HDF5 for Raw Data on DUNE: Offline Feedback and Plans

Tom Junk

DUNE Large Data

October, 2021

Current Status

- DAQ group plans on writing raw data in HDF5 format moving forwards
- Starting with Vertical-Drift coldbox data
- Did so on suggestion from computing experts
- Technical advantages for high-throughput writing? Parallel writes?
- Not known yet – ROOT files that have not been closed properly are usually unreadable.
  - Does HDF5 work better? From the HDF5 User Group meeting: they are working on a journaled write.
  - Do we want it to? Is a partial file more trouble than it's worth? Deadtime accounting.

Current Status

- July 2020: NP04 data written in HDF5 format as a test.
- Data cataloged in SAM
- 281 files. Example file: np04_timeSliceData_run011765_0036_dl1.hdf5
- Kurt Biery wrote an *art* input source that reconstitutes artdaq fragments in memory from data in this format.
- Used H5Cpp.h from the hdf5 UPS product, containing the HDF group's C++ interface.
- Fragments when written to a file look enough like those from previous DAQ running that they could be decoded using the ProtoDUNE-SP offline decoders.

Current Status

- As of larsoft v09_16_00, which depends on hdf5 v1_12_0b, H5Cpp.h has been removed from the hdf5 UPS product.
- I recoded Kurt's access methods using the C API.

code is in dune-raw-data/HDFUtils

input source is in dune-raw-data/ArtModules

- DAQ format changed in 2021.
- Metadata such as timestamps and the run number moved from attributes to packed words at the beginning of the datasets.
- Data format documentation: hdf5_dump.py available in the DAQ environment.

Current Status

- Input utilities in dune-raw-data updated for the new format
- Old (2020) format no longer supported.
- Version attribute changed name:

attribute_style_version →
data_format_version

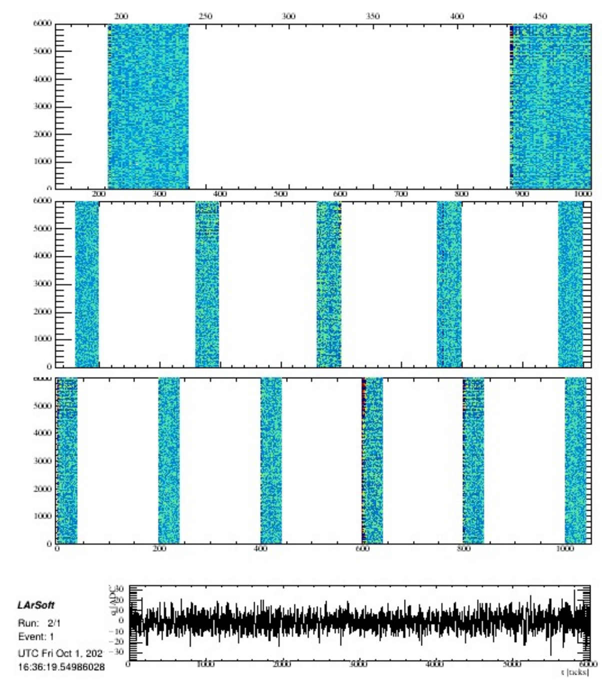
- It is important, when the data format evolves, that enough information is present to discern the format.
- We can test for the presence of a particular attribute, so this isn't quite the problem that moving the version bits in the WIB frame was.

Header Information In New Data

Data Format verison: 2 Error code: 0
Top-Level Group Name: TriggerRecord00001
Detector type: TPC
Geo path: TriggerRecord00001/TPC/APA000
Data Set Path: TriggerRecord00001/TPC/APA000/Link00
Data Set Path (underscore version): TriggerRecord00001_TPC_APA000_Link00
Data Set Size (bytes): 3801168
Retrieved data: ecode: 0 first byte: 22 last byte: 8f
Magic word: 0x11112222
Version: 3
Frag Size: 3801168
Trig Num: 1
Trig Timestamp: 81654470580516100
Window Begin: 81654470580316100
Window End: 81654470580520900
Run Number: 2
Error bits: 0
Fragment type: 1
GeoID version: 1
GeoID type: 0
GeoID region: 0
GeoID element: 0

Current Status

- VD coldbox data has, in TPC datasets, 80 bytes of header and 8192 WIB frames. Dataset type: `H5T_STD_I8LE`
- n.b. no automatic conversion of endianness or other format – just a stream of bytes.
- Can reconstitute FELIX fragments as before, or unpack directly using FelixFormat.hh
- Example VD coldbox data displayed with LArSoft event display using ProtoDUNE-SP channel map (!)



