RFC: HDF5 File Space Management: Paged Aggregation

Vailin Choi

Quincey Koziol

John Mainzer

The current HDF5 file space allocation accumulates small pieces of metadata and raw data in aggregator blocks, which are not page aligned and vary widely in sizes. To provide efficient paged access to these small pieces of metadata and raw data, we propose a file space management mechanism we call Paged Aggregation.

1) Table of Contents:

Table of Contents: 2

1 Introduction 4

2 Overview of Current File Space Management 4

2.1 Public Routines 5

2.2 File Space Info Message 6

3 Overview of Paged Aggregation 7

4 Public routines 8

4.1 Additions to the API 8

4.2 Modifications to the API 9

5 File Space Info message 9

6 Cycle of operation when allocating file space 11

6.1 H5F_FSPACE_STRATEGY_FSM_AGGR 11

6.2 H5F_FSPACE_STRATEGY_PAGE 11

6.3 H5F_FSPACE_STRATEGY_AGGR 12

6.4 H5F_FSPACE_STRATEGY_NONE 12

7 Cycle of operation when freeing file space 12

7.1 H5F_FSPACE_STRATEGY_FSM_AGGR 12

7.2 H5F_FSPACE_STRATEGY_PAGE 13

7.3 H5F_FSPACE_STRATEGY_AGGR 14

7.4 H5F_FSPACE_STRATEGY_NONE 14

8 Cycle of operation when shrinking file space 14

8.1 H5F_FSPACE_STRATEGY_FSM_AGGR 15

8.2 H5F_FSPACE_STRATEGY_PAGE 16

8.3 H5F_FSPACE_STRATEGY_AGGR 16

8.4 H5F_FSPACE_STRATEGY_NONE 16

9 Cycle of operation when extending file space 16

9.1 H5F_FSPACE_STRATEGY_FSM_AGGR 16

9.1.1 Extending the section at EOA 16

9.1.2 Extending the section into either the metadata or raw data aggregator 16

9.1.3 Extending the section into a free-space section 17

9.2 H5F_FSPACE_STRATEGY_PAGE 17

9.2.1 Extending the section at EOA 17

9.2.2 Extending the section into a free-space section 17

9.2.3 Extending the section into the page-end threshold (may delete this optimization) 17

9.3 H5F_FSPACE_STRATEGY_AGGR 18

9.3.1 Extending the section at EOA 18

9.3.2 Extending the section into either the metadata or raw data aggregator 18

9.4 H5F_FSPACE_STRATEGY_NONE 18
9.4.1 Extending the section at EOA 18
10 Shutting Down Free-Space Managers on File Close 18
10.1 H5F_FSPACE_STRATEGY_FSM_AGGR 19
10.2 H5F_FSPACE_STRATEGY_PAGE 20
10.3 H5F_FSPACE_STRATEGY_AGGR 20
10.4 H5F_FSPACE_STRATEGY_NONE 20
11 Floating Free-Space Managers When Reopening Files 21
11.1 H5F_FSPACE_STRATEGY_FSM_AGGR 21
11.2 H5F_FSPACE_STRATEGY_PAGE 21
11.3 H5F_FSPACE_STRATEGY_AGGR 21
11.4 H5F_FSPACE_STRATEGY_NONE 21
12 Tools 22
12.1 h5dump 22
12.2 h5stat 22
12.3 h5repack 22
13 Testing 22
14 Documentation 23
15 Backward/Forward Compatibility 24
16 Limitations 24
17 A New Solution For Shutting Down Free-space Managers 24
17.1 H5FS_sinfo_unlock() 25
17.2 Cache Callbacks For Free-space Manager 26
17.3 Shutting Down Free-Space Managers on File Close 27
17.3.1 H5F_FSPACE_STRATEGY_FSM_AGGR and H5F_FSPACE_STRATEGY_PAGE 27
17.3.2 H5F_FSPACE_STRATEGY_AGGR and H5F_FSPACE_STRATEGY_NONE 28
17.4 Allocation Of Cache Image At File Close 28
17.5 Clean up 28
17.6 Backward and Forward Compatibility 28
17.7 Testing 29
17.8 Time estimate 29
18 Future Issues 29
Acknowledgements 29
Revision History 29

Introduction

This document addresses the changes to the HDF5 library needed to implement the file space handling mechanism we call paged aggregation. This mechanism aggregates small metadata and raw data allocations into constant-sized well-aligned pages, which are suitable for page caching. Paged aggregation together with the *Page Buffering* feature should allow more efficient I/O accesses.

