HDF5 and XRootD

Tom Junk

DUNE Computing Special 1-day precollaboration meeting

May 9, 2022

Context

- HDF5 is becoming more popular all the time on DUNE
  - Machine-learning workflows have been using it for years
  - DAQ group has started writing detector data using this format
    - Coldboxes (VD and HD)
    - ICEBERG
    - upcoming ProtoDUNEs (VD and HD)
- We expect a large amount of raw data to be stored (petabytes) in this format.
- Users will be making increasing amounts of HDF5 data for their own analyses
- MC will also need to be stored in this format (see Barnali's talk)

Context

- Streaming of data to grid jobs via XRootD helps us optimize use of resources, compared with file copying at job begin time.
  - Only transfers the data you need
    - Subset of trigger records
    - Subset of TBranches
  - Delays I/O from job start and spreads it out over the lifetime of the job.
    - Very noticeable when many jobs start at similar times
- More comfortable interactively: If you want to run an event display on your own computer, you get to see the first event right away, rather than having to copy the whole file to your desktop.
- XRootD is much preferred over PNFS access for interactive use of files in dCache
- We already put a lot of HDF5 files in dCache (and enstore)

HDF5 and Streaming

- HDF5 use by the HEP community is still rather new.
- We have unusual requirements (at least historically unusual)
  - large data sizes
  - embarrassingly parallel workflows
  - we use a lot of computers we don't own
- New development for streaming: HSDS: https://www.hdfgroup.org/solutions/highly-scalable-data-service-hsds/
- HSDS may be targeting the data exploration market space
  - data are stored in "shards" in an object store
  - May be similar in idea to PIAF or PROOF
  - Needs an object store: Use “AWS S3”, “Azure Blob”, “OpenIO”, or Posix storage
  - dCache is "none of the above"

XROOTD

- XRootD meets a lot of needs in the HEP community
  - network file access
  - we already have xrootd servers set up and supported
  - authentication mechanisms have been established and we know how to use them
- Tight integration with ROOT. Give ROOT a filename starting with "root://" and it will use XRootD
- More general than I had previously realized, though.
  - any kind of file
  - any kind of application

HDF5 Virtual File Layer

- I had originally worried that we would need to write a Virtual File I/O interface for HDF5 that calls XRootD methods. https://support.hdfgroup.org/HDF5/doc/TechNotes/VFL.html
- Examples of VFLs:
  - H5FDstdio.c (.h)
    - used as an example, not as well tested as the SEC2 driver.
    - Has direct calls to fopen(), fclose(), fseeko(), fread(), fwrite()
  - H5FDsec2.c (.h): I think this is the production driver
    - Uses I/O methods redefined in H5private.h
    - example line from H5private.h:

```
#define HDfopen(S,M)    fopen(S,M)
```

XRootD POSIX Interface

- The XRootD team sure made it easy to use! https://github.com/xrootd/xrootd/tree/master/src/XrdPosix
- See the README for instructions
- XrdPosix.hh is a nice convenience – redefines Posix stdio methods to XRootD versions:
- Example line: #define fopen(a,b) XrdPosix_Fopen(a,b)
- Even easier: Use the dynamic wrapper libXrdPosixPreload.so
- putting this library in LD_PRELOAD will intercept all Posix i/o calls and use the XRootD versions.
- All commands, like cp, ls, od, cat, ... will subsequently use XRootD for "root://" files and ordinary Posix I/O for everything else

XRootD POSIX Wrapper

- I tried it with h5dump-shared and it worked!
- Throwing caution to the winds, an *art* job also worked, streaming an HDF5 input file.
- I told David Adams how to use it and he discovered a problem streaming more than one file
- Found to be a problem with our code not closing all the open objects in an HDF5 file (attributes, groups, datasets all ought to be closed if opened)
- I found that there's an option in HDF5 to force closing of all open objects when a file is closed. Solves that problem!

XRootD, UPS and CVMFS

- The existing XRootD product in LArSoft's CVMFS repository lacks libXrdPosix.so and libXrdPosixPreload.so
- To do the aforementioned tests, I had to build xrootd from source
- I put in a ticket to the Scisofot team asking for a version of the XRootD UPS product that has these libraries and which defines the LD_PRELOAD symbol
- Perhaps there was a reason for the omission?
- We'll see what the discussion is on the ticket.
- One wart I found: The arm forge gui debugger interferes with an *art* job streaming HDF5 files with the dynamic XRootD wrapper. Other file I/O done by the *art* job seemed to work.

Returning to the VFL

- If the dynamic wrapper is distasteful, we can still use XrdPosix.hh to redefine all the file i/o.
- It may be as simple as #including XrdPosix.hh at the *end* of H5private.h
- Or #including XrdPosix.hh in the VFL drivers after #including H5private.h
- Chasing multiple #defines works in simple cases.
- Need to make the hdf5 UPS product depend on the xrootd UPS product then.
- This way, only the HDF5 VFL is affected, not all I/O in a job.

Writing files?

- I've only used XRootD to read files.
- Yujun Wu: "Only write closed files to dCache"
- But streaming output data may have benefits as well.
- I am unaware of us using XRootD on output on DUNE, other than to xrdcp files around.
- XrdPosix.hh defines writing methods too