HDF5 File Space Management

1. Introduction

The space within an HDF5 file is called its *file space*. When a user first creates an HDF5 file, the HDF5 library immediately allocates space to store information called file metadata. *File metadata* is information the library uses to describe the HDF5 file and to identify its associated objects. When a user subsequently creates HDF5 objects, the HDF5 library allocates space to store data values, as well as the necessary additional file metadata. When a user removes HDF5 objects from an HDF5 file, the space associated with those objects becomes *free space*. The HDF5 library manages this free space.

The HDF5 library *file space management* activities encompass both the allocation of space and the management of free space. The HDF5 library implements several *file space management strategies*, and the strategy used for a given HDF5 file is set when the file is created. Depending on the file's usage patterns, one strategy may be better than the others; an inappropriate strategy can lead to file size and access performance issues. HDF5 files that will have objects added or deleted in later sessions, or that will never have objects deleted, may benefit from the use of a non-default strategy.

This document describes how the file space management strategies affect file size and access time for various HDF5 file usage patterns. It also presents the HDF5 utilities and HDF5 library public routines that help users select appropriate file space management strategies for their specific needs.

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 are available that 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. Running *h5dump* shows the initial contents of *no_persist_A.h5*:

```
h5dump 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 data values in the four new dataset objects occupy the 120640 bytes of raw data space. 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 retain 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. Notice that there is still no tracked free space.

Note that the *no_persist_A.h5* file space is now almost 95% unaccounted space and the 120000 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.h5 using 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.h5 before 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 changes 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.

3. HDF5 File Space Allocation and Tuning

Audience:

An HDF5 application developer who has knowledge of the HDF5 library API.

The HDF5 library performs file space management activities related to tracking free space and allocating space to store file metadata and *raw data*, the data values in HDF5 dataset objects. Every HDF5 file has an associated file space management strategy that determines how the HDF5 library carries out these activities for the file.

3.1 File Space Allocation

The HDF5 library includes three different mechanisms for allocating space to store file metadata and raw data:

- Free-Space Managers

The HDF5 library's free-space managers track sections in the HDF5 file that are not being used to store file metadata or raw data. These sections will be of various sizes. When the library needs to allocate space, the free-space managers search the tracked free space for a section of the appropriate size to fulfill the request. If a suitable section is found, the allocation can be made from the file's existing free space. If the free-space manager cannot fulfill the request, the request falls through to the aggregator level.

- Aggregators

The HDF5 library has two aggregators. Each aggregator manages a block of contiguous bytes in the file that have not been allocated previously. One aggregator allocates space for file metadata from the block it manages; the other aggregator handles allocations for raw data. The maximum number of bytes in each aggregator's block is tunable.

If the library's allocation request exceeds the maximum number of bytes an aggregator's block can contain, the aggregator cannot fulfill the request and the request falls through to the virtual file driver level. After space has been allocated from an aggregator's block, that space is no longer managed by the aggregator (i.e. if it was freed later, the free-space manager would be in charge of it). Unallocated bytes in the block continue to be managed by the aggregator.

When an aggregator cannot fulfill an allocation request from the remaining space in its block, it requests a new block of contiguous bytes and any unallocated blocks that remain in the existing block become free space.

- Virtual File Driver

The HDF5 library's virtual file driver interface dispatches requests for additional space to the allocation routine of the file driver associated with an HDF5 file. For example, if the H5FD_SEC2 file driver is being used, its allocation routine will increase the size of the single file on disk that stores the HDF5 file contents to accommodate the additional space that was requested.

File Space Management Strategies

The HDF5 library provides several file space management strategies that control how it tracks free space and uses the free-space managers, aggregators, and virtual file driver to allocate space for file metadata and raw data. The strategies are:

Use all space allocation mechanisms. Track file free space across sessions.	H5F_FILE_SPACE_ALL_PERSIST (or ALL_PERSIST)
Use all space allocation mechanisms. Track file free space only in current session.	H5F_FILE_SPACE_ALL (or ALL)
Use only aggregator and VFD mechanisms. Never track free space.	H5F_FILE_SPACE_AGGR_VFD (or AGGR_VFD)
Use only VFD mechanism. Never track free space.	H5F_FILE_SPACE_VFD (or VFD)

Strategy 1: H5F_FILE_SPACE_ALL_PERSIST (*also called* ALL_PERSIST)

With this strategy, the HDF5 library's free-space managers track the free space that results from manipulating HDF5 objects in an HDF5 file. The tracked free space information is saved when the HDF5 file is closed, and reloaded when the file is re-opened. The tracked free space information **persist**s across HDF5 file sessions, and the free space managers remain aware of free space sections that became available in any file session.

With this strategy, when space is needed for file metadata or raw data, the HDF5 library first requests space from the free-space managers. If the request is not satisfied, the library requests space from the aggregators. If the request is still not satisfied, the library requests space from the virtual file driver. That is, the library will use **all** of the mechanisms for allocating space.

The H5F_FILE_SPACE_ALL_PERSIST strategy offers every possible opportunity for reusing free space. The HDF5 file will contain extra file metadata information about tracked free space. The HDF5 library will perform additional "accounting" operations to track free space, and to search the free space sections when allocating space for file metadata and raw data.

Strategy 2: H5F_FILE_SPACE_ALL (*also called* ALL)

This strategy is the HDF5 library's default file space management strategy. Prior to HDF5 Release 1.9.x, it was the only file space management strategy directly supported by the library.

With this strategy, the HDF5 library's free-space managers track the free space that results from manipulating HDF5 objects in an HDF5 file. The free space managers are aware of free space sections that became available in the current file session, but the tracked free space information is not saved when the HDF5 file is closed. Free space that exists when the file is closed becomes unaccounted space in the HDF5 file. Unallocated space in the aggregators' blocks may also become unaccounted space when the session ends.

As with the strategy ALL_PERSIST, the library will try **all** of the mechanisms for allocating space with the ALL strategy. When space is needed for file metadata or raw data, the HDF5 library first requests space from the free-space managers. If the request is not satisfied, the library requests space from the aggregators. If the request is still not satisfied, the library requests space from the virtual file driver.

The H5F_FILE_SPACE_ALL strategy allows free space incurred in the current session to be reused in the current session. There is no extra file metadata information about tracked free space in the HDF5 file. However, if free space exists when the file is closed the HDF5 file will contain unaccounted space that can never be reused. The HDF5 library will perform some additional "accounting" operations to track free space, but the amount of free space tracked and searched will usually be less than with the ALL_PERSIST strategy, so the number of operations should be less.

Strategy 3: H5F_FILE_SPACE_AGGR_VFD (*also called* AGGR_VFD)

With this strategy, the HDF5 library does not track the free space that results from manipulating HDF5 objects in an HDF5 file. All free space immediately becomes unaccounted space. Unallocated bytes in the aggregators' blocks when the file is closed may also become unaccounted space.

With this strategy, when space is needed for file metadata or raw data, the HDF5 library first requests space from the aggregators. If the request is not satisfied, the library requests space from the virtual file driver. That is, the library will try the **aggr**egator and **v**irtual **f**ile **d**river mechanisms for allocating space.

The H5F_FILE_SPACE_AGGR_VFD strategy never reuses free space. Because small allocation requests can be satisfied from the aggregators' blocks of contiguous bytes, this strategy will deliver better access performance for some file usage patterns. It may be appropriate when

access performance is the highest priority and there are many small writes. Because there are different aggregators for file metadata and raw data, this strategy tends to co-locate file metadata more than some other strategies that can reuse free space scattered throughout the file.

Strategy 4: H5F_FILE_SPACE_VFD (*also called* VFD)

With this strategy, the HDF5 library does not track the free space that results from the manipulation of HDF5 objects in an HDF5 file. All free space immediately becomes unaccounted space.

With this strategy, when space is needed for file metadata or raw data, the HDF5 library requests space from the **v**irtual **f**ile **d**river.

The H5F_FILE_SPACE_VFD strategy never reuses free space. Because allocation requests go directly to the virtual file driver, this strategy is best suited for HDF5 files whose primary file usage pattern consists of writing large amounts of raw data to extend dataset object(s).

Summary of File Space Management Strategies						
Full Name *used with H5Pset_file_space*	Short Name *used with HDF5 repack*	Track Free Space		Allocate Space Using		
		across multiple sessions	within single session	free-space managers	aggregators	virtual file driver
H5F_FILE_SPACE_ALL_PERSIST	ALL_PERSIST	Y	Y	Y	Y	Y
H5F_FILE_SPACE_ALL	ALL	N	Y	Y	Y	Y
H5F_FILE_SPACE_AGGR_VFD	AGGR_VFD	N	N	N	Y	Y
H5F_FILE_SPACE_VFD	SPACE_VFD	N	N	N	N	Y

Specifying a File Space Management Strategy

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

The HDF5 library provides the *H5Pset_file_space* file creation property routine so that users can specify the file space management strategy that should be used when a new HDF5 file is created (see entry in HDF5 Reference Manual). The signature for the routine is:

herr_t H5Pset_file_space(hid_t fcpl_id, H5F_file_space_t strategy, hsize_t threshold)

The first *H5Pset_file_space* parameter, *fcpl_id,* is the file creation property list identifier that will be used when the HDF5 file is created. The second parameter, *strategy,* is one of the four strategies described above. The third parameter, *threshold,* is the free-space section threshold used by the library's free-space managers. This parameter is mainly for performance tuning

purposes, and is discussed in more detail elsewhere. Passing a value of zero for either *strategy* or *threshold* indicates that the file's corresponding existing value should not be modified as a result of the call.

The library provides a companion routine that retrieves the file space management information for an HDF5 file (see entry in HDF5 Reference Manual):

herr_t H5Pget_file_space(*hid_t fcpl_id*, *H5F_file_space_t * strategy*, hsize_t **threshold*)

The first parameter, *fcpl_id,* is the file creation property list identifier associated with the HDF5 file. If the second parameter, *strategy,* is not NULL, the library will retrieve the existing file space management strategy in use for the file and store it in *strategy*. If the third parameter, *threshold,* is not NULL, the library will retrieve the existing free-space section threshold used by the library's free-space manager and store it in *threshold*.

The following code sample shows how these two public routines are used to create an empty HDF5 file, *persist.h5,* with the file space management strategy ALL_PERSIST:

```
/* Create a file creation property list template */
fcpl_id = H5Pcreate(H5P_FILE_CREATE);

/* Set the file space management strategy */
/* Don't update the free-space section threshold */
H5Pset_file_space(fcpl_id, H5P_FILE_SPACE_ALL_PERSIST, (hsize_t)0);

/* Create an HDF5 file with the file creation property list fcpl_id */
fid = H5Fcreate("persist.h5", H5F_ACC_TRUNC, fcpl_id, H5P_DEFAULT);

/* The strategy retrieved will be #1 H5F_FILE_SPACE_ALL_PERSIST */
/* The threshold retrieved will be 1 which is the library default */
H5Pget_file_space(fcpl_id, &strategy, &threshold);

/* Close the file */
H5Fclose(fid);
```

The *h5dump* command line utility reports the file space management information for an HDF5 file. See the following *h5dump –B* output for the file *persist.h5*:

```
HDF5 "persist.h5" {
SUPER_BLOCK {
  SUPERBLOCK_VERSION 2
  :
  :
```

```
    :
    FILE_SPACE_STRATEGY      H5F_FILE_SPACE_ALL_PERSIST
    FREE_SPACE_THRESHOLD   1
}
    :
    :
```

The output indicates that the HDF5 library will use the file space management strategy H5F_FILE_SPACE_ALL_PERSIST and a free-space section threshold of 1 (which is the default) when performing free space management activities on the HDF5 file *persist.h5.*

3.2 Tuning File Space Management

Each of the four file space management strategies has benefits and drawbacks. The appropriate strategy depends on the HDF5 file's usage pattern. In this section we cover the pros and cons of the various strategies in more detail, and use additional scenarios to demonstrate their effect on file size.

Recall the two HDF5 files *no_persist_A.h5* and *persist_B.h5*, used in Section 2 Scenario A, "Default File Space Management Strategy." By using the default file creation property identifier (H5P_DEFAULT) when creating *no_persist_A.h5*, the HDF5 library will automatically use the file space management strategy H5F_FILE_SPACE_ALL for the file. The code sample in the previous section demonstrated how to create the file *persist.h5* that would be managed using the H5F_FILE_SPACE_ALL_PERSIST strategy.

The prior sections have shown that strategy ALL_PERSIST has the benefit of reusing the tracked free space in the file across multiple file sessions, while strategy ALL has the drawback of accumulating unaccounted space in the file over multiple sessions. The key factor contributing to the benefit of strategy ALL_PERSIST is the usage pattern of manipulating (adding/deleting) HDF5 objects across multiple sessions. The fragmentation and unaccounted space with strategy ALL increases with the manipulation of HDF5 objects across sessions.

The H5F_FILE_SPACE_AGGR_VFD and H5F_FILE_SPACE_VFD strategies never use the HDF5 library's free-space manager to track released file space. Therefore, any unused space that results from the manipulation of HDF5 objects will be unaccounted space that can never be reused. For the AGGR_VFD and VFD strategies, the number of sessions in which manipulations occur has negligible (AGGR_VFD) or no (VFD) effect on the file size.

Discussions of the scenarios presented earlier in the Primer are expanded below, and HDF5 files with AGGR_VFD and VFD management policies are also shown. The datasets in this section are identical to those used in the previous scenarios.

Scenario C: ALL_PERSIST Strategy in Single Session

Session 1: Create file, manipulate objects

In the only session of this scenario, a user creates an HDF5 file named *persist_C.h5* using the H5F_FILE_SPACE_ALL_PERSIST strategy. The user then adds four datasets (*dset1, dset2, dset3,* and *dset4*), deletes *dset2*, and adds *dset5* before closing the file.

The file management strategy is the same strategy that was used in Scenario B. The HDF5 objects are manipulated in the same order as they were in Sessions 1-3 of Scenario B.

h5stat –S for *persist_C.h5* shows the following:

```
Filename: ./persist_C.h5
Summary of file space information:
  File metadata: 2409 bytes
  Raw data: 4640 bytes
  Amount/Percent of tracked free space: 117854 bytes/94.4%
  Unaccounted space: 0 bytes
Total space: 124903 bytes
```

The file size for *persist_C.h5* is about 4000 bytes smaller than the file size for *persist_B.h5* after Session 3 of Scenario B. This is because there are some space savings, in both free space and file metadata (fewer free space sections to track), when the HDF5 object manipulations occur in a single session.

Scenario D: ALL Strategy in Single Session

Session 1: Create file, manipulate objects

In the only session of this scenario, a user creates an HDF5 file named *no_persist_D.h5* using the H5F_FILE_SPACE_ALL strategy. The user then adds four datasets (*dset1, dset2, dset3,* and *dset4*), deletes *dset2*, and adds *dset5* before closing the file.

The file management strategy is the same strategy that was used in Scenario A. The HDF5 objects are manipulated in the same order as they were in Sessions 1-3 of Scenario A.

h5stat –S for *no_persist_D.h5* shows the following:

```
Filename: ./no_persist_D.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: 117976 bytes
Total space: 124832 bytes

The file size for *no_persist_D.h5* is about 6000 bytes smaller than the file size for *no_persist_A.h5* after Session 3 of Scenario A. This is because the HDF5 library was able to reuse some of the free space it was tracking when all of the object manipulations took place in a single session. *no_persist_D.h5*, created in Scenario D, still has a substantial amount of unaccounted space (117976 bytes) – almost 95% of the total file space.

Comparing file space information for *persist_C.h5* (Scenario C) and *no_persist_D.h5* (Scenario D), the file size of *no_persist_D.h5* is a bit smaller. For both files, the library's free-space manager tracks the free space resulting from the deletion of *dset2,* and reuses the free space for the addition of *dset5*. Looking at the size of the file metadata for the two files, the greater amount of file metadata in *persist_C.h5* is due to the extra metadata needed to keep free space information persistent when the file is closed. This demonstrates that using strategy ALL, as was done for *no_persist_D.h5,* has some saving in file space compared to strategy ALL_PERSIST when the HDF5 object manipulation occurs in a single session. The exact amount of space savings will depend on the number and size of HDF5 objects that are added and deleted, as well as on the value of the free-space section threshold and other advanced tuning parameters.

Scenario E: AGGR_VFD Strategy in Single Session

Session 1: Create file, manipulate objects

In the only session of this scenario, a user creates an HDF5 file named aggrvfd_E.*h5* using the H5F_FILE_SPACE_AGGR_VFD strategy. The user then adds four datasets (*dset1, dset2, dset3,* and *dset4*), deletes *dset2*, and adds *dset5* before closing the file.

h5stat –S output shows:

Filename: ./aggrvfd_E.h5
Summary of file space information:
File metadata: 2208 bytes
Raw data: 4640 bytes
Amount/Percent of tracked free space: 0 bytes/0.0%
Unaccounted space: 121936 bytes
Total space: 128784 bytes

Scenario F: VFD Strategy in Single Session

Session 1: Create file, manipulate objects

In the only session of this scenario, a user creates an HDF5 file named vfd_F.*h5* using the VFD strategy. The user then adds four datasets (*dset1, dset2, dset3,* and *dset4*), deletes *dset2*, and adds *dset5* before closing the file.

h5stat –S output shows:

Filename: ./vfd_F.h5
Summary of file space information:
File metadata: 2208 bytes
Raw data: 4640 bytes
Amount/Percent of tracked free space: 0 bytes/0.0%
Unaccounted space: 120272 bytes
Total space: 127120 bytes

