2\_User Guide

Audience: An HDF5 library application developer

As a user creates an HDF5 file and manages the associated HDF5 objects, the library services file space requests from three internal storage pools as described below:

* Free-space manager

This is a data structure that tracks free-space sections of different sizes when file space is released. There is a free-space manager for each memory allocation type e.g. raw data, different types of metadata etc.

* Aggregator

The aggregator is a contiguous block of bytes allocated by the library. File space is sub-allocated from the block. There is an aggregator for metadata and an aggregator for raw data.

* Virtual File Driver

File space request is served by the virtual file driver’s allocation routine. The allocation is normally from the end of file.

There are four strategies evolved from how the above three storage pools are used in fulfilling space request:

1. H5F\_FILE\_SPACE\_ALL\_PERSIST

The HDF5 library will service file space requests from all three storage pools, first from the free-space manager, then the aggregator and finally the virtual file driver. The space tracked by the free-space manager in the HDF5 file persists when the file is closed.

1. H5F\_FILE\_SPACE\_ALL

This is the library’s default file space handling strategy. This is the same as the previous strategy except that the space tracked by the free-space manager in the HDF5 file is not persistent at file closing.

1. H5F\_FILE\_SPACE\_AGGR\_VFD

The HDF5 library will service file space requests first from the aggregator and then from the virtual file driver’s allocation routine.

1. H5F\_FILE\_SPACE\_VFD

The HDF5 library will service file space requests only from the virtual file driver’s allocation routine.

The HDF5 library provides three command line tools that users can use in manipulating file space. The first tool is *h5dump* by which the user can find out the file space handling strategy in use for the file. For example, the following *h5dump* *–B* output for *persist.h5* indicates that the file uses strategy #1 in handling file space:

HDF5 "persist.h5" {

SUPER\_BLOCK {

SUPERBLOCK\_VERSION 2

:

:

:

FILE\_SPACE\_STRATEGY H5F\_FILE\_SPACE\_ALL\_PERSIST

FREE\_SPACE\_THRESHOLD 1

}

The second tool is *h5stat* by which the user can find out the file’s distribution of file space. Look at the following *h5stat* output for *persist.h5:*

Filename: persist.h5

:

:

:

Storage information:

Superblock: 48

Superblock extension: 88

User block: 0

Object header (total/unused):

Groups: 40/0

Datasets: 544/288

Datatypes: 0/0

Groups:

B-tree/List: 872

Heap: 120

Attributes:

B-tree/List: 0

Heap: 0

Chunked datasets:

B-tree: 0

Shared Messages:

Header: 0

B-tree/List: 0

Heap: 0

Free space managers:

Header: 153

Amount of free space (in bytes): 374

:

:

:

Dataset storage information:

Total raw data size: 240

Note that *persist.h5* has 374 bytes of free space. To get more specific distribution of free space in the file, look at the output for *h5stat –s* :

Filename: persist.h5

Small size free-space sections (< 10 bytes):

Total # of small size sections: 0

Free-space section bins:

# of sections of size 10 - 99: 2

# of sections of size 100 - 999: 1

Total # of sections: 3

Note that *persist.h5* has three separate free space sections but there are no small holes of less than 10 bytes in the file.

Finally, the command line tool *h5repack* allows the user to create a new file with a specified file space strategy from an existing file. For example, the user can re-pack *persist.h5* with file space handling strategy #2 to become *not\_persist.h5* as follows:

h5repack –S 2 persist.h5 not\_persist.h5

Beside command line tools, the HDF5 library also provides file creation property public routines for users to manipulate file space when the file is created:

*herr\_t H5Pset\_file\_space(hid\_t fcpl\_id, H5F\_file\_space\_t strategy, hsize\_t threshold)*

This routine sets the strategy that the library will use in handling file space for the file asscociated with *fcpl\_id*. *strategy* is one of the four types specified above. Note that the *strategy* set via this public routine cannot be changed once the file is created. However, the user can use *h5repack* to repack the file with a different strategy. A positive non-zero threshold value will notify the free space managers to track free space sections >= *threshold*.

A zero value for strategy and/or threshold will retain the existing strategy and/or threshold set in the file creation property.

herr\_t H5Pget\_file\_space(hid\_t fcpl\_id, H5F\_file\_space\_t \* strategy, hsize\_t \*threshold)

This routine retrieves the strategy and the free space section threshold that the library uses in managing file space.

See the following sample coding for the usage of the two public routines:

/\* Create file-creation property template \*/

fcpl = H5Pcreate(H5P\_FILE\_CREATE);

/\* Set the file space management strategy to use for the file \*/

H5Pset\_file\_space(fcpl, H5P\_FILE\_SPACE\_AGGR\_VFD, 0);

/\* Create the file with the file-space info \*/

fid = H5Fcreate(“file”, H5F\_ACC\_TRUNC, fcpl, H5P\_DEFAULT);

H5Pget\_file\_space(fcpl, &strategy, NULL);

/\* Close the file \*/

H5Fclose(fid1);

The above output shows that the total of metadata and raw data (96 + 40 + 272 + 872 + 120 + 400) is 1800, while the file size is 2448 with a discrepancy of 648 bytes. The missing space is due to the default file space strategy used internally by the library that does not persist free space being tracked at file closing (that means free space is lost at file closing and cannot be reclaimed.)

The total of metadata and raw data (48 + 88 + 40 + 128 + 872 + 120 + 135 + 617) is 2448, which is equal to the file size. Note that there is 617 bytes of free space in the file. To further see the distribution of free space in the file, use *h5stat –s* option on *out\_example.h5*: