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	
blosc2(lz4, DEFLATE, 9)	~240	2.6	62% larger

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
blosc2(lz4, DEFLATE, 9)	1.9 12x faster	2.5 6x faster	3.6

Decompression tests under different data access patterns

- using `lgdo.lh5.store.read()` with different options (all Ge channels)
  1. accessing the whole data without using the `idx` or `n_rows` parameters
  2. accessing a row-contiguous subset of the data (i.e. providing `n_rows` or `idx` when `idx` is a contiguous range starting at 0, e.g. [0,1,2,...])
  3. accessing a random waveform in the middle, e.g. with `idx`

  4.-6. the same as 1-3 but using the flag `h5_idx = True`

- Just reading 25 channels to speed things up in testing

	scenario 1 all rows, no idx or n_rows		scenario 2a first 100 rows with idx		scenario 2b all 2936 rows with idx		scenario 3a idx = [1000]		scenario 3b idx = [10#]		scenario 4 all rows, no idx or n_rows, use_h5idx=True		scenario 5a first 100 rows with idx, use_h5idx=True		scenario 5b all 2936 rows with idx, use_h5idx=True		scenario 6a idx = [1000], use_h5idx=True		scenario 6b idx = [10#], use_h5idx=True	
compression filter	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)	waveform_presummed decompression time (s)	waveform_windowed decompression time (s)
custom encoders (default)	6.0	5.0	0.22	0.2	~6.8	~5	0.28	0.28	0.22	0.26	4.3	3.5	0.16	0.14	4.3	3.3	0.39	0.33	2.9	2.4
no compression	0.22	0.3	0.07	0.08	0.22	0.33	0.2	0.4	0.2	0.3	0.14	0.21	0.05	0.06	0.14	0.2	0.05	0.05	0.1	0.1
GZIP	1.5	2.1	0.14	0.19	1.7	2.3	1.9	2.2	1.6	2.2	1.1	1.5	0.11	0.14	1.1	1.6	0.11	0.10	0.7	0.6
SZIP	2.4	3.0	0.22	0.18	2.8	3.8	3.0	3.6	2.9	3.1	1.9	2.1	0.16	0.18	1.8	2.2	0.16	0.11	1.1	0.7
LZF	1.0	1.3	0.09	0.12	1.0	1.4	1.2	1.6	1.0	1.3	0.7	1.0	0.08	0.10	0.7	1.0	0.09	0.08	0.4	0.35
LZ4	0.8	1.2	0.09	0.1	0.8	1.2	0.9	1.2	0.8	1.1	0.6	0.9	0.08	0.09	0.6	0.8	0.08	0.07	0.37	0.31
blosc2(LZ4, DEFLATE)	**0.6**	**0.8**	**0.06**	**0.07**	**0.7**	**0.8**	**0.5**	**0.7**	**0.7**	**0.9**	**0.4**	**0.6**	**0.06**	**0.07**	**0.4**	**0.5**	**0.07**	**0.06**	**0.25**	**0.21**

Takeaways

- **recommend switching to LZF or blosc2(LZ4, DELTA, 9)** - file size is 50-62% 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`