RFC: HDF5 Parallel Compressed Writer Requirements and Use Cases

Mark Basham, Nick Rees, Tobias Richter and Jon Thompson

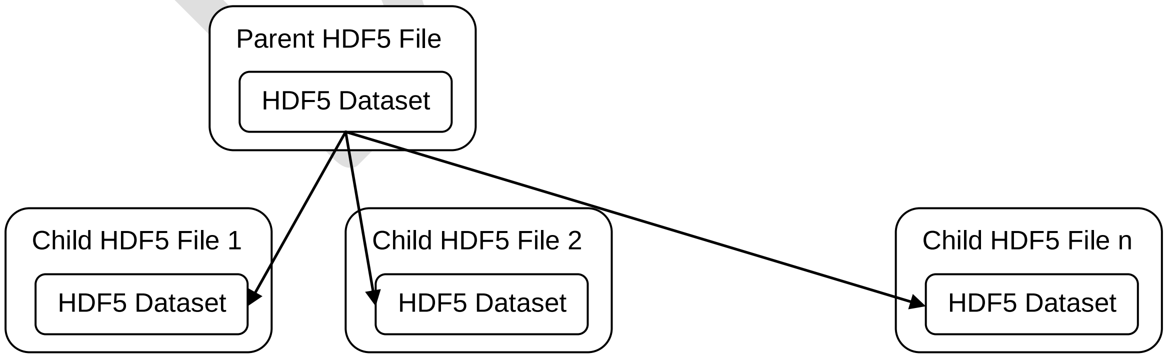
1 Introduction

An increasing number of synchrotron detectors have high data rates and highly parallel architectures. In addition, most of the relevant detectors are 2-D imaging ones and the traditional approach of one file per data frame is approaching its limits because, with frame rates in the kHz region, it is testing the metadata performance of high throughput file systems. Hence, there is a move to store data in compressed, mostly 3-dimensional datasets. Since HDF5 doesn't support compressed parallel writing this has lead to two approaches:

1. Doing compression in parallel out of the library, and then passing it through a single writing thread. This is ultimately limited by the single writing thread.
2. Forego compression and use parallel HDF5 to write with multiple processes. This means that the 10-100 times data reduction that you can sometimes obtain with compression isn't available. It also requires that the writers be MPI processes, which often doesn't work well with other messaging and event mechanisms used by the detector control systems.

To address this problem this, RFC envisages multiple processes writing compressed HDF5 files completely independently and in parallel, but allowing a reading process to access the data in these files via a composite dataset (most probably in another file) which links to datasets in the files via some form of "multiple soft link" syntax, It is anticipated that that there would be relatively few (<10) files being written, but this should not be a hard limit – it is just that too many files will either test the file system or slow down reading because of the file open overheads.

Finally, some data analysis problems are also embarrassingly parallel, and are included as possible use cases.



2 Requirements

1. Multiple processes must be able to construct a single HDF5 dataset in parallel with no inter-process communication.
2. The dataset must be able to have compressed chunks.
3. Readers must be able to use SWMR to read the dataset while it is being written.
4. The baseline design is to have multiple underlying HDF5 files, with a dataset in a higher level HDF5 file being comprised of multiple "soft-links" to the lower level datasets.
5. All files must be able to be read as valid HDF5 files.
6. If a lower-level file or dataset doesn't exist, it should be treated the same as if a chunk is missing in a normal dataset.
7. Performance should be similar to a normal HDF5 dataset, with a possible fixed overhead per file open (however, parallelizing the file opens could be investigated).
8. It is acceptable to have all soft-links for one dataset specified by a single printf-like statement.
9. It is acceptable to require dimensionality of the underlying files to be specified by a fixed layout definition like chunk layouts (with a stride and offset specified if we allow the Percival use case). If the underlying file has a different dimensionality additional data can be ignored, and missing data handled as in 6 above.
10. It is acceptable to require the underlying files to have the same chunk layout.

