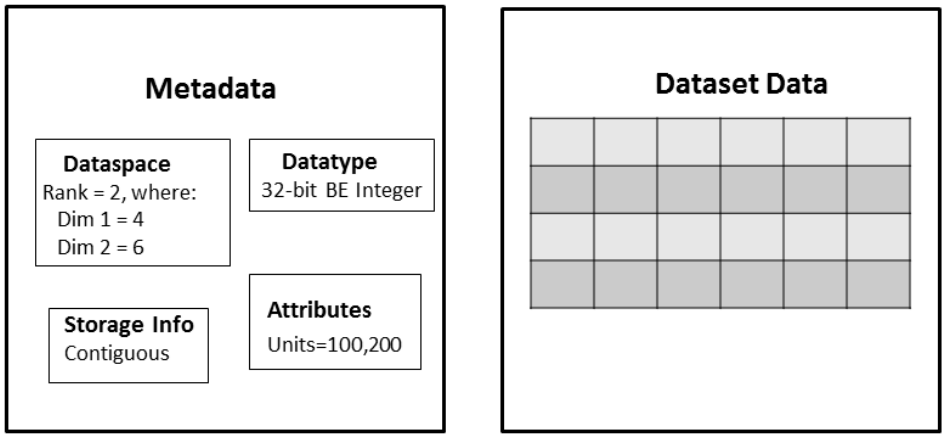
Dataset Storage Layout

Description of a Dataset

The Creating a Dataset tutorial topic defines a dataset as a multidimensional array of data elements *together with supporting metadata*, where:

- The array of elements consists of the raw data values that a user wishes to store in HDF5.
- The supporting metadata describes that data. The metadata is stored in the dataset (object) header of a dataset.

Datatype, dataspace, attribute, and *storage layout information* were introduced as part of the metadata associated with a dataset:



Dataset Storage Layout

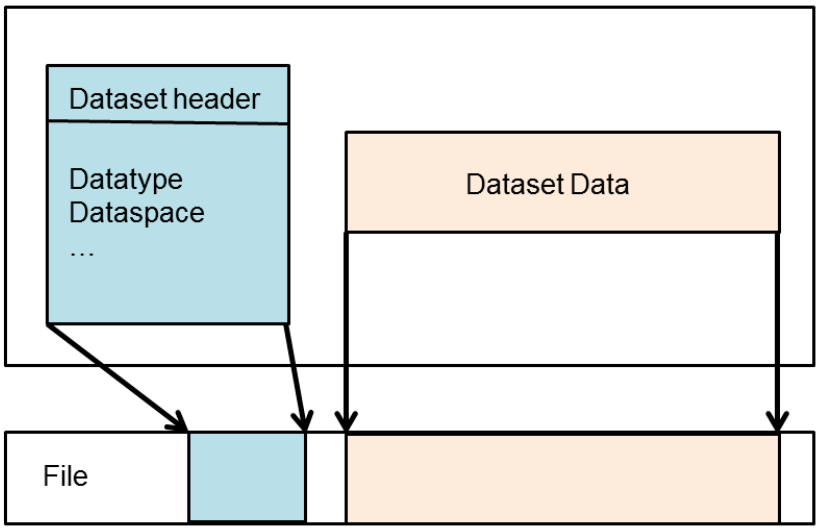
The storage information, or storage layout, defines how the raw data values in the dataset are physically stored on disk. There are three ways that a dataset can be stored:

- contiguous
- chunked
- compact

See the H5Pset_layout/H5Pget_layout APIs for details.

Contiguous

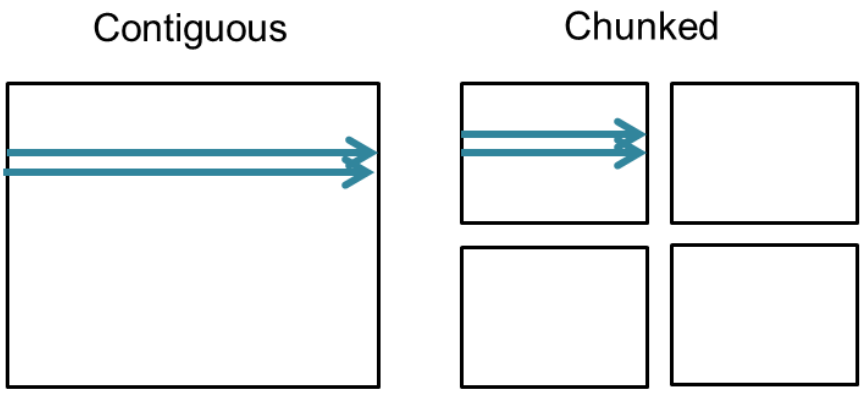
If the storage layout is contiguous, then the raw data values will be stored physically adjacent to each other in the HDF5 file (in one contiguous block). This is the default layout for a dataset. In other words, if you do not explicitly change the storage layout for the dataset, then it will be stored contiguously.



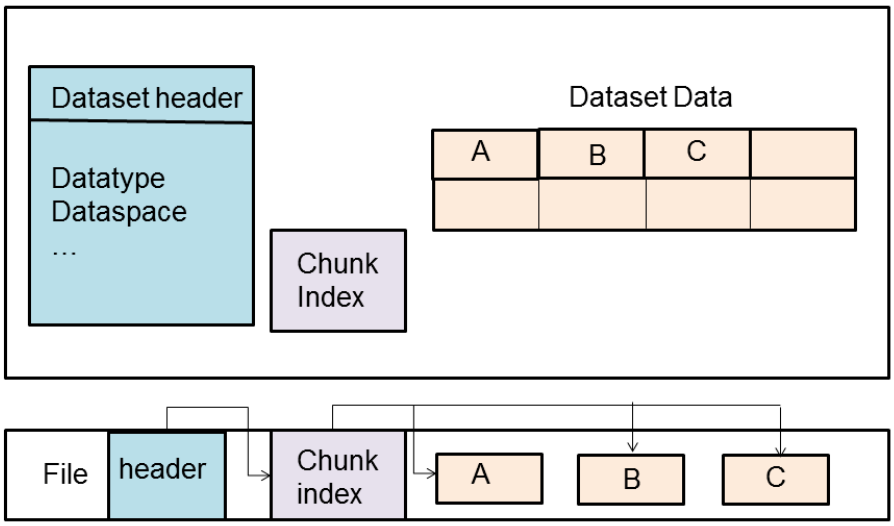
Contiguous Storage Layout

Chunked

With a chunked storage layout the data is stored in equal-sized blocks or chunks of a pre-defined size. The HDF5 library always writes and reads the entire chunk:



Each chunk is stored as a separate contiguous block in the HDF5 file. There is a chunk index which keeps track of the chunks associated with a dataset:



Chunked Storage Layout

Why Chunking ?

Chunking is required for enabling compression and other filters, as well as for creating extendible or unlimited dimension datasets.

It is also commonly used when subsetting very large datasets. Using the chunking layout can greatly improve performance when subsetting large datasets, because only the chunks required will need to be accessed. However, it is easy to use chunking without considering the consequences of the chunk size, which can lead to strikingly poor performance.

Note that a chunk always has the same rank as the dataset and the chunk's dimensions do not need to be factors of the dataset dimensions.

Writing or reading a chunked dataset is **transparent** to the application. You would use the same set of operations that you would use for a contiguous dataset. For example:

```
H5Dopen (...);
   H5Sselect_hyperslab (...);
   H5Dread (...);
```

Problems Using Chunking

Issues that can cause performance problems with chunking include:

- Chunks are too small.

  If a very small chunk size is specified for a dataset it can cause the dataset to be excessively large and it can result in degraded performance when accessing the dataset. The smaller the chunk size the more chunks that HDF5 has to keep track of, and the more time it will take to search for a chunk.
- Chunks are too large.

  An entire chunk has to be read and uncompressed before performing an operation. There can be a performance penalty for reading a small subset, if the chunk size is subtantially larger than the subset. Also, a dataset may be larger than expected if there are chunks that only contain a small amount of data.

- A chunk does not fit in the *Chunk Cache*.

  Every chunked dataset has a chunk cache associated with it that has a default size of 1 MB. The purpose of the chunk cache is to improve performance by keeping chunks that are accessed frequently in memory so that they do not have to be accessed from disk. If a chunk is too large to fit in the chunk cache, it can significantly degrade performance. However, the size of the chunk cache can be increased by calling H5Pset_chunk_cache.

It is a good idea to:

1. Avoid very small chunk sizes, and be aware of the 1 MB chunk cache size default.
2. Test the data with different chunk sizes to determine the optimal chunk size to use.
3. Consider the chunk size in terms of the most common access patterns that will be used once the dataset has been created.

Compact

A compact dataset is one in which the raw data is stored in the object header of the dataset. This layout is for very small datasets that can easily fit in the object header.

The compact layout can improve storage and access performance for files that have many very tiny datasets. With one I/O access both the header and data values can be read. The compact layout reduces the size of a file, as the data is stored with the header which will always be allocated for a dataset. However, the object header is 64 KB in size, so this layout can only be used for very small datasets.

Programming Model to Modify the Storage Layout

To modify the storage layout, the following steps must be done:

- Create a Dataset Creation Property list. (See H5P_CREATE)
- Modify the property list.
  To use chunked storage layout, call: H5P_SET_CHUNK
  To use the compact storage layout, call: H5P_SET_LAYOUT
- Create a dataset with the modified property list. (See H5D_CREATE)
- Close the property list. (See H5P_CLOSE)

For example code, see the HDF5 Examples page. Specifically look at the Examples by API. There are examples for different languages.

The C example to create a chunked dataset is: h5ex_d_chunk.c
The C example to create a compact dataset is: h5ex_d_compact.c

Changing the Layout after Dataset Creation

The dataset layout is a Dataset Creation Property List. This means that once the dataset has been created the dataset layout cannot be changed. The h5repack utility can be used to write a file to a new with a new layout.

Sources of Information

Chunking in HDF5 (See the documentation on Advanced Topics in HDF5)
See "Properties and Property Lists" in the HDF5 User's Guide.