Reading

LEGEND-specific:

- GitHub: Encoded waveforms load slower than compressed waveforms #77
- Confluence: Investigation of LH5 File Structure and I/O

Other

- HDF5 format
- HDF5 datasets
- **`hdf5plugin`** Python module
- LZF filter distributed with **`h5py`**

Basics

- LH5 uses two custom encoders for `raw` Ge waveforms
  - `ZigZag` for `waveform_presummed`
  - `radware-sigcompress` for `waveform_windowed`
- These achieve good compression ratio but are very slow
- HDF5 / h5py are distributed with standard filters (GZIP, SZIP, LZF, LZ4, etc.) - compare these to our custom ones to see if we can improve
- Drop-in replacement - no action by users needed, just change a compression argument
- Two metrics: **compression ratio** and **decompression speed**
- Mostly don't care about compression speed since `raw` is generated infrequently

Advanced

- LH5 stores each variable/column as a separate HDF5 dataset
- Compression filters require chunking of data - an entire chunk is compressed together
  - → Want chunks of reasonable size (not too small → bad compression ratio, not too large → slow decompression)
- Datasets (columns) are chunked independently (i.e. not across datasets) and each row is one event → most of our data is so small that we can’t use a big enough chunk size for good performance.
- Custom encoded waveforms are a single chunk → probably too large for good performance
- For compression tests with standard filters, just going to let chunking handle itself and not specify → HDF5 will pick something reasonable and we don’t really care.

Compression tests

- Tests performed on perlmutter `/global/cfs/`
- input file = "l200-p08-r000-phy-20231004T160832Z-tier_raw.lh5" (input file size: 1.6 GB)

compression filter	compression write time (s)	overall disk file size (GB)	
custom encoders (default)	**-**	**1.6**	
no compression	109	9.4	
GZIP	193	1.9	20% larger
SZIP	149	1.9	
LZF	**120**	**2.4**	50% larger
LZ4	111	2.6	

Decompression tests

- Tests performed on perlmutter `/dvs_ro/`- read all Ge channels in the file
  1. `waveform_presummed`
  2. `waveform_windowed`
  3. everything else (default is GZIP compressed, I believe, but these times might be influenced by other stuff? - grain of salt)

- Each test performed 5 times in a row, compare only last ~3 attempts to remove influence of file caching (average by eye)

compression filter	**`waveform_presummed` decompression time (s)**	**`waveform_windowed` decompression time (s)**	**everything else decompression time (s)**
custom encoders (default)	**23.7**	**17**	**4.5**
no compression	~1.3	1.1	4.3
GZIP	5.3	7.8	3.6
SZIP	9.3	10.6	3.5
LZF	**3.2** 7x faster	**4.6** 3x faster	**3.5**
LZ4	3.0	3.8	3.5

tested on 2 different days and got similar results (within ~10%)

Takeaways

- **recommend switching to LZF** - file size is 50% larger but speed increase is nearly an order of magnitude
  - speed increase measured reading all Ge channels, all waveforms
  - suspect that reading random waveforms will be relatively even faster due to chunking layout - not tested - (and the way LH5.store.read works, we don't access random rows but read the whole thing in and then slice it).
- recommend LZF over GZIP due to 2x better decompression speed - we can handle the larger file size for `raw`
- could also consider LZ4, ~10% worse compression ratio for ~10% speed increase