ProtoDUNE-SP WIB Frame Format

DUNE-Doc-1701-v3

Created: 13Sep2016
Updated: 09/20/2017 (no change for RCE from 08/02/2017)
Version: 1.1-rc2-RCE

Word	K/D	31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0	Data Source	Notes
1	0001	Reserved (8)		SlotNo	CrateNo	FiberNo	Version = 0x1	SOF		WIB	SOF is K28.5(0xBC) Format version currently = 1.
2	0000	WIB Errors				Reserved (14)			OOS MM	WIB	OOS: bad syncs coming from the PDTS. MM: COLDATA convert
3	0000	Timestamp [31:0]								WIB	
4	0000	Z Timestamp [62:48] or WIB counter [3]				Timestamp [47:32]				WIB	Bits [30:16] of word 3 may be either the upper 8 bits of the tim
5	0000	ChkSm B [7:0]		ChkSm A [7:0]		Reserved (8)		Stream 2 ERR	Stream 1 ERR	COLDATA 1/WIB	WIB-CD1
6	0000	COLDDATA Convert Count				ChkSm B [15:8]		ChkSm A [15:8]		COLDATA 1	16 bit frame checksum will be calculated for each 8 bit wide str increments with 2 MHz ADC Covert; resets with SYNC
7	0000	Reserved				Error Register				COLDATA 1	Reports alignment errors on input streams; TBD
8	0000	HDR8	HDR6	HDR7	HDR5	HDR4	HDR2	HDR3	HDR1	COLDATA 1	Header Bits
9	0000	ADC2 CH2[3:0]	ADC2 CH1[11:8]	ADC1 CH2[3:0]	ADC1 CH1[11:8]	ADC2 CH1[7:0]		ADC1 CH1[7:0]		COLDATA 1	
10	0000	ADC2 CH3[7:0]		ADC1 CH3[7:0]		ADC2 CH2[11:4]		ADC1 CH2[11:4]		COLDATA 1	
11	0000	ADC2 CH4[11:4]		ADC1 CH4[11:4]		ADC2 CH4[3:0]	ADC2 CH3[11:8]	ADC1 CH4[3:0]	ADC1 CH3[11:8]	COLDATA 1	
12	0000	ADC2 CH6[3:0]	ADC2 CH5[11:8]	ADC1 CH6[3:0]	ADC1 CH5[11:8]	ADC2 CH5[7:0]		ADC1 CH5[7:0]		COLDATA 1	
13	0000	ADC2 CH7[7:0]		ADC1 CH7[7:0]		ADC2 CH6[11:4]		ADC1 CH6[11:4]		COLDATA 1	
14	0000	ADC2 CH8[11:4]		ADC1 CH8[11:4]		ADC2 CH8[3:0]	ADC2 CH7[11:8]	ADC1 CH8[3:0]	ADC1 CH7[11:8]	COLDATA 1	
15	0000	ADC4 CH2[3:0]	ADC4 CH1[11:8]	ADC3 CH2[3:0]	ADC3 CH1[11:8]	ADC4 CH1[7:0]		ADC3 CH1[7:0]		COLDATA 1	
16	0000	ADC4 CH3[7:0]		ADC3 CH3[7:0]		ADC4 CH2[11:4]		ADC3 CH2[11:4]		COLDATA 1	
17	0000	ADC4 CH4[11:4]		ADC3 CH4[11:4]		ADC4 CH4[3:0]	ADC4 CH3[11:8]	ADC3 CH4[3:0]	ADC3 CH3[11:8]	COLDATA 1	
18	0000	ADC4 CH6[3:0]	ADC4 CH5[11:8]	ADC3 CH6[3:0]	ADC3 CH5[11:8]	ADC4 CH5[7:0]		ADC3 CH5[7:0]		COLDATA 1	
19	0000	ADC4 CH7[7:0]		ADC3 CH7[7:0]		ADC4 CH6[11:4]		ADC3 CH6[11:4]		COLDATA 1	
20	0000	ADC4 CH8[11:4]		ADC3 CH8[11:4]		ADC4 CH8[3:0]	ADC4 CH7[11:8]	ADC3 CH8[3:0]	ADC3 CH7[11:8]	COLDATA 1	