The detailed rationale behind this proposed mechanism is described in the *RFC: HDF5 File Space Allocation and Aggregation*. The *Page Buffering* feature is described in the *RFC: Page Buffering*.

The new features discussed in this RFC have been implemented and released in HDF5 1.10.1. Until such time as a document discussing file free space management in HDF5 is written, this document should be updated to reflect any API or algorithmic changes in the features discussed.

Overview of Current File Space Management

At present, the HDF5 library uses three mechanisms to manage space in an HDF5 file. They are:

- Free-space managers

  They track sections of the file of various sizes that are currently free or unallocated. Each free-space manager corresponds to a file space type. There are two main groups of file space types: metadata and raw data. Metadata is further divided into five types: superblock, B-tree, global heap, local heap, and object header.

- Aggregators

  The library manages two aggregators, one for metadata and one for raw data (but note that metadata aggregation is turned off In the split and multi file drivers). The aggregator is a contiguous block of free space in the file. The size of each aggregator is tunable via public routines *H5Pset_meta_block_size* and *H5Pset_small_data_block_size* respectively.

  The current implementation of the aggregator blocks is not page-aligned, and the sizes may vary from the original specified block size.

- Virtual file drivers

  The library's virtual file driver interface dispatches requests for additional space to the allocation routine of the current file driver, which manages I/O to the file. For example, if the *sec2* file driver is being used, its allocation routine will increase the size of the file to service requests.

For files with contiguous address space, the default behavior is to have one free-space manager to handle metadata and one free-space manager to handle raw data.

For files with non-contiguous address space, it is possible to have up to one free-space manager for

each of the six file space types: raw data and five types of metadata.

Based on these mechanisms, there are four file space handling strategies available to users for managing file space:

1. H5F_FILE_SPACE_ALL
    - Mechanisms used: free-space managers, aggregators, and virtual file drivers
    - Free space managed by the free-space managers is discarded at file close, and is thus not persistent across file close/open cycles.
    - This strategy is the library default
2. H5F_FILE_SPACE_ALL_PERSIST
    - Mechanisms used: free-space managers, aggregators, and virtual file drivers
    - Free space managed by the free-space managers is saved at file close, and thus persists across file close/open cycles
3. H5F_FILE_SPACE_AGGR_VFD
    - Mechanisms used: aggregators and virtual file drivers
    - Free-space is not retained across file close/open cycles
4. H5F_FILE_SPACE_VFD
    - Mechanisms used: virtual file drivers
    - Free-space is not retained across file close/open cycles

Please refer to the *HDF5 File Space Management* document for full details.

Public Routines

- *herr_t H5Pset_file_space(hid_t fcpl, H5F_file_space_type_t strategy, hsize_t threshold)*
    - Set the file space handling *strategy* and free-space section *threshold* in the file creation property list *fcpl*; the setting cannot be changed for the life of the file.
    - *strategy* is the file space handling strategy defined as:

```
typedef enum H5F_file_space_type_t {
  H5F_FILE_SPACE_DEFAULT=0,           /* Not Used?? */
   H5F_FILE_SPACE_ALL_PERSIST=1, /* Persistent FSM, aggregators, VFD
*/
   H5F_FILE_SPACE_ALL=2,      /* Non-persistent FSM, aggregators, VFD
*/
  H5F_FILE_SPACE_AGGR_VFD=3, /* Aggregators, VFD */
  H5F_FILE_SPACE_VFD=4,     /* VFD */
  H5F_FILE_SPACE_NTYPES
} H5F_file_space_type_t;
```

    - *threshold* is the smallest free-space section size that the free-space manager will track.

- *herr_t H5Pget_file space(hid_t fcpl, H5F_fspace_strategy_t *strategy, hsize_t *threshold)*
    - Retrieve the file space handling strategy and free-space threshold value in the parameters *strategy* and *threshold* respectively.
    - Return the library default value as follows when not set via *H5Pset_file_space*:
        - *strategy*— H5F_FILE_SPACE_ALL
        - *threshold*—1

File Space Info Message

The library's default setting for handling file space is:

- File space handling strategy is H5F_FILE_SPACE_ALL
- Free-space section threshold is 1

If the user sets file space info that deviates from any of the above, the library will create the *File Space Info* message to store the non-default setting and the message is stored in the superblock extension.

