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. But these blocks are not page aligned and vary widely in sizes. To provide efficient paged access of these small pieces of metadata and raw data, we propose in this RFC the file space handling mechanism called Paged Aggregation.

1 Introduction

This document addresses the changes to the HDF5 library in order to support the file space handling mechanism called paged aggregation. This mechanism aggregates small metadata and raw data allocations into constant-sized well-aligned pages, which are suitable for page caching. Thus, paged aggregation together with the *Page Buffering* feature will allow 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*.

2 Overview of current file space management

The HDF5 library uses three mechanisms to manage space in an HDF5 file. They are:

- Free-space managers

  They track free-space sections of various sizes in the file that are not currently allocated. 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. 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 file driver associated with the file. For example, if the *sec2* file driver is being used, its allocation routine will increase the size of the file to service the requests.

For file 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 file with non-contiguous address space, it is possible to have one free-space manager for each of the six file space types: raw data and five types of metadata.

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

1. H5F_FILE_SPACE_ALL
    - Mechanisms used: free-space managers, aggregators, and virtual file drivers
    - Does not persist free-space across file opens
    - This strategy is the library default
2. H5F_FILE_SPACE_ALL_PERSIST
    - Mechanisms used: free-space managers, aggregators, and virtual file drivers
    - Persist free-space across file opens
3. H5F_FILE_SPACE_AGGR_VFD
    - Mechanisms used: aggregators and virtual file drivers
    - Does not persist free-space across file opens
4. H5F_FILE_SPACE_VFD
    - Mechanisms used: virtual file drivers
    - Does not persist free-space across file opens

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

2.1 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

2.2 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.

3 Overview of paged aggregation

The design rationale behind 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 section whose size is < file space page size
    - Satisfy request 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:
        - When the section ends at EOA and the section is equal to page size.
    - 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 section whose size is >= file space page size
    - Satisfy request 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 page in the manager
    - Merge two free-space sections when they adjoin

For a file 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 a file with non-contiguous address space, it is possible to have 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
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 the above strategies, the default is not persisting free-space across file opens. User can use the public routine *H5Pset_file_space_strategy()* to request persisting free-space. As the last two strategies do not use free-space managers, the request to persisting free-space is not applicable.

4 Public routines

4.1 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.
- *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*, persisting free-space *persist* 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* is persisting free-space or not.
    - *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, *persist* and *threshold* setting will be ignored.
- *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, persisting free-space condition 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

4.2 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.

5 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.

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***.

6 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.

6.1 H5F_FSPACE_STRATEGY_FSM_AGGR

- The library will request space from the free-space manager depending on the file space type.
- If the request is satisfied, return the address to the caller.
- If the request is not satisfied, the library will request space from either the metadata or raw data aggregator depending on the file space type.
- If the request is satisfied, return the address to the caller.
- If the request is not satisfied, the library will request space from the virtual file driver and return the address to the caller.

6.2 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 file space type.
    - For request size >= page size, request is sent to the large-sized free-space manager for the file space type.
- If the request is satisfied, return the address to the caller.

- If the request is not satisfied:
    - For the small-sized manager, it will request file space page size from the large-sized manager for the file space type. It will then fulfill the space request from the page and put the remaining space in the page into the small-sized free-space manager. Then it will return the address to the caller.
    - For the large-sized free-space manager, it will request the needed space plus possibly misaligned space from the virtual file driver. The misaligned space is the extra space at file end to ensure the file EOA ends on page boundary. It will put the misaligned space into the large-sized free-space manager and then return the page-aligned address to the caller.

6.3 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 is satisfied, return the address to the caller.
- If the request is not satisfied, the library will request space from the virtual file driver and return the address to the caller.

6.4 H5F_FSPACE_STRATEGY_NONE

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

7 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.

7.1 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***.
- 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***.
    - If the section is not merged away or shrunk, add the section to the manager.
    - Return to the caller.

7.2 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***.
- 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:
    - 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***
- If the section is not merged away or shrunk, add the section to the corresponding manager.
- Return to the caller.

