <3

Q. When using FORTRAN interface, the array dimensions are transposed (related to memory storage). Is there additional arguments to change that at read time / write time or when creating the dataspace ?

A. This question often comes up. We have great FORTRAN people on the team. Do what makes sense to you in FORTRAN and C and the library should do the right thing. It should be OK.

Q. In my understanding, the number of fields in a compound datatype is limited by the Object Header size. Is there any workaround for this?

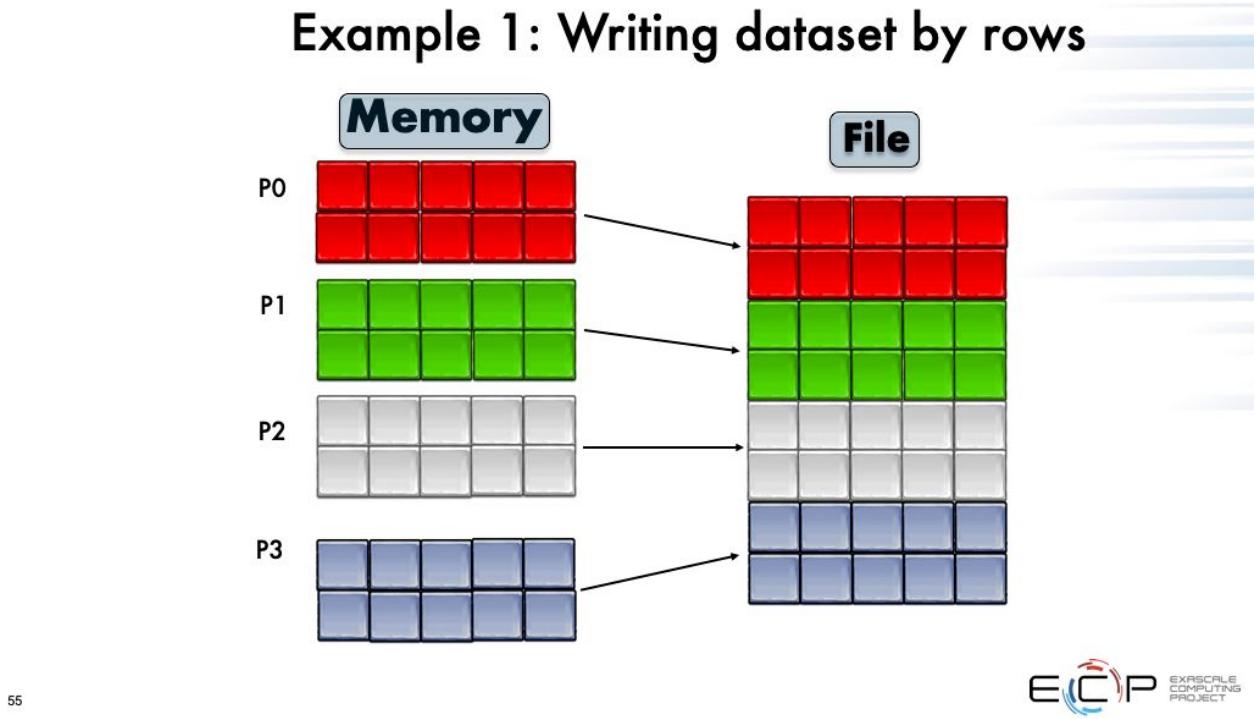
A: Very knowledgeable question! Technically yes, this is a limit. Would be great to expand the size. Not technically hard, but requires a few months of manpower and funding. No explicit mandate to do that now. Can advise you on how to do that; this is an open source project. <3

Q. Are parallel HDF5 features accessible via bindings other than C? E.g., C++ or Python?

A: We (The HDF Group) support Fortran bindings. H5py community supports Python bindings. People use C++ applications and call C parallel library, there is no support for parallel in HDF5 C++ bindings. You may take a look at H5CPP project https://portal.hdfgroup.org/display/HDF5/HDF5+Language+Bindings. Also, HDF5 C++ webinar: https://www.hdfgroup.org/2019/01/hdf5-c-webinar-followup/

Q. (Slide 42): Do all ranks specify the same data set size, i.e. the global size of the dataset?

A: All process in communicator should participate in a dataset creation, so yes, dataspace will be the same. You can think of the global size as the size of the dataset on disk, you can think of this in HDF5 speak as the file_space.



Q. (follow up to above) Ok, so then the different process write to different slices or hyperslabs of the shared (global) dataspace?

A: Each process has a part of the dataset. This is termed in HDF the “memory space”. You then select the hyperslab in the filespace to where in the file space each process should do I/O to and from disk.