Layout: Version 0 File Space Info message

Byte	**Byte**	**Byte**	**Byte**
Version	File space strategy	*This exists to align table nicely.*	
Free-space section threshold^{L}			
Addresses^{O} of free-space managers for the six file space types: H5FD_MEM_SUPER, H5FD_MEM_BTREE, H5FD_MEM_DRAW, H5FD_MEM_GHEAP, H5FD_MEM_LHEAP, H5FD_MEM_OHDR			

Field Name	**Description**
Version	The version number is used to indicate the format of the message. The value is 0.
File space strategy	This is the file space strategy used to manage file space: • H5F_FILE_SPACE_ALL • H5F_FILE_SPACE_ALL_PERSIST • H5F_FILE_SPACE_AGGR_VFD • H5F_FILE_SPACE_VFD
Free-space section threshold	The smallest free-space section size that the free-space manager will track.
Addresses of free-space managers	The addresses of free-space managers for the six file space types

	when strategy is H5F_FILE_SPACE_ALL_PERSIST.

Overview of Paged Aggregation

The goal of paged aggregation is to accumulate metadata and raw data into well-aligned pages, which we call file space pages. The library defines a default file space page size but user can set the page size via a new public routine, *H5Pset_file_space_page_size*.

The free-space manager mechanism is modified to handle paged aggregation as follows:

- Small-sized free-space manager:
  - Track free-space sections whose size is < file space page size
  - Satisfy requests either from existing free space if available; if not, request a page from large-sized manager:
    - Returned space: no page alignment constraint; cannot cross page boundary
  - Shrink a free-space section:
    - Never – but release sections of size equal to the page size to the large-sized free-space manager, which may do so.
  - Merge two free-space sections:
    - When the two sections adjoin and they are on the same page
    - When the merged section is equal to page size, return to the large-sized free-space manager
- Large-sized free-space manager
  - Track free-space sections whose size is >= file space page size, along with smaller, misaligned sections left over from allocations >= file space page size.
  - Satisfy requests either from existing free space if available; if not, request space from virtual file driver
    - Returned space: page aligned; can cross page boundary
  - Shrink a free-space section:
    - When the section ends at EOA and the section is >= page size
    - To keep EOA at page boundary: shrink only full-sized pages but retain partial pages in the manager
  - Merge two free-space sections when they adjoin

For files with contiguous address space, the default behavior is to have two small-sized free-space managers (one for metadata and one for raw data) and one large-sized free-space manager for

generic data (can be metadata or raw data).

For files with non-contiguous address space, it is possible to have up to 6 small-sized free-space managers and 6 large-sized free-space managers. They correspond to the six file space types: raw data and five types of metadata.

We propose four file space-handling strategies available to users for handling file space:

1. H5F_FSPACE_STRATEGY_FSM_AGGR:
    - Mechanisms used: free-space managers, aggregators, and virtual file drivers
    - This is the library default, and is the same as the current library default
2. H5F_FSPACE_STRATEGY_PAGE:
    - Mechanisms used: free-space managers with embedded paged aggregation and virtual file drivers
3. H5F_FSPACE_STRATEGY_AGGR:
    - Mechanisms used: aggregators and virtual file drivers
4. H5F_FSPACE_STRATEGY_NONE:
    - Mechanisms used: virtual file driver

For all the above strategies, the default is to not retain free-space across file close/open cycles. The user can use the public routine *H5Pset_file_space_strategy()* to request persistent file free-space, which is retained across file close/open cycles. Requests for persistent file free space management are not applicable to the last two strategies, as they do not use file free space managers.