21	0000	ADC6 CH2[3:0]	ADC6 CH1[11:8]	ADC5 CH2[3:0]	ADC5 CH1[11:8]	ADC6 CH1[7:0]		ADC5 CH1[7:0]		COLDATA 1	
22	0000	ADC6 CH3[7:0]		ADC5 CH3[7:0]		ADC6 CH2[11:4]		ADC5 CH2[11:4]		COLDATA 1	
23	0000	ADC6 CH4[11:4]		ADC5 CH4[11:4]		ADC6 CH4[3:0]	ADC6 CH3[11:8]	ADC5 CH4[3:0]	ADC5 CH3[11:8]	COLDATA 1	
24	0000	ADC6 CH6[3:0]	ADC6 CH5[11:8]	ADC5 CH6[3:0]	ADC5 CH5[11:8]	ADC6 CH5[7:0]		ADC5 CH5[7:0]		COLDATA 1	
25	0000	ADC6 CH7[7:0]		ADC5 CH7[7:0]		ADC6 CH6[11:4]		ADC5 CH6[11:4]		COLDATA 1	
26	0000	ADC6 CH8[11:4]		ADC5 CH8[11:4]		ADC6 CH8[3:0]	ADC6 CH7[11:8]	ADC5 CH8[3:0]	ADC5 CH7[11:8]	COLDATA 1	
27	0000	ADC8 CH2[3:0]	ADC8 CH1[11:8]	ADC7 CH2[3:0]	ADC7 CH1[11:8]	ADC8 CH1[7:0]		ADC7 CH1[7:0]		COLDATA 1	
28	0000	ADC8 CH3[7:0]		ADC7 CH3[7:0]		ADC8 CH2[11:4]		ADC7 CH2[11:4]		COLDATA 1	
29	0000	ADC8 CH4[11:4]		ADC7 CH4[11:4]		ADC8 CH4[3:0]	ADC8 CH3[11:8]	ADC7 CH4[3:0]	ADC7 CH3[11:8]	COLDATA 1	
30	0000	ADC8 CH6[3:0]	ADC8 CH5[11:8]	ADC7 CH6[3:0]	ADC7 CH5[11:8]	ADC8 CH5[7:0]		ADC7 CH5[7:0]		COLDATA 1	
31	0000	ADC8 CH7[7:0]		ADC7 CH7[7:0]		ADC8 CH6[11:4]		ADC7 CH6[11:4]		COLDATA 1	
32	0000	ADC8 CH8[11:4]		ADC7 CH8[11:4]		ADC8 CH8[3:0]	ADC8 CH7[11:8]	ADC7 CH8[3:0]	ADC7 CH7[11:8]	COLDATA 1	
33	0000	ChkSm B [7:0]		ChkSm A [7:0]		Reserved (8)		Stream 2 ERR	Stream 1 ERR	COLDATA 2/WIB	WIB-CD2
34	0000	COLDDATA Convert Count				ChkSm B [15:8]		ChkSm A [15:8]		COLDATA 2	16 bit frame checksum will be calculated for each 8 bit wide str increments with 2 MHz ADC Covert; resets with SYNC
35	0000	Reserved				Error Register				COLDATA 2	Reports alignment errors on input streams; TBD
36	0000	HDR8	HDR6	HDR7	HDR5	HDR4	HDR2	HDR3	HDR1	COLDATA 2	Header Bits
37	0000	ADC2 CH2[3:0]	ADC2 CH1[11:8]	ADC1 CH2[3:0]	ADC1 CH1[11:8]	ADC2 CH1[7:0]		ADC1 CH1[7:0]		COLDATA 2	
38	0000	ADC2 CH3[7:0]		ADC1 CH3[7:0]		ADC2 CH2[11:4]		ADC1 CH2[11:4]		COLDATA 2	
39	0000	ADC2 CH4[11:4]		ADC1 CH4[11:4]		ADC2 CH4[3:0]	ADC2 CH3[11:8]	ADC1 CH4[3:0]	ADC1 CH3[11:8]	COLDATA 2	
40	0000	ADC2 CH6[3:0]	ADC2 CH5[11:8]	ADC1 CH6[3:0]	ADC1 CH5[11:8]	ADC2 CH5[7:0]		ADC1 CH5[7:0]		COLDATA 2	
41	0000	ADC2 CH7[7:0]		ADC1 CH7[7:0]		ADC2 CH6[11:4]		ADC1 CH6[11:4]		COLDATA 2	
42	0000	ADC2 CH8[11:4]		ADC1 CH8[11:4]		ADC2 CH8[3:0]	ADC2 CH7[11:8]	ADC1 CH8[3:0]	ADC1 CH7[11:8]	COLDATA 2	

Just the first few rows. 256 channels (FELIX link)

Crate, slot, fiber numbers set by the WIB. Used to identify channels.