3 Use Cases

In this section are presented a number of use cases from systems we know are being developed at the present time. A summary of the use cases are:

1. Excalibur (6 identical focal plane areas being written in parallel, 1 chunk/focal plane area/frame).
2. Dual PCO.edge (2 identical focal plane areas written in parallel, with chunking in the time dimension, and each focal plane area spread across ~20 chunks).
3. Eiger (One system writing a series of files, each file consisting of a fixed number of consecutive frames, 1 chunk/file).
4. Percival Frame builder. (8 systems each writing a full frame with a time stride of 8, but each with a different time phase). This is not a critical use case since the system is in such an early stage of development that other designs are possible.
5. Dual PCO.edge tomography analysis (~20 different systems reading output of (2) with an x-t frame, processing it, and writing ~20 separate files which are tomographic reconstructions. Possibly a SWMR reader, so as the dimension of the dataset being read grows, additional x-t frame extents are read incrementally).
6. Other Data analysis use cases (preferably some which aren't three dimensional).
7. Possibly sparse arrays?

Note: at time of this draft (v0.2 dated 5 April 2013) the use cases for the final three cases haven't been generated yet....

3.1 Excalibur

EXCALIBUR is an advanced photon counting detector being designed and built by a collaboration of Diamond and the STFC. It is based around 48 CERN Medipix3 chips arranged as an 8 x 6 array for a total frame size (including gaps) of 2069 x 1793. The hardware architecture is shown in figure 1 below:

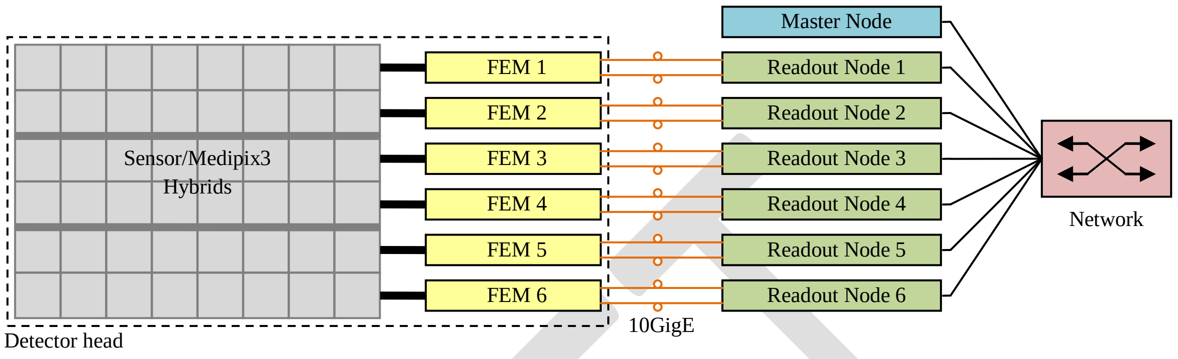


Figure 1: Excalibur hardware architecture.

It can be seen that each row of eight Medipix 3 chips is read by a front end module (FEM) into a readout node computer. The six readout nodes then all write their portion of the image to the output file in parallel. This process repeats for each image, roughly (but not completely) in synchronism. There are gaps between the Medipix 3 chips, shown in figure 2. The small (3 pixel) gaps are filled by the readout software, but the large gaps are not.

