HDF5 File Space Management

Introduced with

HDF5 Release 1.10.0

in

<Date> ???????



<http://www.HDFGroup.org>

Copyright Notice and License Terms for HDF5 (Hierarchical Data Format 5) Software Library and Utilities

HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright 2006-2012 by The HDF Group.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright 1998-2006 by the Board of Trustees of the University of Illinois.

**All rights reserved.**

Redistribution and use in source and binary forms, with or without modification, are permitted for any purpose (including commercial purposes) provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or materials provided with the distribution.
3. In addition, redistributions of modified forms of the source or binary code must carry prominent notices stating that the original code was changed and the date of the change.
4. All publications or advertising materials mentioning features or use of this software are asked, but not required, to acknowledge that it was developed by The HDF Group and by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and credit the contributors.
5. Neither the name of The HDF Group, the name of the University, nor the name of any Contributor may be used to endorse or promote products derived from this software without specific prior written permission from The HDF Group, the University, or the Contributor, respectively.

**DISCLAIMER:** THIS SOFTWARE IS PROVIDED BY THE HDF GROUP AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. In no event shall The HDF Group or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage.

Contributors: National Center for Supercomputing Applications (NCSA) at the University of Illinois, Fortner Software, Unidata Program Center (netCDF), The Independent JPEG Group (JPEG), Jean-loup Gailly and Mark Adler (gzip), and Digital Equipment Corporation (DEC).

Portions of HDF5 were developed with support from the Lawrence Berkeley National Laboratory (LBNL) and the United States Department of Energy under Prime Contract No. DE-AC02-05CH11231.

Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL). The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials:

This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL.

**DISCLAIMER:** This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately- owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Contents

1. Introduction 4

1.1. Definitions and Concepts 4

2. File Space Allocation Mechanisms 6

2.1. Free-Space Manager 6

2.2. Aggregators 6

2.3. Virtual File Driver 6

3. File Space Management Strategies 7

3.1. The All Persist Strategy 8

3.2. The All Strategy 8

3.3. The Aggregator VFD Strategy 9

3.4. The VFD Strategy 9

4. Setting or Changing a File Space Management Strategy 10

4.1. Specifying a Strategy at File Creation with H5Pset\_file\_space 10

4.2. Changing the Strategy with h5repack 11

4.3. Summary of Strategies and Implementation 11

5. Example Scenarios 12

5.1. Scenario A: Default File Space Management Strategy 12

5.1.1. Session 1: Create an Empty File 12

5.1.2. Session 2: Add Datasets 13

5.1.3. Session 3: Add One Dataset and Delete Another 14

5.2. Scenario B: Alternative File Space Management Strategy 15

5.2.1. Session 1: Create an Empty File 15

5.2.2. Session 2: Add Datasets 15

5.2.3. Session 3: Add One Dataset and Delete Another 16

5.3. Scenario C: ALL\_PERSIST Strategy in Single Session 17

5.3.1. Session 1: Create File, Manipulate Objects 17

5.4. Scenario D: ALL Strategy in Single Session 17

5.4.1. Session 1: Create File, Manipulate Objects 17

5.5. Scenario E: AGGR\_VFD Strategy in Single Session 18

5.5.1. Session 1: Create File, Manipulate Objects 18

5.6. Scenario F: VFD Strategy in Single Session 18

5.6.1. Session 1: Create File, Manipulate Objects 18

5.7. Comparison of HDF5 Files from Scenarios A-F after HDF5 Object Manipulation 19

5.8. Scenario G: AGGR\_VFD Strategy in Single Session, no Objects Deleted 19

5.8.1. Session 1: Create File, Add Objects 20

5.9. Scenario H: VFD Strategy in Single Session, no Objects Deleted 20

5.9.1. Session 1: Create File, Add Objects 20

5.10. Comparison of HDF5 Files from Scenarios A, B, G, and H after HDF5 Objects were Added 21

# Introduction

The space within an HDF5 file is called its *file space*. When a user first creates an HDF5 file, the HDF5 Library (also known as the library) immediately allocates space to store 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 library allocates space to store data values and 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 library manages this free space. The library’s *file space management* activities encompass both the allocation of space and the management of free space.

The library has a variety of mechanisms that allow it to implement several different *file space management strategies*. Users can select a strategy when they create an HDF5 file. Depending on the file’s usage patterns, one strategy may be better than the others. Users of HDF5 files that have large datasets added and removed on a regular basis might prefer one strategy while users of HDF5 files that are fairly static might prefer a different strategy.

This document describes the available file space allocation mechanisms and strategies, the tools - API and command line - that are available to set or change a strategy, and how the file space management strategies affect the file size and access time for various HDF5 file usage patterns.