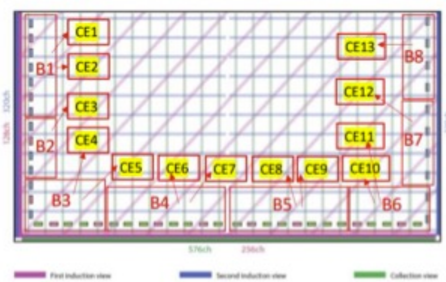
HDF5 dataset headers have redundant geometrical information. What to do if they conflict?

Things we need

- short term: Vertical drift coldbox channel map

Slide from Wenqiang Gu and Nitish Nayak

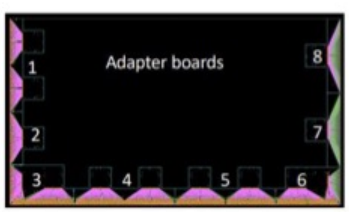
Channel Mapping -- BDE

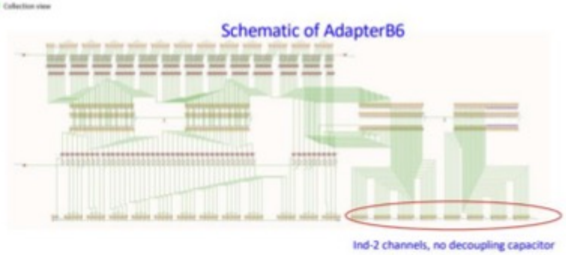


Adapter Boards (total: 1600 channels)
B1: 224 = 7*32 channels
B2: 128 = 4*32
B3: 256 = 8*32
B4: 256 = 8*32
B5: 256 = 8*32
B6: 224 = 7*32
B7: 128 = 4*32
B8: 128 = 4*32

# of ind-2 channels is consistent with the schematics

CE modules (13 modules, 128 chs / module)





- Two main pieces for the channel mapping :
  - Strip IDs to Connector pins on Adapter boards
  - Connector -> CE channels
- We have the CAD drawings and PCB schematic files from Bo.
  - Understand how to match strip IDs to the connector pins
  - Also know which view this corresponds to
  - Agreed on convention to number them with Slavic, should be consistent with how its done in the VD simulation as well
- BDE specific mapping between Connector on adapter board to CE channels is still a bit unclear
  - Especially for Adapter boards 1 and 6 which have some unconnected pins

Need Channel Map for VD Coldbox

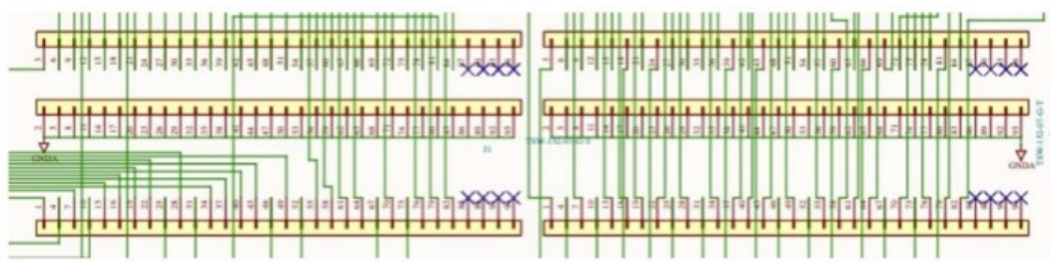
Slide from Wenqiang Gu and Nitish Nayak

Channel Mapping -- BDE

CE-1 (or CE-2): 128 channels

- 16 unused channels (Cheng-Ju) : 0, 1, 2, 3, 46, 47, 48, 49, 80 81, 82, 83, 124, 125, 126, 127

Adapter board layout



- We see the right number of unused channels but the numbering scheme is unclear
  - For eg : CE channel # 46 is unused but which pin on the connector does it correspond to?
- Once this is worked out, I think the channel map can be completed
- Mapping from WIB channels to offline – same scheme as NP04, already used by Tom to produce RawDigits from test data fragments

Short-Term Needs

- Need a new raw decoder tool

dunetpc/dune/Protodune/singlephase/RawDecoding/PDSPTPCDataInterface_tool.cc

That one has a lot of baggage from ProtoDUNE-SP

(RCE or FELIX)*(Container or Noncontainer Fragments)*(Compressed or non-compressed data)*(read all data or just for a specified APA)

Input classes in dune-raw-data and dunepdsprce handled decompression automatically.