Public routines

Additions to the API

- *herr_t H5Pset_file_space_page_size(hid_t fcpl, hsize_t fsp_size)*
    - Set the file space page size *fsp_size* for paged aggregation in the file creation property list *fcpl*; the size set via this routine cannot be changed for the life of the file.
    - *fsp_size* has a minimum size of 512. Setting value less than 512 will return an error.
    - The library default value for file space page size when not set is 4096.
    - The maximum page size is 1 GB. Attempts to set page size larger than this value will result in an error.
- *herr_t H5Pget_file_space_page_size(hid_t fcpl, hsize_t *fsp_size)*
    - Retrieve the file space page size for paged aggregation in the parameter *fsp_size* from the file creation property list *fcpl*.
    - Return the library default 4KB (4096) in *fsp_size* If the page size is not set via

H5Pset_file_space_page_size.

- *herr_t H5Pset_file_space_strategy(hid_t fcpl, H5F_fspace_strategy_t strategy, hbool_t persist, hsize_t threshold)*
    - Set the file space handling *strategy*, whether free space should *persist across file close/open cycles*, and free-space section *threshold* in the file creation property list *fcpl*; the setting cannot be changed for the life of the file.
    - *strategy* is the file space handling strategy defined as:

```
typedef enum H5F_fspace_strategy_t {
      H5F_FSPACE_STRATEGY_FSM_AGGR = 0, /* FSM, Aggregators, VFD
      */
      H5F_FSPACE_STRATEGY_PAGE = 1     /* Paged FSM, VFD */
      H5F_FSPACE_STRATEGY_AGGR = 2     /* Aggregators, VFD */
      H5F_FSPACE_STRATEGY_NONE = 3,     /* VFD */
      H5F_FSPACE_STRATEGY_NTYPES
} H5F_fspace_strategy_t;
```

    - *persist* indicates whether file free space should be retained across file close/open cycles.
    - *threshold* is the smallest free-space section size that the free-space manager will track.
    - As H5F_FSPACE_STRATEGY_AGGR and H5F_FSPACE_STRATEGY_NONE strategies do not use free-space managers, the *persist* and *threshold* settings will be ignored for those strategies.
- *herr_t H5Pget_file space strategy(hid_t fcpl, H5F_fspace_strategy_t *strategy, hbool_t *persist, hsize_t *threshold)*
    - Retrieve the file space handling strategy, free-space persistence, and threshold value in the parameters *strategy*, *persist* and *threshold* respectively.
    - Return the library default value as follows when not set via *H5Pset_file_space_strategy*:
        - *strategy*— H5F_FSPACE_STRATEGY_FSM_AGGR
        - *persist*—FALSE
        - *threshold*—1

Modifications to the API

- *herr_t H5Pset_alignment(hid_t plist, hsize_t threshold, hsize_t alignment)*
    - Add the description to the reference manual entry that if H5F_FSPACE_STRATEGY_PAGE strategy is used, the alignment set via this routine is ignored.

File Space Info message

The library's default setting for handling file space is:

- File space strategy is H5F_FSPACE_STRATEGY_FSM_AGGR
- Not persisting free-space
- Free-space threshold is 1
- File space page size is 4096

If the user sets file space info that deviates from any of the above, the library will create the *File Space Info* message to store the non-default setting and the message is stored in the superblock extension.

Layout: Version 1 File Space Info message

Byte	**Byte**	**Byte**	**Byte**
Version	File space strategy	Persisting free-space	*This exists to align table nicely.*
Free-space section threshold[L]			
File space page size[L]			
Page-end metadata threshold		*This exists to align table nicely.*	
EOA[O]			
Addresses[O] of small-sized free-space managers for the six file space types: H5FD_MEM_SUPER, H5FD_MEM_BTREE, H5FD_MEM_DRAW, H5FD_MEM_GHEAP, H5FD_MEM_LHEAP, H5FD_MEM_OHDR			
Addresses[O] of large-sized free-space managers for the six file space types: H5FD_MEM_SUPER, H5FD_MEM_BTREE, H5FD_MEM_DRAW, H5FD_MEM_GHEAP, H5FD_MEM_LHEAP, H5FD_MEM_OHDR			

[L]This item in the table is the *size of lengths* as defined in the superblock.

[O]This item in the table is the *size of offsets* as defined in the superblock.

