Sandia National Laboratories

Exceptional service in the national interest

REBASING FOR THE WIN!

Why HDF5 should break with the past

Jay Lofstead (gflofst@sandia.gov)

Sandia National Laboratories

HDF User Group 2024

2024-08-05

SAND2024-10106C

MY HISTORY WITH HDF5

- Developed ADIOS 1.0 for thesis work (2007-2010)
- Motivation was a reaction to HDF5
  - Too many API calls
  - Too slow
  - Not performance portable
- Two-phase collective IO with data sieving not scaling well with large, 3D data in particular
  - Yu W, Vetter J. Parcoll: Partitioned collective i/o on the cray xt. In2008 37th International Conference on Parallel Processing 2008 Sep 9 (pp. 562-569). IEEE.
  - Lofstead J, Oldfield R, Kordenbrock T, Reiss C. Extending scalability of collective io through nessie and staging. In Proceedings of the sixth workshop on Parallel Data Storage 2011 Nov 13 (pp. 7-12).
  - Exodus file format gyrations; Chombo gyrations

ADIOS 1.0

- Independent IO in blocks (log-based format)
- Pluggable transport layer WITHOUT changing source code
- No special hints needed
- Indexed metadata and data

Problems:

1. Block sizes varied requiring duplicating a lot of metadata
2. Indexed metadata still a linear search, but compact representation worked for a long time
3. Moving to tape and back to storage could change the file layout removing performance advantages (e.g., striping configuration for directory/files lost when restored from tape)
4. No optimizations based on data size (small data should be handled differently)
5. Adaptive IO approach imbalanced storage system and only worked with Lustre
6. File system specific optimizations had to be written explicitly for each one

ECP, HDF, AND ADIOS

- HDF5 is the standard for ECP
- ADIOS was supported as well based on a few applications and the “2 solutions” idea
- HDF5 *API* is the winner
    - Rob Ross’ comment to me many years ago
- Virtual object layer re-implements API for many new purposes
    - Pass through or leaf-node versions to stack and stream VOL connectors
- HDF VOL Connector for ADIOS created
    - Re-implanting HDF5 API to talk with ADIOS strongly suggested in 2009 as marketing

- Net result, HDF API stronger than ever; ADIOS living alongside (NetCDF still for Climate, but weakening)

HDF5 IS THE STANDARD

- For HPC-related IO libraries, it is the baseline
- The API is ubiquitous in simulations and analysis tools
- Both parallel and serial APIs supported
- All IO innovations compared against HDF5
- HDF5 adds innovations to keep pace
  - Chunking
  - Using MPI hints to adjust
  - Other file format changes
  - Differing layout and performance tuning options available

HOW APPLICATION SCIENTISTS USE IO LIBRARIES

- Scientist hears from friend, “HDF5 hooks in with all of these analysis tools and has a Python interface as well. You should try it for your data management.”
- Scientist looks up HDF5 and sees the truth
- Scientist implements HDF5 in their code
- Scientist gets quicker access to a wider variety to tools and LOVES! LOVES! LOVES! It!
- Scientist adjusts their output frequency in accordance with what they can afford and HDF5 achieves—hopefully it is enough
  - If yes, done!
  - If no, might suffer because of the tool integration
  - Might ask for some help tuning

PROBLEM

- Nobody* uses advanced file features since you have to turn them on/use special APIs
  - MPI Hints offer excellent performance tuning options—if you turn them on
  - Chunking—if you turn it on
  - Parallel compression—if you turn it on
  - “Memory Leaks”—because scoping requires disciplined programmers
    - Dataspace objects managed independently than data, for good reason mostly
  - Explicit MPI related tuning calls—who does this?
  - Contiguous vs. compact storage are different
    - Should be automatic
  - Metadata caching vs. API to search metadata
  - Alignment calls to interact with file system
  - And more!

*Nobody = very, very few people.

PROBLEM, PART 2

- Needing tutorials for performance tuning
  - E.g., https://www.hdfgroup.org/wp-content/uploads/2020/06/2020-06-26-Parallel-HDF5-Performance-Tuning.pdf
- Needing to understand storage systems, detailed machine architecture, and IO patterns
- Needing to re-validate code after the IO tuning is done
- Extensive, specialized API calls for debugging and tuning are anything but friendly
  - Application scientist do not want to be IO experts. They are domain science experts.

- What should be tried with so many options?
- Will any of these special flags still work with my analysis workflow?
- Can I just set some file system flags and get it to work well enough?
- Can I just run on a different machine and avoid all of this mess?

THE RESULT

1. Forward compatibility requirements make innovations largely invisible/unused
2. MPI hints are poorly documented
3. How to turn on a variety of features can be difficult
   - Hari’s experiences trying to get help
4. HDF5 is used for compatibility rather than as a high-performance subsystem

- Why shortchange both HDF5 and the Community?

PROPOSAL

- Sandia's HDF5 installations go back to version 1.10. Make this the new baseline turning on all features introduced as of this release by default
  - Forward compatibility is strictly eliminated
- Develop ways to auto-tune or offer platform specific global tuning parameters
  - Sysadmins can run tests to generate system-specific settings for various data organizations
- Simplify the API by deprecating and quickly eliminating old calls rarely used or those that have been updated
  - Yes, this will cause some pain
  - Number of API calls is ENORMOUS with many obsolete
- Rewrite the base code strictly as VOL connectors making NO base version!
  - Sysadmin selects default for the platform
  - Make older HDF5 format compatibility a VOL connector

OUTCOMES

- Short term
  - Lots of worry
- Long term
  - People find HDF5 is widely supported/compatible, high performance, and good performance portability

Alternative

- Just stop innovating as it isn’t going to be used
  - You can show a scientist how to tune their IO, but you can’t make them do it