2. Basic HDF5 File Space Management

Audience:

A user who handles HDF5 files and has knowledge of the HDF5 data model,   
but who may not be familiar with the HDF5 library API or internals.

The HDF5 library manages the allocation of space in an HDF5 file for storing file metadata and HDF5 dataset values. It also manages free space that results from the manipulation of the file’s HDF5 objects. The HDF5 library uses one of several available file space management strategies in performing these management activities for a given HDF5 file.

HDF5 command line utilities allow users to view any HDF5 file’s contents, obtain information about its file space and file space management, and create a copy of the file with a different file space management strategy.

The following examples describe various HDF5 file usage patterns and illustrate how different file space management strategies can affect the HDF5 file size.

## Scenario A: Default File Space Management Strategy

### Session 1: Create an Empty File

In the first session, a user creates an HDF5 file named *no\_persist\_A.h5* and closes the file without adding any HDF5 objects to it. No file space management strategy is specified, so the file is created with the default file space management strategy (H5F\_FILE\_SPACE\_ALL, defined elsewhere).

The *h5dump* utility displays the contents of a given HDF5 file. The following *h5dump* output, generated with the command ‘*h5dump no\_persist\_A.h5’,* shows the initial contents of *no\_persist\_A.h5*:

HDF5 "no\_persist\_A.h5" {

GROUP "/" {

}

}

This reveals that the HDF5 library automatically created the root group and allocated space for initial file metadata when *no\_persist\_A.h5* was created. This empty HDF5 file does not yet contain any user-created HDF5 objects.

The *h5stat –S* command reports information on the file space for a given HDF5 file. The report for the file *no\_persist\_A.h5* is shown:

Filename: no\_persist\_A.h5

Summary of file space information:

File metadata: 800 bytes

Raw data: 0 bytes

Amount/Percent of tracked free space: 0 bytes/0.0%

Unaccounted space: 0 bytes

Total space: 800 bytes

Note that *no\_persist\_A.h5* contains 800 bytes of file metadata and nothing else; there is no user data and no free space in the file. The file size of the empty HDF5 file *no\_persist\_A.h5* equals the size of the file metadata.

### Session 2: Add Datasets

In this session, a user opens the empty HDF5 file *no\_persist\_A.h5,* adds four datasets (*dset1*, *dset2*, *dset3, and dset4*) of different sizes, and closes the file*.*

Running *h5dump* *–H* on the updated file produces the following output:

HDF5 "no\_persist\_A.h5" {

GROUP "/" {

DATASET "dset1" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 10 ) / ( 10 ) }

}

DATASET "dset2" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 30000 ) / ( 30000 ) }

}

DATASET "dset3" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 50 ) / ( 50 ) }

}

DATASET "dset4" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 100 ) / ( 100 ) }

}

}

}

*h5stat –S* for the updated *no\_persist\_A.h5* reports:

Filename: no\_persist\_A.h5

Summary of file space information:

File metadata: 2216 bytes

Raw data: 120640 bytes

Amount/Percent of tracked free space: 0 bytes/0.0%

Unaccounted space: 1976 bytes

Total space: 124832 bytes

The 120640 bytes of raw data is the space that stores the data values in the four new dataset objects. The amount of tracked free space in the file is 0 bytes, while there are 1976 bytes of unaccounted space. The unaccounted space is due to the file space management strategy in use for the *no\_persist\_A.h5* HDF5 file.

The HDF5 library’s default file space management strategy does not persist tracked free space information across multiple sessions with an HDF5 file. This means the information about free space that is collected by the library during the current session (since the file was opened) is not saved when the file is closed. With the default strategy, free space that is incurred during a particular session can be reused during that session, but is unavailable for reuse in all future sessions. This unavailable file free space is reported as “unaccounted space” in the *h5stat -S* output.

As demonstrated in this example, file free space can be created not only when HDF5 objects are deleted from a file, but also when they are added. This is because adding an object may introduce gaps in the file as new space is allocated for file metadata and HDF5 dataset values. HDF5 files that might develop large amounts of unaccounted space are candidates for non-default file space management strategies if file size is a concern.

### Session 3: Add One Dataset and Delete Another

In session 3 with *no\_persist\_A.h5,* a user opens the file, adds a new dataset (*dset5*), and then deletes an existing dataset (*dset2*) before closing it. After the file is closed, *h5dump –H* outputs the following:

HDF5 "./no\_persist\_A.h5" {

GROUP "/" {

DATASET "dset1" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 10 ) / ( 10 ) }

}

DATASET "dset3" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 50 ) / ( 50 ) }

}

DATASET "dset4" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 100 ) / ( 100 ) }

}

DATASET "dset5" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 1000 ) / ( 1000 ) }

}

}

}

*h5stat –S* reports:

Filename: ./no\_persist\_A.h5

Summary of file space information:

File metadata: 2216 bytes

Raw data: 4640 bytes

Amount/Percent of tracked free space: 0 bytes/0.0%

Unaccounted space: 124024 bytes

Total space: 130880 bytes

At this point, the amount of unaccounted space consists of the 1976 bytes that were there when the user opened the file, and the additional free space incurred in the latest session due to the addition of *dset5* and the deletion of *dset2*. The HDF5 file *no\_persist\_A.h5* now contains fragments of lost space resulting from the manipulation of the HDF5 objects in the file and the use of the default file space management strategy.

Note that the *no\_persist\_A.h5* file space is now almost 95% unaccounted space and the 12000 bytes of space that originally stored the data values for *dset2* make up a substantial fraction of that. HDF5 files that will have dataset objects deleted from them are candidates for non-default file space management strategies if file size is a concern.

## Scenario B: Alternative File Space Management Strategy

### Session 1: Create an Empty File

In the first session of this scenario, a user creates an HDF5 file named persist\_B.h5using a non-default file space management strategy (H5F\_FILE\_SPACE\_ALL\_PERSIST, defined elsewhere). The file is closed before any HDF5 objects are added to it.

### Session 2: Add Datasets

The HDF5 file persist\_B.h5 is re-opened and the same four datasets (*dset1*, *dset2*, *dset3, and dset4*) that were added to *no\_persist\_A.h5* in Scenario A, Session 2 are added to persist\_B.h5before it is closed.

*h5stat –S* for the updated *persist\_B.h5* reports:

Filename: ./persist\_B.h5

Summary of file space information:

File metadata: 2391 bytes

Raw data: 120640 bytes

Amount/Percent of tracked free space: 1854 bytes/1.5%

Unaccounted space: 0 bytes

Total space: 124885 bytes

In contrast to *no\_persist\_A.h5* after Session2, persist\_B.h5 contains no unaccounted space. It does, however, contain 1854 bytes of tracked free space. The amount of file metadata in *persist\_B.h5* (2391 bytes) is slightly larger than what was in *no\_persist\_A.h5* (2216 bytes). This increase is due to the extra metadata used by the library to save the tracked free space information.

The *h5stat –s* command shows more detail about the distribution of tracked free space persist\_B.h5:

Filename: persist\_B.h5

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

Total # of small size sections: 0

Free-space section bins:

# of sections of size 10 - 99: 1

# of sections of size 1000 - 9999: 1

Total # of sections: 2

There are two free-space sections in *persist\_B.h5*; one section contains between 10 and 99 bytes and the second contains between 1000 and 9999 bytes.

### Session 3: Add One Dataset and Delete Another

A user reopens *persist\_B.h5*, adds *dset5,* deletes *dset2,* and closes the file. After the file is closed *h5stat –S* reports:

Filename: ./persist\_B.h5

Summary of file space information:

File metadata: 2427 bytes

Raw data: 4640 bytes

Amount/Percent of tracked free space: 121854 bytes/94.5%

Unaccounted space: 0 bytes

Total space: 128921 bytes

The amount of tracked free space after the addition of *dset5* and deletion of *dset2* reflects the 1854 bytes of tracked free space that was previously in the file and the free space adjustments resulting from the dataset object manipulations in Session 3.

In this scenario, the HDF5 library allocated space for the file metadata for *dset5* from the pool of tracked free space; the free space in the pool resulted from activities in Session 2. When *dset2* was deleted, the bytes that were used for that dataset’s raw data and file metadata were added to the file’s tracked free space by the HDF5 library. The tracked free space information was saved (persisted) when the file was closed. Although the file persist\_B.h5 still contains unused bytes in the form of tracked free space, it is 5995 bytes smaller than the file *no\_persist\_A.h5* was after Session 3 in Scenario A because the HDF5 library was able to reuse free space incurred in Session 2.

*h5stat –s* shows the distribution of free space in *persist\_B.h5* at the end of Session 3:

Filename: ./persist\_B.h5

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

Total # of small size sections: 0

Free-space section bins:

# of sections of size 10 - 99: 1

# of sections of size 100 - 999: 1

# of sections of size 1000 - 9999: 1

# of sections of size 100000 - 999999: 1

Total # of sections: 4

Note that *persist\_B.h5* now has two additional free-space sections resulting from the manipulation of the HDF5 objects in the file during Session 3.

## Changing the File Space Management Strategy

The file space management strategy for a given HDF5 file is specified when the file is created; it cannot be changed thereafter.

As demonstrated in the previous scenarios, some usage patterns can benefit from non-default file space management strategies. It is not always possible to know in advance how a file will be used, and *h5stat –S* may show that a given file has a large amount of unaccounted space.

The HDF5 utility *h5repack* can be used to copy the contents of an existing HDF5 file to a new HDF5 file, reclaiming unaccounted space and tracked free space in the process. In addition to reclaiming space, *h5repack -S* allows the user to specify a different file space management strategy for the new HDF5 file. While this does not change the strategy used to manage file space in the original file, subsequent sessions with the new file will utilize the new file’s specified file space management strategy.

For example, the user can repack *no\_persist\_A.h5* with a non-default strategy that always allocates file space from the end of file, coded *VFD*. The new file is *no\_persist\_outvfd.h5:*

h5repack –S VFD no\_persist\_A.h5 no\_persist\_outvfd.h5

*h5stat –S* shows the following:

Filename: no\_persist\_outvfd.h5

Summary of file space information:

File metadata: 1632 bytes

Raw data: 4640 bytes

Amount/Percent of tracked free space: 0 bytes/0.0%

Unaccounted space: 0 bytes

Total space: 6272 bytes

Comparing this output with the *h5stat –S* output for *no\_persist\_A.h5* in Scenario A, Session 3 shows several differences. After repacking, there is no unaccounted space, the file metadata is smaller, and there is a substantial decrease in file size.

Although not apparent from the *h5stat* output, the file management strategy for *no\_persist­­\_outvfd.h5* is different from the default strategy used for *no\_persist\_A.h5.* Subsequent sessions that manipulate HDF5 objects in the new file, *no\_persist\_outvfd.h5,* will always operate under the “allocate file space from the end of file” file management strategy.

The next section discusses the file space management strategies supported by the HDF5 library and describes the public routines used to specify a non-default strategy or to learn what strategy is being used for an existing file.