

Layout in HDF5 is defined as the way in which a multidimensional dataset is mapped to the serial file 1-dimensional stream of data. Layout can be contiguous (dataset is serialized entirely into a monolithic block on disk) and chunked (dataset is split into multiple “chunks” which are all stored separately in the file). Chunks can then be read and written individually, improving performance when operating on a subset of the dataset. It is possible to fine tune I/O to maximize performance for any access pattern by modifying the chunk size.

Figure 1 Chunk definition in RUN4 1 x 1 x 251

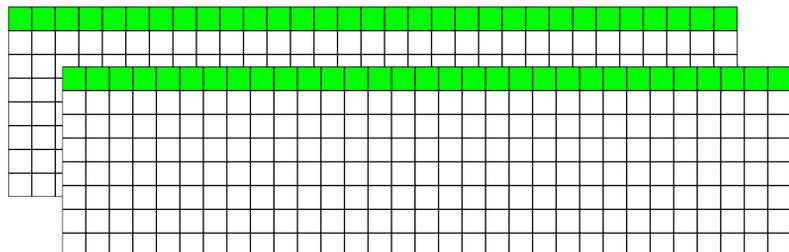
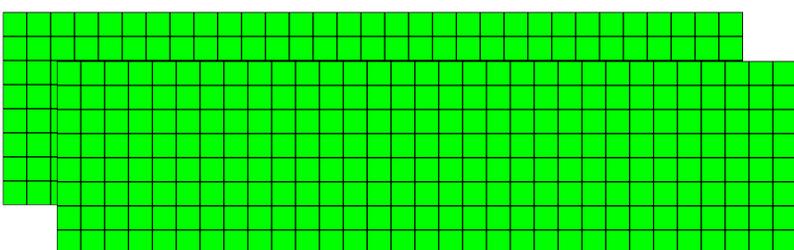


Figure 2 Chunk definition in RUN2-3 64 x 64 x 251



The NeXus HDF5 file format was used, and some parameters regarding compression/chunking/layout were modified in each run.

Table 1 shows these definitions. The X, Y, Z in Table 1 refer to Figures 1 and 2 (dataset as Z layers, Y columns, X rows)

Table 2 show the results for runs regarding BSS_5663 (dimensions 64 x 64 x 251). Column named Time refers to the time it took to save the all file. The percentage values are all relative to RUN1. In percentage terms (red cells) the chunked/compressed are expected to be slower then the contiguous case, but produce smaller files.

Best performance is obtained using a 64 x 64 x 251. This definition is shown in the Figure 2. Worst performance is obtained defining 64 (Z) x 64 (Y) chunks of 1 x 251. This definition is shown in the Figure1

Table 1

RUN	Chunk definition	
RUN1		Contiguous layout, no compression. This run is expected to be the fastest to complete but it also produces the largest files.
RUN2	[Z][Y][X]	Chunked layout, no compression
RUN3	[Z][Y][X]	Chunked layout, H5Z_FILTER_DEFLATE compression
RUN4	[1][1][X]	Chunked layout, H5Z_FILTER_DEFLATE compression.

Table 2

RUN	Chunk dimensions	Compression	Time	%	Rate	File size	%
			(millisec)				
1			585		60.8	36,474	
2	64 x 64 x 251		609	-4%	58.5	36,511	0%
3	64 x 64 x 251	H5Z_FILTER_DEFLATE	650	-11%	0.6	402	99%
4	1 x 1 x 251	H5Z_FILTER_DEFLATE	1,839	-214%	1.6	3,015	92%

Links

<http://www.nexusformat.org/>
<http://www.hdfgroup.org/>
<http://www.space-research.org/>



Best performance is obtained for RUN1 (contiguous layout). After that, starting with adding chunking (RUN2), compression (RUN3) and a small chunk size (RUN4), performance decreases.

For RUN4, the chunk is too small. When chunks are made smaller, there are more of them in the dataset (for RUN4 there are 4096 chunks). When performing I/O on a dataset, if there are many chunks in the selection, it will take extra time to look up each chunk. In addition, since the chunks are stored independently, more chunks results in more I/O operations, further compounding the issue. The extra metadata needed to locate the chunks also causes the file size to increase as chunks are made smaller.

Conclusion: Chunk definition for RUN3 was adopted. However, the optimization for one instrument might not work well for other instrument data layouts.

In particular, defining just one chunk might not work well for larger datasets. Because the entire chunk must be read from disk and decompressed before performing any operations, this will impose a great performance penalty when operating on a small subset of the dataset if the cache is not large enough to hold the one-chunk dataset. In addition, if the dataset is large enough, since the entire chunk must be held in memory while compressing and decompressing, the operation could cause the operating system to page memory to disk, slowing down the entire system.