Comparison of HDF5 Files from Scenarios A-F after HDF5 Object Manipulation

Scenario / # Sessions	Strategy	File Name	File Size (bytes)	File Metadata (bytes)	Raw Data (bytes)	Tracked Free Space (bytes)	Unaccounted Space (bytes)
A / 3	ALL	no_persist_A.h5	130880	2216	4640	0	124024
B / 3	ALL_PERSIST	persist_B.h5	128921	2427	4640	121854	0
C / 1	ALL_PERSIST	persist_C.h5	124903	2409	4640	117854	0
D / 1	ALL	no_persist_D.h5	124832	2216	4640	0	117976
E / 1	AGGR_VFD	aggrvfd_E.h5	128784	2208	4640	0	121936
F / 1	VFD	vfd_F.h5	127120	2208	4640	0	120272

Files *no_persist_A.h5* and *persist_B.h5,* which were written over three sessions, have the largest file sizes. Since the unused space in *persist_B.h5* is tracked free space, it may be reused in later sessions if more HDF5 objects are added to the file, or if new data values are added to existing dataset objects.

The file sizes of *aggrvfd_E.h5* and *vfd_F.h5* are larger than *persist_C.h5* and *no_persist_D.h5* – files that were also created in a single session. This is because strategies AGGR_VFD and VFD do not track free space, even within a single session, and therefore do not reuse any space that is released as HDF5 objects are manipulated. *aggrvfd_E.h5* is larger than *vfd_F.h5* because bytes in the aggregators' blocks have become unaccounted in the process of managing space. The VFD strategy does not use the aggregators, but allocates space directly from the file driver.

The final Scenarios G and H illustrate that the strategies AGGR_VFD and VFD have the benefit of saving file space when the usage pattern is adding HDF5 objects without deletion. They may also be faster, because no time is spent tracking free space in the file.

Scenario G: AGGR_VFD Strategy in Single Session, no Objects Deleted

Session 1: Create file, add objects

In the only session of this scenario, a user creates an HDF5 file named aggrvfd_G.*h5* using the H5F_FILE_SPACE_AGGR_VFD strategy. The user then adds four datasets (*dset1, dset2, dset3,* and *dset4*) and closes the file.

h5stat –S shows:

Filename: ./aggrvfd_G.h5
Summary of file space information:
  File metadata: 2208 bytes
  Raw data: 120640 bytes
  Amount/Percent of tracked free space: 0 bytes/0.0%
  Unaccounted space: 1936 bytes
Total space: 124784 bytes

Scenario H: VFD Strategy in Single Session, no Objects Deleted

Session 1: Create file, add objects

In the only session of this scenario, a user creates an HDF5 file named vfd_H.*h5* using the H5F_FILE_SPACE_VFD strategy. The user then adds four datasets (*dset1, dset2, dset3,* and *dset4*) and closes the file.

h5stat –S shows:

Filename: ./vfd_H.h5
Summary of file space information:
  File metadata: 2208 bytes
  Raw data: 120640 bytes
  Amount/Percent of tracked free space: 0 bytes/0.0%
  Unaccounted space: 0 bytes
Total space: 122848 bytes

Comparison of HDF5 Files from Scenarios A, B, G, H after HDF5 Objects Added

Scenario / # Sessions	Strategy	File Name	File Size (bytes)	File Metadata (bytes)	Raw Data (bytes)	Tracked Free Space (bytes)	Unaccounted Space (bytes)
A / 2	ALL	no_persist_A.h5	124832	2216	120640	0	1976
B / 2	ALL_PERSIST	persist_B.h5	124885	2319	120640	1854	0
G / 1	AGGR_VFD	aggrvfd_G.h5	124784	1936	120640	0	1936

H / 1	VFD	vfd_H.h5	122848	2208	120640	0	0

The table above shows the file space information for HDF5 files after four datasets have been added. This corresponds to the state of the files after Session 2 in Scenarios A and B.

The *aggrvfd_G.h5* and *vfd_H.h5* files are smaller than *no_persist_A.h5* and *persist_B.h5*. The HDF file *vfd_H.h5,* managed with the VFD strategy, has the smallest size with no tracked free space or unaccounted space. Even though the file *aggrvfd_G.h5* has less saving in file space than *vfd_H.h5*, it will have the benefit of better I/O performance due to the use of aggregators for servicing space allocation requests. Metadata in *aggrvfd_G.h5* will also tend to be more concentrated in contiguous blocks than in *vfd_H.h5*.

The section *Performance Report for File Space Management* provides more information about selecting file space management strategies to optimize access performance.

Particularly aggravated when all within one setting all_one_setting () (~4k, 2k) or add_close_adddelete() (this one is worser?hm…not really diff 2k but I have the bug in this one for VFD) : for #3, then #4
Bug in add_close_adddelete() for VFD