Q. Are there any plans to reduce restrictions of the SWMR mode?

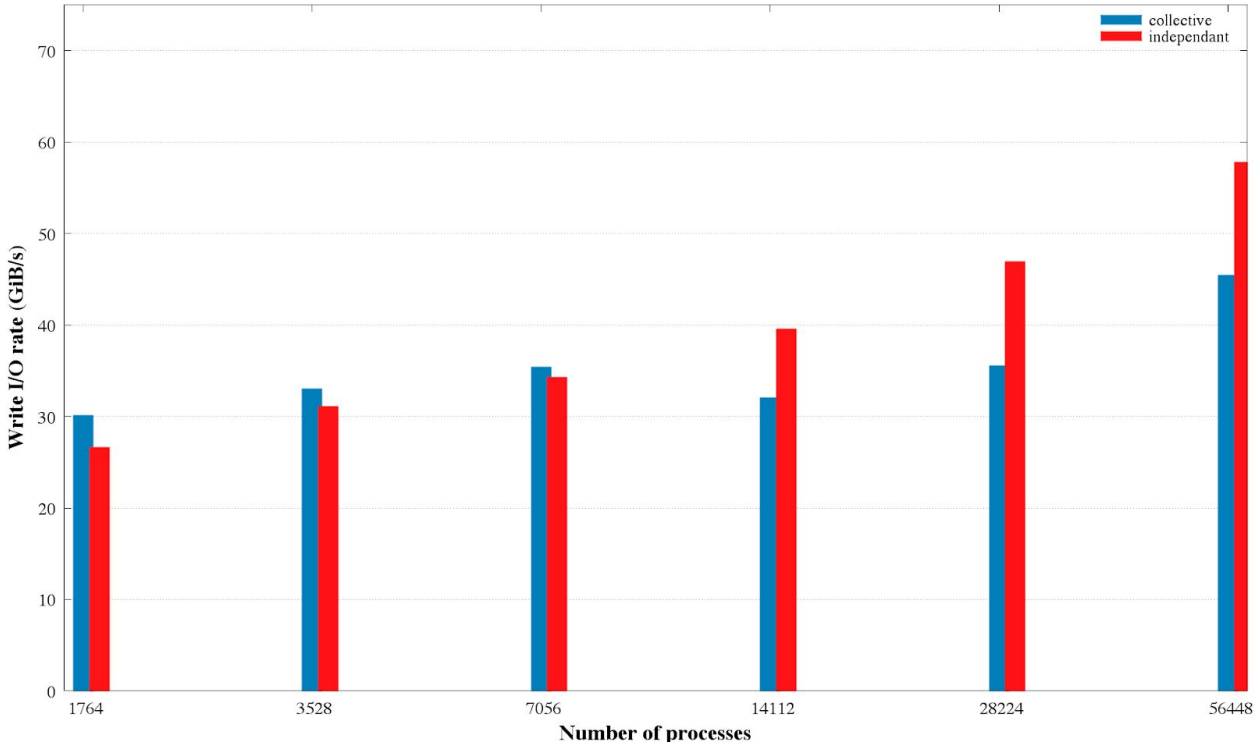
A: Yes, under ECP project Quincey implemented full SWMR (i.e., one can create objects and add attributes while file is under construction). The HDF Group under another project prototyped a solution that doesn't depend on POSIX semantics.

Q. Will the SWMR mode be supported on NFS file systems?

A: It's not likely with the ECP funded version, as the NFS I/O semantics don't seem to allow it. The alternate implementation that the HDF Group is working on may allow this though.

Q. Any experience on GPFS? We did some tests and collective is better than independent in some cases. The independent mode was better for Lustre and Collective for GPFS but for sure it is not the rule.

A. We test on both GPFS and Lustre. It depends on your IO pattern. On summit at ORNL with VPIC we see a transition from better collective to better independent, depending on the number of processes.



Q. Does the permission to use non-collective calls to write to datasets extend to datasets that are chunked/compressed/extensible?

A: If dataset is compressed writes must be collective.

Q. Is there an advantage to using VOL versus a direct file converter for NetCDF?

A: VOL will allow one to use HDF5 APIs to directly read from netCDF file. It really depends on what your goals are.

Q. Does interoperability with NetCDF4 include writing those files from HDF5 library?

A. What does interoperability mean to you? One can create netCDF4 file just using HDF5 APIs.
    a. I was thinking of the features in netcdf not in hdf. Is there some combination or "takeover" of netcdf capabilities? I'm less familiar with netcdf.
    b. Ah, I see… you are talking about netcdf-3 format, right?
    c. Not sure, new to both. I have seen more unit/dimensions in netcdf.

PLease contact me at epourmal@hdfgroup.org to continue the conversation! This is a very good question :-)

Q. Will the SWMR mode be supported on NFS file systems?

A: Yes, with a different implementation of SWMR that The HDF Group is working on.

Q. Hi Ashley, I was wondering about planned performance enhancements for support of parallel IO without MPI IO

A: I'm not certain what this is referring to, but possibly the hyperslab improvements that are on the way? Those will be included as part of the upcoming 1.12.0 release.