Fields: File Space Info Message

Field Name	**Description**
Version	The version number is used to indicate the format of the message. The value is 1.
File space strategy	This is the file space strategy used to manage file space: • H5F_FSPACE_STRATEGY_FSM_AGGR • H5F_FSPACE_STRATEGY_PAGE • H5F_FSPACE_STRATEGY_AGGR • H5F_FSPACE_STRATEGY_NONE

Persisting free-space	True or False in persisting free-space
Free-space section threshold	The smallest free-space section size that the free-space manager will track. This optimization is currently disabled, and may be deleted from the code.
File space page size	The file space page size, which is used when paged aggregation is enabled.
Page-end metadata threshold	The smallest free-space section size at the end of a page that the free-space manager will track. This is used when paged aggregation is enabled.
EOA	The EOA before the allocation of free-space header and section info for the self-referential* free-space managers when the library is persisting free-space.
Addresses of free-space managers	The addresses of small-sized free-space managers for the six file space types when persisting free-space.
Addresses of free-space managers	The addresses of large-sized free-space managers for the six file space types when persisting free-space and when paged aggregation strategy is enabled.

*Please see description of self-referential free-space managers in ***Shutting down free-space managers on file close*** (see section 10 below).

Cycle of operation when allocating file space

The library calls the routine *H5M_alloc()* to request file space when creating objects for the HDF5 file. The mechanisms used to fulfill the request will depend on the file space strategy with the cycle of operation described below.

H5F_FSPACE_STRATEGY_FSM_AGGR

- The library will request space from the free-space manager depending on the file space type.
- If the request can be satisfied by the appropriate free space manager, the library will do so and return the address to the caller.
- If the request cannot be satisfied by the appropriate free space manager, the library will request space from either the metadata or raw data aggregator depending on the file space type.
- If the request can be satisfied by the appropriate aggregator, the library will do so and return the address to the caller.
- If the request cannot be satisfied by the appropriate aggregator, the library will request space

from the virtual file driver and return the address to the caller.

H5F_FSPACE_STRATEGY_PAGE

- The library will request space from the free-space manager depending on the request size:
  - For request size < page size, request is sent to the small-sized free-space manager for the requested file space type.
  - For request size >= page size, request is sent to the large-sized free-space manager for the requested file space type.
- If the request can be satisfied from existing free space, the free space manager will do so, and return the address to the caller.
- If the request cannot be satisfied from existing free space:
  - The small-sized free space manager will request a page of file space from the large-sized free-space manager for the file space type, satisfy the file space request from the newly allocated page, and return the address of the new space to the caller.
  - The large-size free space manager will request sufficient pages from the virtual file driver, satisfy the request out of the newly allocated space, and return the address of the new space to the caller.

    Since EOA must always be on a page boundary, the large-sized free space manager will retain any mis-aligned left over space so that it can be merged with the allocation that caused it when and if that allocation is released.

H5F_FSPACE_STRATEGY_AGGR

- The library will request space from either the metadata or raw data aggregator depending on the file space type.
- If the request can be satisified by the appropriate aggregator, the library will do so and return the address to the caller.
- If the request cannot be satisfied by the appropriate aggregator, the library will request space from the virtual file driver and return the address to the caller.

H5F_FSPACE_STRATEGY_NONE

- The library will request space from the virtual file driver and return the address to the caller.

Cycle of operation when freeing file space

The library calls the routine *H5MF_xfree()* when releasing file space with the cycle of operation described below.

H5F_FSPACE_STRATEGY_FSM_AGGR

- If there is no existing free-space manager for the file space type, the library will attempt shrinking action as described in ***Cycle of operation when shrinking file space*** (see section 8 below).
- If the shrinking action succeeds, return to the caller.
- If the section size is less than the free-space threshold, drop the section on the floor and return to the caller.
- If the section size is >= free-space threshold, start up the free-space manager for the file space type and perform the following:
    - Try merging the section with existing sections in the free-space manager if they adjoin.
    - Try shrinking action as described in ***Cycle of operation when shrinking file space*** (see section 8 below).
    - If the section is not merged away or shrunk, add the section to the manager.
    - Return to the caller.