New tool (and a producer module if we want to keep that) will have to use the new channel map (doesn't yet exist).

Short-Term Needs

- UPS product for DUNE-DAQ/dataformats

  https://github.com/DUNE-DAQ/dataformats

- Currently use the old FelixFormat.hh out of dune_raw_data, but the dataformats product will be maintained moving forwards

- Needs a Jenkins script and a release manager

Short-Term Needs

- A VD geometry that corresponds to the VD coldbox strips.
  - Is a FD module or workspace adequate? Just use a subset of channels?
  - Steve Kettell said the number of channels per CRU in the design has recently changed from 1600 to 1536. Coldbox has the old channel count/mapping?
- Someone should make wire endpoint location dumps: useful in checking the channel map.
  - ProtoDUNE-SP example: https://cdcvs.fnal.gov/redmine/projects/dunetpc/wiki/_ProtoDUNE-SP_Wire_Dumps_

Art's event loop

Just one place where the source is called. 👉

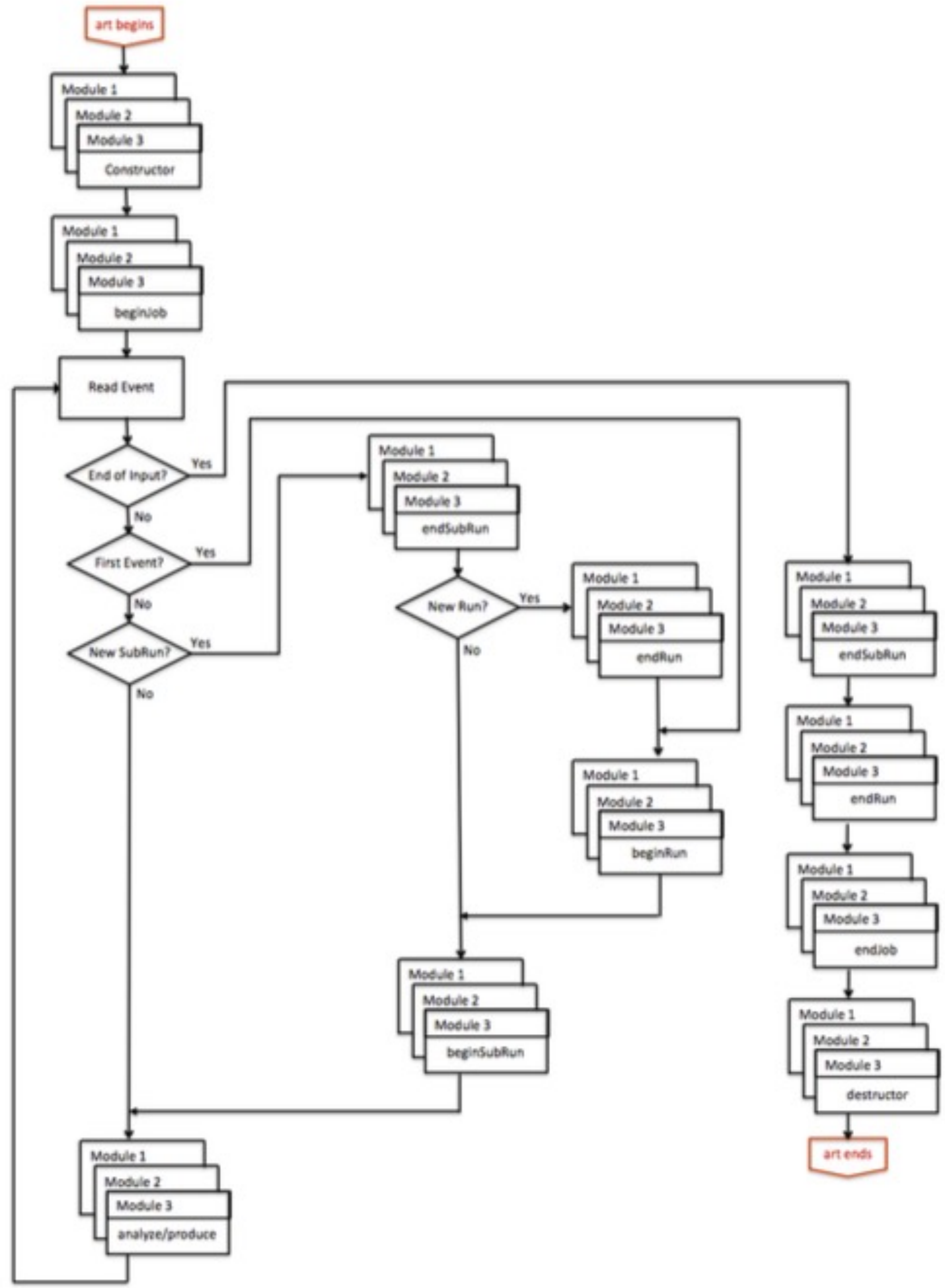


Fig. 3.2 from the *art* workbook

Longer-Term Needs

- Need to be able to read in a subset of the data from a trigger record into memory.
- ProtoDUNE-SP decoder tool took advantage of *art*'s delayed reader
- Input source does not actually read data from the rootfile. Only when getByLabel or getHandle or other getters are called are data read
- Decoder tool calls removeCachedProduct to free up *art* event memory

A Better Source

- Currently: HDF5 source gets its filename from a fcl parameter
- Would like to use *art*'s file handling features.
  - specify input files on the command line
  - or from a list of input files
- Would like to delay reading in data.
  - Either use *art*'s delayed-reader functionality (how much do we have to do here?), or
  - Expose enough information about the HDF5 file so downsteam methods can do the readin themselves. The filename is enough. Or a file handle if we want to keep the file open (just an hid_t)
  - HDF5 is thread-safe with a global lock on access methods, if compiled with the right switch (do we use it?)
  - I tried a simple program that opens a file twice for read and it worked (though I only read using one of the file descriptors)

Differences between HDF5 and ROOT

- HDF5 is good at storing arrays of data. Names (groups and attributes) provide metadata and allow access to the arrays.
- ROOT stores data associated with C++ classes and provides features for schema evolution.
- Either way, we are writing most of the serializer/deserializer functionality
- Good not to have the extra ROOT buffer needed for automated schema evolution– we can do that part ourselves.
- Both ROOT and HDF5 (I assume) translate machine architecture-dependent formats. By using flat arrays of bytes in both, we don't take advantage of that functionality.