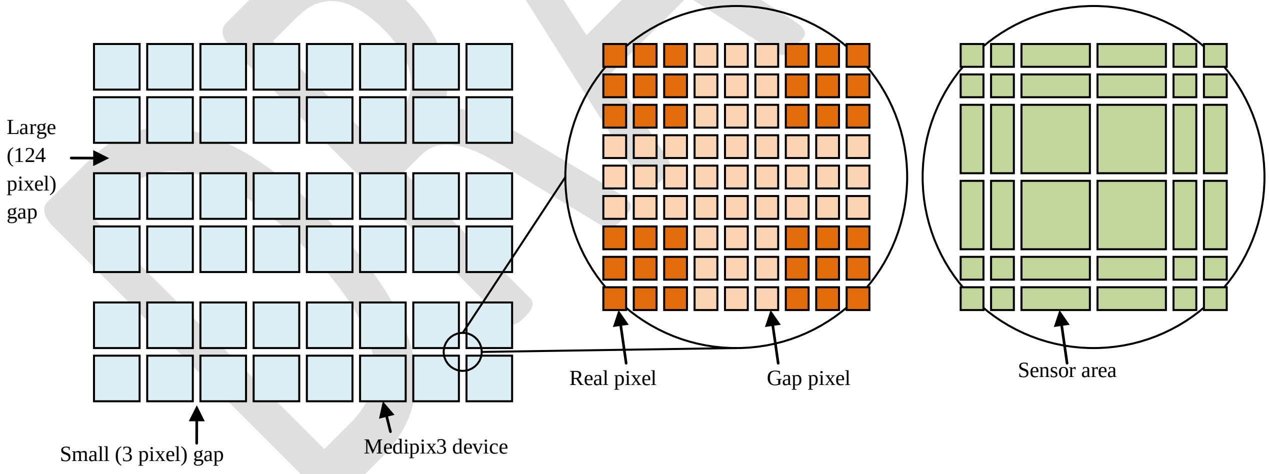


Figure 2. Excalibur Medipix3 chip layout and gap details.

The HDF requirements are therefore:

1. Each full image to consist of six horizontally cut chunks. A readout node will write its part of the total image as a single chunk. Chunk size is 2069x259x1 (odd numbered nodes), or 2069x256x1 (even numbered nodes); the chunk streams do not have exactly the same spatial dimensions.

2. Gaps between nodes 2 & 3 and nodes 4 & 5 (124 pixels wide) to be constant filled (from the point of view of the reader).
3. Writing of the six streams of chunks to occur in parallel.
4. No requirement for strict synchronisation between the streams. The total number of chunks written by each stream will be the same, but the timing between streams is not synchronised.
5. Compression of one chunk stream to have no dependency on other chunk streams. In other words, compression must have no spatial requirement across chunk boundaries. Temporal dependencies are allowed.

3.2 Dual PCO Edge

Two identical detectors acquiring images simultaneously. Hardware triggering keeps the image streams roughly in synchronisation. This is a very similar case to Excalibur, the main differences being the gaps and the chunking of the image parts. The gap may be particularly tricky as it will depend on the method of mounting the two cameras, but this is not really an HDF problem.

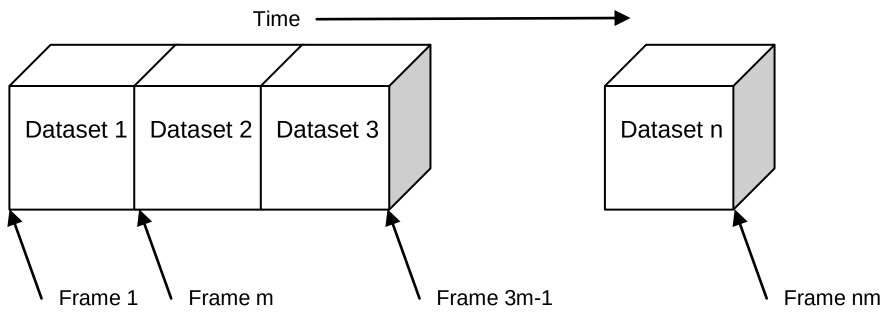
The HDF requirements are:

1. Full images to consist of a sub-image from each camera captured at the same time.
2. Each camera sub-image may consist of data from multiple chunks.
3. Chunking may occur in the time dimension as well as the spatial dimensions.
4. Writing of the two streams of chunks to occur in parallel.
5. No requirement for strict synchronisation between the streams. The total number of chunks written by each stream will be the same, but the timing between streams is not synchronised.
6. Compression of one chunk stream to have no dependency on the other chunk stream.
7. Gap between the cameras to be constant filled (from the point of view of the reader).

3.3 Eiger

Eiger is a high speed detector system being developed by DECTRIS Ltd. It is based on combining arrays of the ~256x256 pixel Eiger chip developed by PSI. These are combined into a 4x2 array to form a module and this basic module building block will be used to build up the final systems, which are available in a number of sizes.

Type	Modules	Active area [mm2]	Pixel array	Frame rate [Hz]	DCB-PC [Gbps]
1M	2	77 x 80	1030 x 1065	3000	40
4M	8	155 x 163	2070 x 2167	750	40
9M	18	233 x 245	3110 x 3269	333	40
16M	32	311 x 328	4150 x 4371	187	40



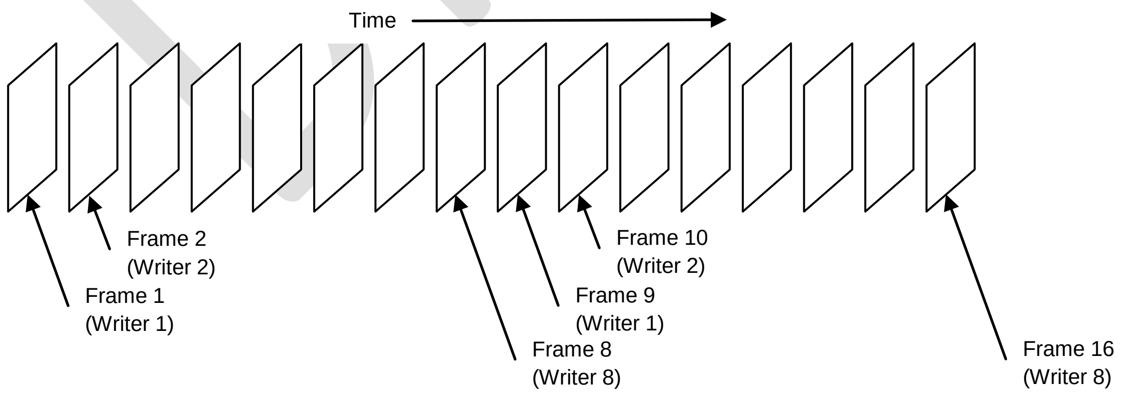
In the proposed software design it is intended to write data as a time series of files, each containing a pixel data arranged in a 3-D dataset of a fixed number of 2-dimensional frames. Each 2-D frame, will be an HDF5 chunk that is compressed outside the HDF5 library. So, when written, the data comprises n files, each with a fixed number of m frames, and a total of n*m frames. In the baseline design all data is written by a single process, so there is no parallel writing required.

The HDF5 requirements are:

- Writer must be able to do out-of-library compression.
- Multiple files written independently must be able to be viewed as a single dataset.
- Top level dataset is a 3-D stack of frames composed of a concatenation along the slowest dimension of mostly fixed sized lower level 3-D data sets. The only exception to the fixed size is that the last dataset may have fewer frames.

3.4 Percival Frame Builder

This system is currently under development and the baseline design writes ~4000 x ~4000 pixel frames at 120Hz. The detector data format is 14bit, so we anticipate 2 byte pixels, so the overall data rate is about 3 GB/sec. There is an engineering mode proposed which has 4 bytes/pixel. One proposed way of parallelising the data path is with multiple parallel writers, each writing a full frame, but only one writer for any specific frame. Thus each writer is writing a part of the total dataset with a fixed stride and a different offset in the slowest changing dimension. In the baseline design there are 8 backend writing systems, so the stride length is 8 frames. The total number of frames is not



necessarily a multiple of the number of writers, so the number of frames written by the writers may differ by 1.

The HDF5 requirements for this system are:

- Writing systems to be writing in parallel and completely independent.
- Data written by a single writer to a lower level dataset has a fixed stride and offset in the slowest changing dimension of the upper level dataset.
- Writer should be able to write compressed data in the lower level dataset.

It should be noted that since this is not the final design for the Percival system it should not be considered as a top level requirement if implementation proves difficult.