H5F_FSPACE_STRATEGY_PAGE

- If there is no existing free-space manager for the file space type, the library will attempt shrinking action as described in ***Cycle of operation when shrinking file space*** (see section 8 below).
- If the shrinking action succeeds, return to the caller.
- If the section size is less than the free-space threshold, drop the section and return to the caller.
- If the section size is >= free-space threshold, start up the small-sized or large-sized free-space manager for the file space type and perform the following (Note: we are experimenting with deleting the page-end threshold region optimization, as we question whether the added complexity is worth the gain.):
    - For a small-sized section:
        - If the section resides in the page-end threshold region, drop the section and return to the caller.
        - If the section ends within the page-end threshold region, increase section size to end of page and continue.
    - Try merging the section with existing sections:
        - For a small section: if they adjoin and the merged section does not cross page boundary.
        - For a large section: if they adjoin.
- Try shrinking action as described in ***Cycle of operation when shrinking file space*** (see section 8 below)

- If the section is not merged away or shrunk, add the section to the corresponding manager.
- Return to the caller.

H5F_FSPACE_STRATEGY_AGGR

- The library will attempt shrinking action as described in ***Cycle of operation when shrinking file space*** (see section 8 below)**.**
- If the shrinking action succeeds, return to the caller.
- Otherwise drop the section on the floor and return to the caller.

H5F_FSPACE_STRATEGY_NONE

- The library will attempt shrinking action as described in ***Cycle of operation when shrinking file space*** (see section 8 below)**.**
- If the shrinking action succeeds, return to the caller.
- Otherwise drop the section on the floor and return to the caller.

Cycle of operation when shrinking file space

The library performs shrinking actions via the *can_shrink()* and *shrink()* callbacks according to the free-space section class. There are 3 kinds of section classes:

- *simple* section class:
    - Used by the following strategies:
        - H5F_FSPACE_STRATEGY_FSM_AGGR
        - H5F_FSPACE_STRATEGY_AGGR
        - H5F_FSPACE_STRATEGY_NONE
    - Attempt 2 kinds of shrinking action:
        - Shrink via EOA: shrink the file at EOA by section size
        - Shrink via aggregator:
            - Try merging the section into the aggregator
            - Try absorbing the aggregator into the section
            - Applicable only for H5F_FSPACE_STRATEGY_FSM_AGGR and H5F_FSPACE_STRATEGY_AGGR strategies
- *small* section class:
    - The *can_shrink*() and *shrink*() callbacks are not defined for the small section class. However, the small free space managers will release complete pages to the appropriate large free space manager when they become free.

- *large* section class:
    - Used by the H5F_FSPACE_STRATEGY_PAGE strategy for the large-sized manager.
    - Attempt 1 kind of shrinking action:
        - Shrink via EOA: shrink the file at EOA when the section size is >= file space page size.

The *can_shrink()* callback determines the shrinking action that can be done and the *shrink()* callback actually performs the shrinking action allowed. The following data structure is used to pass information to/from the client and the callbacks:

```
typedef struct H5MF_sect_ud_t {
    /* Down */
    H5F_t *f;                   /* Pointer to file to operate on */
    hid_t dxpl_id;              /* DXPL for VFD operations */
    H5FD_mem_t alloc_type;      /* File space type */
    hbool_t allow_sect_absorb; /* See below */
    hbool_t allow_eoa_shrink_only; /* See below */

    /* Up */
    H5MF_shrink_type_t shrink;  /* See below */
    H5F_blk_aggr_t *aggr;       /* Aggregator block to operate on */
} H5MF_sect_ud_t;
```

- *allow_sect_absorb:* whether the section is allowed to absorb the aggregator
    - TRUE: allow the section to absorb the aggregator
    - FALSE: does not allow the section to absorb the aggregator i.e. only allow merging the section into the aggregator; this setting is mainly used when starting up the free-space manager is not desirable
- *allow_eoa_shrink_only:* whether only *shrink via EOA* is allowed
    - TRUE: allow only *shrink via EOA*; this setting is mainly used when shutting down the free-space managers on file closing
    - FALSE: no restriction
- *shrink:* type of shrink operation to perform as determined by *can_shrink()* callback
    - **H5MF_SHRINK_EOA**: section should shrink the EOA value
    - **H5MF_SHRINK_AGGR_ABSORB_SECT**: section should merge into the aggregator
    - **H5MF_SHRINK_SECT_ABSORB_AGGR**: aggregator should merge into the section

H5F_FSPACE_STRATEGY_FSM_AGGR

- If the section ends at EOA, shrink the file by section size and return to the caller.
- Otherwise, try *shrink via aggregator*:
    - If the section adjoins the beginning or end of the aggregator, merge the section into the

aggregator or absorb the aggregator into the section.