7.3 H5F_FSPACE_STRATEGY_AGGR

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

7.4 H5F_FSPACE_STRATEGY_NONE

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

8 Cycle of operation when shrinking file space

The library performs shrinking action 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:
  - Used by the H5F_FSPACE_STRATEGY_PAGE strategy for the small-sized manager
  - Attempt 1 kind of shrinking action:
    - Shrink via EOA: shrink the file at EOA when section size is equal to file space page size
- *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 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

8.1 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.

8.2 H5F_FSPACE_STRATEGY_PAGE

- For a small section at EOA, shrink the file if the section size is equal to file space page size.
- 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.

8.3 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.

8.4 H5F_FSPACE_STRATEGY_NONE

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

9 Cycle of operation when extending file space

After allocating a section of file space and more space is needed, the library might try to extend the existing section for more space if possible via *H5MF_try_extend()* with the cycle of operation described below.

9.1 H5F_FSPACE_STRATEGY_FSM_AGGR

9.1.1 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.

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

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.

9.2 H5F_FSPACE_STRATEGY_PAGE

9.2.1 Extending the section at EOA

- If the section ends at EOA:
    - For a small-sized section:
        - Extend the file by extra requested only if the resulted section does not cross page boundary; otherwise continue with 9.2.2.
    - 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.

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.

9.2.3 Extending the section into the page-end threshold

- 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.

9.3 H5F_FSPACE_STRATEGY_AGGR

9.3.1 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.

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.

9.4 H5F_FSPACE_STRATEGY_NONE

9.4.1 Extending the section at EOA

- If the section ends at EOA:
    - Extend the file by extra requested and return *extended* to the caller.
- Otherwise return *not extended* to the caller.

10 Shutting down free-space managers on file close

When the file is closing down and persisting free-space, the library will first call *H5MF_settle_raw_data_fsm()* to settle file space for the free-space managers that are not self-referential, then it will call *H5MF_settle_meta_data_fsm()* to settle file space for the self-referential free-space managers. These two settle routines are called before the final close of file space via *H5MF_close().*

Self-referential free-space managers are managers that involve file space allocation for the managers' free-space header and section info. The reasons to differentiate these managers from others are:

- The allocation of file space for header and section info may result in the managers being empty, thus forcing us to free the space.
- The allocation of file space for section info may change the size of the section info itself, thus forcing us to free the space.

To avoid the above potential problems that may result in infinite loop, the routine *H5MF_settle_meta_data_fsm()* will first settle file space for managers that are not self-referential by:

- Reduce the file's EOA to the extent possible
- Allocate file space for the *file space info* message
- Allocate file space for the header and section info for managers that are not self-referential

After calling this routine, all raw data allocations are finalized as well as metadata allocations not involving self-referential managers.

Then the routine *H5MF_settle_meta_data_fsm()* will settle file space by:

- Reduce the file's EOA to the extent possible
- Allocate file space for the header and section info for managers that are self-referential

Note that file space for the header and section info of self-referential managers are allocated directly from the virtual file driver. As this may increase the file size on each subsequent file close/open, the problem is resolved by floating the free-space managers on each file open as described in ***Floating free-space managers when reopen file***.

Note: *H5FD_free()* and *H5FD_alloc()* mentioned below are the virtual file driver's *free* and *alloc* callbacks to free and allocate file space.

10.1 H5F_FSPACE_STRATEGY_FSM_AGGR

The *H5MF_settle_raw_data_fsm()* will perform the following:

- Free file space for the metadata and raw data aggregators via *H5MF_xfree()*.
- Free file space allocated for each free-space manager's header and section info via *H5MF_xfree()*.
- Delete the *file space info* message from the superblock extension if allocated.
- Shrink the file's EOA to the extent possible via *H5FD_free().*
- Re-allocate file space for the *file space info* message.
- Re-allocate file space for each free-space manger's header and section info via *H5MF_alloc())*; this is done only for managers that are not self-referential.

The *H5MF_settle_meta_data_fsm()* will perform the following:

- Free file space for the metadata and raw data aggregators via *H5MF_xfree()*.
- Shrink the file's EOA to the extent possible via *H5FD_free()*.
- Save the EOA in *f->shared->eoa_pre_fsm_fsalloc* which is the EOA before the allocation of header and section info for self-referential managers.
- Allocate file space from EOA via *H5FD_alloc()* for each free-space manager's header and section info; this is done only for self-referential free-space managers.