## Definitions and Concepts

The following are some terms and concepts used in this document.

Session

A session is the time between when a file is opened and closed.

Tracked Free Space and Unaccounted Space

Space within an HDF5 file is freed up when an object is removed from the file. The freed up space is monitored while the file is open and is reported as tracked free space. Depending on the file space management strategy that was chosen for the file, the tracked free space may be tracked after the file is closed (after the end of the current session). If the tracked free space is not tracked after the file is closed, then the free space will be considered unaccounted space the next time the file is open.

Tracked free space is sometimes referred to as tracked space.

Tracked free space and unaccounted space are reported in the output of the command line h5stat -S.

Tracked free space and unaccounted space can be reclaimed with the h5repack tool.

VFD

VFD is short for Virtual File Driver.

Raw Data

Raw data are the data values in HDF5 dataset objects. In a file that holds weather data, the raw data might include temperatures at different locations and at a variety of times during the day.

File Metadata

File metadata is information the library uses to describe the HDF5 file and to identify its associated objects. One example is the file space management strategy used by a file. The strategy is stored in file metadata. For more information, see the [“HDF5 Metadata”](http://www.hdfgroup.org/HDF5/doc/Advanced/HDF5_Metadata/index.html) paper.

# File Space Allocation Mechanisms

The HDF5 Library has three different mechanisms for allocating space to store file metadata and raw data. These are described in the sections below.

## Free-Space Manager

The HDF5 Library’s free-space manager tracks 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 manager searches 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 has 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. If at some point in the future that space is freed, then the free-space manager would be in charge of the space and not the aggregator. In other words, the freed space would not revert back to the aggregator. Unallocated bytes in the block continue to be managed by the aggregator.

If an aggregator does not have enough space in its block to fulfill a request, it will then request a new block of contiguous bytes from the library. Any unallocated space from the old block will become free space.

## Virtual File Driver

The HDF5 Library’s virtual file driver (VFD) interface dispatches requests for additional space to the allocation routine of the file driver associated with an HDF5 file. For example, if the POSIX file driver, H5FD\_SEC2, 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. For more information on VFDs, see the “Alternate File Storage Layouts and Low-level File Drivers” section in “The HDF5 File” chapter in the *HDF5 User’s Guide*.

# File Space Management Strategies

The file space allocation mechanisms described above can be used to implement a variety of file space management strategies. The strategies differ in two main ways: when the library will track free space and how many of the mechanisms the library will use to allocate space for file metadata and raw data. The strategies are listed in the table below and are described in more detail in the sections following the table.

| Table 1. File space management strategies |
| --- |
| Strategy Name | The strategy might be useful under these conditions: | Implementation Comments |
| **All Persist** | Use with files where raw data and metadata are added and removed frequently and where the files are opened and closed frequently. Maximizes the use of space in a file over any number of sessions. | * Uses all of the file space allocation mechanisms
* Tracks file free space across sessions
 |
| **All** | Use with files where raw data and metadata are added and removed frequently. Maximizes the use of space in a file during a single session. | * Uses all of the file space allocation mechanisms
* Tracks file free space only in the current session
 |
| **Aggregator VFD** | Use with files where small datasets might be added and where few if any datasets are removed. Adding small datasets means the library can take advantage of the aggregators. Maximizes rate at which small datasets are written to the file. | * Uses the aggregator and VFD mechanisms
* Never tracks free space
 |
| **VFD Only** | Use with files where large amounts of raw data are added and where few if any datasets are removed. Maximizes rate at which data is written to the file. | * Uses only the VFD mechanism
* Never tracks free space
 |

For more information on implementing one of these strategies, see the “Setting or Changing a Strategy” section on page 10.

## The All Persist Strategy

The aim of the All Persist strategy is to maximize the use of space within an HDF5 file over a number of sessions.

With the All Persist strategy, the HDF5 Library’s free-space manager tracks 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 is reloaded when the file is re-opened. The tracked free space information **persist**s across HDF5 file sessions, and the free-space manager remains 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 manager. 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 All Persist strategy offers every possible opportunity for reusing free space. The HDF5 file will contain extra file metadata for the tracked free space. The 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.

This strategy was added to the library in HDF5 release 1.10.0.

## The All Strategy

The aim of the All strategy is to maximize the use of space within an HDF5 file during a single session.

With this strategy, the HDF5 Library’s free-space manager tracks the free space that results from manipulating HDF5 objects in an HDF5 file. The free-space manager is 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 All Persist strategy, the library will try all of the mechanisms for allocating space. When space is needed for file metadata or raw data, the HDF5 Library first requests space from the free-space manager. 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 All strategy allows free space incurred in the current session to be reused in the current session. Since the free space is not saved after the file is closed, no extra file metadata is needed to keep track of the free space.

The All strategy is the HDF5 Library’s default file space management strategy. Prior to HDF5 Release 1.10.0, the All strategy was the only file space management strategy directly supported by the library.

## The Aggregator VFD Strategy

The aim of the Aggregator VFD strategy is to deliver better access performance in situations where many small datasets might be written to a file. Space requests to hold small datasets would go to the aggregators. Not having to find an appropriate size among the tracked free space will improve access performance.

With the Aggregator VFD strategy, the 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 will 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 use the **aggr**egators and VFD mechanisms to fill requests for file space.

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.

This strategy was added to the library in HDF5 release 1.10.0.

## The VFD Strategy

The aim of the VFD strategy is to maximize file writing performance. 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 objects.

With the VFD 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 VFD.

This strategy was added to the library in HDF5 release 1.10.0.

# Setting or Changing a File Space Management Strategy

The purpose of this chapter is to describe how to set or change the file space management strategy for a file. The file space management strategy that a file operates under is set when the file is created. The strategy can be changed when the h5repack command line utility program is run.

## Specifying a Strategy at File Creation with H5Pset\_file\_space

To set the strategy when an HDF5 file is created, use the [H5Pset\_file\_space](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5P.html#Property-SetFileSpace) routine.

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 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. Valid values for this parameter are also shown in Table 2 on page 11. The third parameter, threshold*,* is the free-space section threshold used by the library’s free-space manager.

The library provides a companion routine, H5Pget\_file\_space, that retrieves the file space management information for an HDF5 file. See the entry in the [*HDF5 Reference Manual*](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5Front.html).

The following code sample shows how to create an empty HDF5 file, *persist.h5,* with the All Persist file space management strategy:

|  |
| --- |
| /\* 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);/\* Close the file \*/H5Fclose(fid); |
| Example 1. Use H5Pset\_file\_space to set the file space management strategy |

To see what a file’s strategy is, use the h5dump command line utility program.

For more information on creating a file, see the entry in the *HDF5 Reference Manual* for H5Fcreate and “The HDF5 File” chapter in the *HDF5 User’s Guide*.

## Changing the Strategy with h5repack

It is not always possible to know in advance how a file will be used. Running the h5stat –S command line utility may show that a given file has a large amount of unaccounted space. This would indicate that the file space management strategy might need to be changed. The [h5repack](http://www.hdfgroup.org/HDF5/doc/RM/Tools.html#Tools-Repack) command line utility program can be used to reclaim the unaccounted space, and it can also be used to change the strategy.

The –S or –fs\_strategy options can be used with h5repack to specify the strategy to be used with the output file. The example below shows h5repack specifying the VFD strategy. The input file name is no\_persist\_A.h5, and the output file name is no\_persist\_outvfd.h5.

|  |
| --- |
| h5repack –S VFD no\_persist\_A.h5 no\_persist\_outvfd.h5 |
| Example 2. Using h5repack to change the file space management strategy |

The valid values for the –S and –fs\_strategy options are listed in the table below.

| Table 2. Strategies and Values |
| --- |
| Strategy | h5repack Value | H5Pset\_file\_space Value |
| All Persist | ALL\_PERSIST | H5F\_FILE\_SPACE\_ALL\_PERSIST |
| All | ALL | H5F\_FILE\_SPACE\_ALL |
| Aggregator VFD | AGGR\_VFD | H5F\_FILE\_SPACE\_AGGR\_VFD |
| VFD | SPACE\_VFD | H5F\_FILE\_SPACE\_VFD |

## Summary of Strategies and Implementation

The file space management strategies and file allocation mechanisms that have been discussed above are summarized in the table below.

| **Table 3. Summary of file space management strategies and mechanisms** |
| --- |
| **Strategy** | **H5Pset\_file\_space Value****h5repack Value** | **Track Free Space** | **Allocate Space Using** |
| **Across Multiple Sessions** | **Within a Single Session** | **Free-space Manager** | **Aggregators** | **VFD** |
| All Persist | H5F\_FILE\_SPACE\_ALL\_PERSISTALL\_PERSIST | Y | Y | Y | Y | Y |
| All | H5F\_FILE\_SPACE\_ALLALL | N | Y | Y | Y | Y |
| Aggregator VFD | H5F\_FILE\_SPACE\_AGGR\_VFDAGGR\_VFD | N | N | N | Y | Y |
| VFD | H5F\_FILE\_SPACE\_VFDSPACE\_VFD | N | N | N | N | Y |

# Example Scenarios

## 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.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 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.

## 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.

## 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, and H after HDF5 Objects were 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.