H5F_FSPACE_STRATEGY_PAGE

- For a small section at EOA or elsewhere, if the section is page aligned, and the section size is equal to file space page size, release the section to the large-size free-space manager. Note that this is part of the normal operation of the small free space managers, and is not triggered via the *can_shrink*() and *shrink*() callbacks.
- For a large section at EOA, shrink the file if the section size is >= file space page size. Note that only full-sized pages are shrunk with partial page put into the large-sized manager to keep EOA at page boundary.

H5F_FSPACE_STRATEGY_AGGR

- If the section ends at EOA, shrink the file by section size and return to the caller.
- Otherwise, try *shrink via aggregator*:
    - If the section adjoins the beginning or end of the aggregator, merge the section into the aggregator.

H5F_FSPACE_STRATEGY_NONE

- If the section ends at EOA, shrink the file by section size and return to the caller.

Cycle of operation when extending file space

Under appropriate circumstances, the library will attempt to extend an existing section of allocated memory via the *H5MF_try_extend()* call. The cycle of operation for this action is described below.

H5F_FSPACE_STRATEGY_FSM_AGGR

Extending the section at EOA

- If the section ends at EOA:
    - Extend the file by extra requested and return *extended* to the caller.
- Otherwise continue with 9.1.2.

Extending the section into either the metadata or raw data aggregator

- If the section adjoins the beginning of the aggregator:
    - If the aggregator is not at EOA:
        - If the aggregator has enough space to fulfill the extra requested:
            - Extend the section into the aggregator and return *extended* to the caller.

        - Otherwise continue with 9.1.3.
    - If the aggregator is at EOA:
        - If the extra requested is below the extension percentage threshold:
            - Extend the section into the aggregator and return *extended* to the caller.
        - If the extra requested is above the extension percentage threshold:
            - Increase the size of the aggregator, extend the section into the aggregator and return *extended* to the caller.
- Otherwise continue with 9.1.3

Extending the section into a free-space section

- If the section adjoins an existing free-space section in the manager with size large enough to fulfill the extra requested:
    - Extend the section into the adjoined free-space section and return *extended* to the caller.
- Otherwise return *not extended* to the caller.

H5F_FSPACE_STRATEGY_PAGE

Extending the section at EOA

- If the section ends at EOA:
    - For a small-sized section:
        - Return *not extended* to the caller.
    - For a large-sized block:
        - Extend the file by extra requested plus misaligned fragment to keep the EOA at page boundary; put the misaligned fragment into the large-sized free-space manager.
    - Return *extended* to the caller.
- Otherwise continue with 9.2.2.

Extending the section into a free-space section

- If the section adjoins an existing free-space section in the manager with size large enough to fulfill the extra requested:
    - Extend the section into the adjoined free-space section and return *extended* to the caller.
- Otherwise continue with 9.2.3.

Extending the section into the page-end threshold (may delete this optimization)

- For a metadata section which ends in the page-end threshold region and the threshold size can fulfill the extra requested:
    - Extend into the threshold region and return *extended* to the caller.
- Otherwise return *not extended* to the caller.

H5F_FSPACE_STRATEGY_AGGR

Extending the section at EOA

- If the section ends at EOA:
    - Extend the file by extra requested and return *extended* to the caller.
- Otherwise continue with 9.3.2.

Extending the section into either the metadata or raw data aggregator

- If the section adjoins the beginning of the aggregator:
    - If the aggregator is not at EOA:
        - If the aggregator has enough space to fulfill the extra requested:
            - Extend the section into the aggregator and return *extended* to the caller.
        - Otherwise return *not extended* to the caller.
    - If the aggregator is at EOA:
        - If the extra requested is below the extension percentage threshold:
            - Extend the section into the aggregator and return *extended* to the caller.
        - If the extra requested is above the extension percentage threshold:
            - Increase the size of the aggregator, extend the section into the aggregator and return *extended* to the caller.
- Otherwise return *not extended* to the caller.

H5F_FSPACE_STRATEGY_NONE

Extending the section at EOA

- If the section ends at EOA:
    - Extend the file by extra requested and return *extended* to the caller.