10.2 H5F_FSPACE_STRATEG_PAGE

The *H5MF_settle_raw_data_fsm()* will perform the following:

- Free file space allocated for each free-space manager's header and section info via *H5MF_xfree()*; this is done for the small-sized and large-sized free-space managers.
- Delete the *file space info* message from the superblock extension if allocated.

- Shrink the file's EOA to the extent possible via *H5FD_free()*.
- Re-allocate file space for the *file space info* message.
- Re-allocate file space for each free-space manger's header and section info via *H5MF_alloc())*; this is done only for the small-sized and large-sized managers that are not self-referential.

The *H5MF_settle_meta_data_fsm()* will perform the following:

- Shrink the file's EOA to the extent possible via *H5FD_free()*.
- Save the EOA in *f->shared->eoa_pre_fsm_fsalloc* which is the EOA before the allocation of header and section info for self-referential managers.
- Allocate file space from EOA via *H5FD_alloc()* for each free-space manager's header and section info; this is done only for self-referential free-space managers. The amount of file space allocated is extended to the next file space page size.

10.3 H5F_FSPACE_STRATEGY_AGGR

Since no free-space managers are involved with this strategy, the library will just perform final close of file space via *H5MF_close()*.

10.4 H5F_FSPACE_STRATEGY_NONE

Since no free-space managers are involved with this strategy, the library will just perform final close of file space via *H5MF_close()*.

11 Floating free-space managers when reopen file

When the file is re-opened with read/write access and persisting free-space, the library will float the self-referential free-space managers so as to avoid ever increasing file space for each subsequent file close/open. To do so, it calls *H5MF_tidy_self_referential_fsm_hack()* on the first file space allocation or de-allocation as described below.

11.1 H5F_FSPACE_STRATEGY_FSM_AGGR

- Get the current EOA. If the EOA is the same as *f->shared->eoa_pre_fsm_fsalloc*, we are done and return to the caller.
- Load and open the managers for the free-space header and section info.
- Float the managers for the free-space header and section info via *H5FS_free()* which will unload and release the cache ownership.
- Free the file space for the free-space header and section info via *H5FD_free()*. The EOA after freeing should be *f->shared->eoa_pre_fsm_fsalloc.*

11.2 H5F_FSPACE_STRATEGY_PAGE

- Get the current EOA. If the EOA is the same as *f->shared->eoa_pre_fsm_fsalloc*, we are done and return to the caller.

- Load and open the small-sized managers for the free-space header and section info.
- Load and open the large-sized managers for the free-space header and section info.
- Float the managers for the free-space header and section info via *H5FS_free()* which will unload and release the cache ownership.
- Free the file space for the free-space header and section info via *H5DF_free()*. The EOA after freeing should be *f->shared->eoa_pre_fsm_fsalloc.*

11.3 H5F_FSPACE_STRATEGY_AGGR

Since no free-space managers are involved for this strategy, nothing needs to be done to float the managers on the first file space allocation or de-allocation when reopen file.

11.4 H5F_FSPACE_STRATEGY_NONE

Since no free-space managers are involved for this strategy, nothing needs to be done to float the managers on the first file space allocation or de-allocation when reopen file.

12 Tools

12.1 h5dump

When printing the file creation property information for superblock via the -B option, include the page size obtained via *H5Pget_file_space_page_size*.

12.2 h5stat

When printing file space information via the -S option, include the page size obtained via *H5Pget_file_space_page_size*.

12.3 h5repack

- Add a new option -*G* FS_PAGEISZE
    - Set the file space page size to FS_PAGESIZE via *H5Pset_file_space_page_size*
- Add a new option -P FS_PERSIST
    - Set persisting free-space to FS_PERSIST via *H5Pset_file_space_strategy*
    - FS_PERSIST can be 1 (persist) or 0 (not persist)
    - Options already implemented to set file space info via *H5Pset_file_space_strategy*:
        - -S FS_STRATEGY to set the file space management strategy
        - -T FS_THRESHOLD to set free-space section threshold

13 Testing

Two files, *mf.c* and *tfile.c*, in the *test* directory contain tests to verify the correctness of paged aggregation:

Tests in *mf.c*:

- *test_page_small()*
  - verify that small-sized allocations and de-allocations of file space function as described for paged aggregation.
- *test_page_large():*
  - verify that large-sized allocations and de-allocations of file space function as described for paged aggregation.
- *test_page_small_try_extend()*
  - verify that the extension of an allocated block that is < file space page size function as described for paged aggregation.
- *test_page_large_try_extend()*
  - verify that the extension of an allocated block that is >= file space page size function is as described for paged aggregation.
- *test_page_try_shrink()*
  - verify that shrinking an allocated block that is large or small sized will function as described for paged aggregation.
- *test_page_alloc_xfree()*
  - verify for both persisting or non-persisting free-space, the allocations and de-allocations of file space function as described for paged aggregation.
- *test_page_alignment()*
  - verify that alignment set via H5Pset_alignment() or paged aggregation strategy work as described for page aggregation.

Tests in *tfile.c*:

- *test_userblock_alignment_paged()*
  - verify that the public routines *H5Pset_userblock*, *H5Pset_alignment* and paged aggregation work as expected.
- *test_filespace_info()*
  - verify that the public routines *H5Pget/set_file_space strategy* and *H5Pget/set_file_space_page_size* work as expected.
- *test_file_freespace()*

    - verify that the public routine *H5Fget_freespace()* returned the expected amount of free-space.
- *test_sects_freespace()*
    - verify that the public routine *H5Fget_free_sections()* returned the expected information for free-space sections.

The tests in *test/dsets.c, test/stab.c,* and *test/fheap.c* are modified to run with and without paged aggregation enabled.

14 Documentation

Add reference manual entries for the following two new public routines:

- *H5Pset/get_file_space_page_size*
- *H5Pset/get_file_space_strategy*

Update the following reference manual entries about the changes described previously due to paged aggregation:

- *H5Pset_alignment*
- Tools: *h5repack*, *h5dump*, *h5stat*
- *Version 1 File Space Info* message in *HDF5 File Format Specification*
- Modify description in *HDF5 File Space Management* to reflect file space handling strategies due to paged aggregation

15 Backward/Forward Compatibility

The first compatibility issue is about library release 1.10.0, which has been delivered with the file space handling as described in section 2. For release 1.10.1, paged aggregation as described in this RFC will be used instead. For the following description, we will refer the file space handling in release 1.10.0 as the old method and the one in release 1.10.1 as the new method. The compatibility issue will be addressed as below:

- The public routines *H5Pset/get_file_space()* will be deprecated in release 1.10.1.
- When opening a file that is created under release 1.10.0 with the old method, the library will read the old *File Space Info* message with version 0 and map the information to the new *File Space Info* message with version 1. Then the library will delete the old message and create the new message with the mapped information in the superblock extension. This means the file will have the version 1 *File Space Info* message from this point on.
- Generate HDF5 files with the old method under library release 1.10.0. File a bug report about the format change and deposit the files there for future reference.

Another compatibility issue is the possibility of a round trip scenario between library release 1.10.1 that supports the new method and library releases like 1.8 that do not understand the new method:

- Create an HDF5 file and dataset under library release 1.10.1
- Open the HDF5 file and modify the dataset under library release 1.8
- Open the HDF5 file and modify the dataset again under library release 1.10.1

This issue is addressed by enabling the H5O_MSG_FLAG_MARK_IF_UNKNOWN flag in the *File Space Info* message when a file is created under library release 1.10.1. When 1.8 library opens the file, it will mark the *File Space Info* message as unknown by setting the H5O_MSG_FLAG_WAS_UNKNOWN flag. Later on, when the 1.10.1 library opens the file again, it will detect the flag being set and will operate with the library default file space handling.

16 Limitations

Currently, when multi/split driver is used in combination with paged aggregation strategy and/or persisting free-space, file creation/open will fail.

17 Future Issues

Acknowledgements

This work was supported by Lawrence Livermore National Laboratory (LLNL). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author[s] and do not necessarily reflect the views of LLNL.

Revision History

August 22, 2012	Version 0 – Modifications based on *RFC: HDF5 File Space Allocation and Aggregation* by John Mainzer
January 10, 2016	Version 1 –Updated to reflect current implementation.
January 13, 2016	Version 2—Updated after review.