Where to Put Deserializer Code?

- Source or tool?
- The tool knows what portion of the data to read (each APA or CRU)
- Delayed-reader would have to invoke the deserializer, or the caller of delayed reader would have to deserialize
- Do we have to define the intermediate, un-deserialized data products so the delayed reader knows how to communicate the request back to the source?
- *art*'s getHandle and similar methods require data product definitions.
- Intermediate, short-lived data buffers would need data products defined for them.
- Would like to simply hand data to deserialization code.

Longer-Term Goals and Feedback

- Currently have been focusing on just ingesting HDF5-formatted data as-is from the DAQ into LArSoft.
- We do not want to persist multiple formats of the same data
  - $30k per petabyte for long-term storage.
  - 30 petabytes/year for DUNE
- LArSoft jobs should be able to read in raw data files from the DAQ.
- LArSoft processing is slow and data decompression/reformatting is the least of the computational needs.
- Need to control memory usage

Longer-Term Goals and Feedback

- A problem: No replacement of xrootd for HDF5. HDF5 Users Group Meeting news: people are working on this.
- Can use xrdcp still of course, but xrootd's streaming functionality helps ease the burden on computing resources
- But what about using HDF5 for what it is "good" at?
- ML tools expect HDF5-formatted input data
- Distribution of data inside a supercomputer to the various nodes inside. What tools are there? Do they map onto what we need?
- Do we need to reformat the DAQ data so they are easier to import directly to non-LArSoft applications?
- Can we read HDF5 data directly into WireCell?
- What about the dataprep stage?
- Decompression? Rearranged waveform data compresses better.
- Event mixing?

Dataprep Functionality

- Works on a per-APA basis for TPC data
- Sticky-code mitigation
- Bad-channel flagging
- Correlated-noise removal
- AC-coupling tail correction
- Pedestal subtraction
- Gain adjustment

Downstream uses of HDF5

- Data from the DAQ are produced in a highly-controlled environment.
- Few consumers need to be maintained in order to read it
  - DAQ debugging
  - Online monitoring
  - LArSoft
- But analysis-tier data can be written in any format that fits the needs of specific analyses
- ML programs can read in processed (dataprep, deconvolved, filtered) data.
- Users may wish to do analyses with modern tools.
- ROOT has developed over many years to address HEP needs, but other tools have larger external ecosystems and may be easier to use.
- It's possible to make a misleading plot with any tool, or misunderstand the functionality.