Q. Are there any plans to reduce restrictions of the SWMR mode?

A. Full SWMR feature is being developed in the ECP ExaHDF5 project. (See slide # 115 in the presentation @ https://www.exascaleproject.org/wp-content/uploads/2019/03/2019-03-13-ECP-HDF5-Webinar.pdf)

Correction:

1.10.5 was just released
1.10.6 will be in November
New major release 1.12.0 or 2.0 is coming in May-June
We recommend to use supported HDF5 portal
https://portal.hdfgroup.org/display/HDF5/HDF5

Shared notes:
Speaker = Quincey Koziol

- We hear “you should go use HDF5”, but what is it?
    - Like XML = self describing. Markup and annotate your data.
    - High performance, compact as binary flat file like Fortran write statement.
    - Like “PDF of science”, standard exchange format, long term, archival. Can contain lots of kinds of data like text and images in PDFs.
    - Hierarchy is similar to file system for logical grouping for your applications.
    - Random access and subsetting similar to database.
- 3 levels of HDF5:
    - File format
    - Most people work with the software, usually through middleware.
    - The data model. What concepts do I need to acquire to put data in HDF5 files?
- Data model:
    - Many object concepts:
        - Dataset and Attribute are the primary objects
        - Library is written in C, so not real objects like C++
    - Putting “data” means putting a dataset.
    - Every dataset is an ND array of identically type (floats, ints, structs, strings)
    - Metadata that describes it
    - Dataspace
        - Can technically have empty / NULL attributes with no elements.
        - Scalar
        - Array (most common) - can specify max size or unlimited
    - Selection is a wrapper around dataspace.
    - Datatypes describe elements in the homogenous elements of a dataset:
        - Integer, Float, Enum, … and more complex types e.g. 17-bit floating type if you really need it.
    - (Slide 18) Showed example of complex datatype)

- Advantages of chunking
    - Accessing data is much faster when subsetted.
    - Chunking required for compression
    - Faster when sending over network
- Can store external datasets in other HDF5 files for virtual datasets. Such as single view using many cameras.
- To organize HDF5 objects this we use HDF5 Groups and Links
    - (Slide 21) Hard links to shared dataset
- Attributes used for metadata e.g. parameter settings. Similar to key-values. Unique attribute for a given object. Don't support partial I/O or compression. If you have something large, use a dataset instead.
- See HDF5 home page for more resources that we can't cover today.
- 1.10.6 is latest. Main library in C. Great bindings for python.
- Wrapper scripts h5cc, h5++, h5fc are wrapper scripts that link application together, etc.
- Programming API overview:
    - C API routines begin with h5 prefix + 1 letter prefix. H5d = dataset, h5f = file, etc.
    - Rivals large APIs like MPI. Hundreds of routines.
    - Very C oriented paradigms. Do a lot yourself. Create object, access it, destroy it.
    - Create dataspaces instead of datatypes because the HDF5 library has no idea what you mean by an array: is it 2 dimensional, etc?
    - Select subsets with hyperslab and elements for partial I/O
- Parallel HDF5 (PHDF5)
    - Designed to be the same as serial and uses serial interfaces.
    - Uses MPI-IO under the cover.
    - Writes same HDF5 files conforming to the HDF5 spec.
    - (Slide 44) More restrictions on operations
    - (Slide 42) You must specify Collective IO to disable independent IO!
    - Lustre striping factor should be maximum for the system. Much larger than 1, otherwise only 1 server manages access to the file.
- What does collective mean?
    - Combines individual operations.
    - Important use cases where individual is better.
    - Generally follows MPI convention.
    - Collective is not necessarily synchronous and doesn't necessarily require MPI communication. Might only change state of file.
- (Slide 46) See Parallel IO tutorial

- Diagnostics for high performance HDF5
    - Structure metadata ito bundles so that you don't have millions of attributes. Use latest library version of HDF5 for better optimization of file format.
    - We do lots of metadata caching.
    - (Slide 54-60) Skipped for time
    - h5perf command-line tool. Darshan also useful from Argonne National Labs. Tells you the I/O pattern in use. Measures serial I/O vs parallel I/O
- HDF5 tuning
    - (Slide 91) Sometimes you want reads to be collective to hundreds of processes reading lots of metadata. Don't DDOS your filesystem! (Slide 92) Set all collctive metadata ops property.
- (Slide 99) ECP project features
- Some feature this year
    - Virtual Object Layer (VOL) common data model and API, but new plugins for new features in file systems. Or for legacy features. Available in git repo and will be available in 1.12 over the summer.
    - Database style indexing and querying. Queries across many files, e.g. Locations in datasets that meet some criteria. Then look at results with selecting I/Os. We want to avoid requerying and accelerate using indexes. We're implementing pluggable architecture that allows us to use those things.
- Support and maintenance with docs, answers.
    - Start at helpdesk help@hdfgroup.org
    - Or post on furm https://forum.hdfgroup.org
- New topology aware features, node local storage, async I/O for metadata and dataset operations