# RFC: Read Attempts for Metadata with Checksum

**Vailin Choi**

The HDF5 library encounters intermittent checksum failure from reading metadata of an HDF5 file opened with single-writer/multiple-reader (SWMR) access. The failure is from checksum check that does not reflect the data read. This RFC describes the modifications to the library that address this problem.

## 1 Introduction

There are intermittent test failures from running the tests in test/testswmr.sh. The test script covers tests to expand datasets and write to random locations in datasets for an HDF5 file opened with SWMR access. The failures encountered are from comparing the checksum computed for the metadata with the checksum stored within the piece of metadata. The debugging efforts for these test failures seem to indicate that old and new data are read from disk on a system that is not atomic (see page 3 footnote of [1]). To mitigate this problem, the library will attempt reading the piece of metadata again for a set number of times so as to obtain data that will pass the checksum check.

## 2 Approach

To provide consistent metadata for SWMR operations on systems that are not atomic, the solution is as follows:

- Use the library's data structures that have check-summed metadata. This condition will be ensured when a user creates and opens a file with the latest library format. For data structures that do not have checksum, the library will refuse to create such data structures under SWMR write conditions. Such change is currently in place only for the version 1 B-tree data structure that does not have checksum, and will be extended to other non-check-summed data structures in the future.
- Turn off the usage of the library's accumulator for a file opened with SWMR access. This will avoid possible ordering problem on a system that is not atomic.
- Modify cache clients in the library to re-read check-summed metadata so as to obtain consistent metadata that will pass the checksum check.
- Add two new public routines *H5Pget/set_metadata_read_attempts* to assist a user in tuning the number of read attempts to other than the library default value.
- Collect and provide statistics regarding the number of read retries that occur for metadata entries that have a checksum. Provide the information to the user via a new public routine *H5Fget_metadata_read_retries_info*.

# 3 Modifications in the Cache Clients

Changes will be done to the coding of the cache clients where reading of the metadata is performed. When a file is opened with SWMR access, the library will repeatedly read the piece of check-summed metadata until the checksum check passes or the allowed number of read attempts is reached. If the maximum number of read attempts is reached, the return code for the read call will be failure.

The library sets a default value for the maximum number of read attempts to try. For a file opened with SWMR access, the library's default maximum number of read attempts is 100. This value is derived based on the results of the atomicity tests (see explanation in Appendix A). For a file opened without SWMR access, this number is 1.

A user can set a different maximum number of read attempts via the public routine *H5Pset_metadata_read_attempts* as described in the next section. The following table lists the 8 cache clients with check-summed metadata entries where attempts to do re-reads will be implemented:

Table 1. Cache clients with metadata entries that have checksum

| **Cache clients** | **Metadata entries**[a] | **H5AC_type_t**[b] |
|---|---|---|
| Object header[c] | Object header (version 2) | H5AC_OHDR_ID |
| | Object header chunk (version 2) | H5AC_OHDR_CHK_ID |
| B-tree[d] | B-tree header (version 2) | H5AC_BT2_HDR_ID |
| | B-tree internal node (version 2) | H5AC_BT2_INT_ID |
| | B-tree leaf node (version 2) | H5AC_BT2_LEAF_ID |
| Fractal heap | Fractal heap header | H5AC_FHEAP_HDR_ID |
| | Fractal heap direct block (optional checksum) | H5AC_FHEAP_DBLOCK_ID |
| | Fractal heap indirect block | H5AC_FHEAP_IBLOCK_ID |
| Free-space manager | Free-space header | H5AC_FSPACE_HDR_ID |
| | Free-space sections | H5AC_FSPACE_SINFO_ID |
| Shared object header message | Shared object header message table | H5AC_SOHM_TABLE_ID |
| | Shared message record list | H5AC_SOHM_LIST_ID |
| Extensive array | Extensive array header | H5AC_EARRAY_HDR_ID |
| | Extensive array index block | H5AC_EARRAY_IBLOCK_ID |
| | Extensive array super block | H5AC_EARRAY_SBLOCK_ID |
| | Extensive array data block | H5AC_EARRAY_DBLOCK_ID |
| | Extensive array data block page | H5AC_EARRAY_DBLK_PAGE_ID |
| Fixed array | Fixed array header | H5AC_FARRAY_HDR_ID |
| | Fixed array data block | H5AC_FARRAY_DBLOCK_ID |
| | Fixed array data block page | H5AC_FARRAY_DBLK_PAGE_ID |

| | | |
|---|---|---|
| File's superblock[e] | Superblock (version 2) | H5AC_SUPERBLOCK_ID |

[a]All metadata entries are of version 0 except indicated otherwise.

[b]The corresponding defines for the cached metadata entries in H5ACprivate.h.

[c]Version 1 of object header does not have checksum.

[d]Version 1 of B-tree does not have checksum.

[e]Version 0 and version 1 of a file's superblock do not have checksum.

# 4 Tuning the Number of Read Attempts

The library will provide two new public routines for users to set and retrieve the number of read attempts as described below.

## 4.1 H5Pset_metadata_read_attempts

**Name:**

`H5Pset_metadata_read_attempts`

**Signature:**

*herr_t* `H5Pset_metadata_read_attempts`*(hid_t* `plist_id`*, unsigned* `attempts`*)*

**Purpose:**

Sets the number of metadata read attempts in a file access property list.

**Description:**

*`H5Pset_metadata_read_attempts`* sets the number of metadata reads that the library will try when reading check-summed metadata in an HDF5 file opened with SWMR access. When reading such metadata, the library will compare the checksum computed for the metadata with the checksum stored within the piece of metadata. When performing SWMR operations on a file, the checksum check might fail when the library reads data on a system that is not atomic. To remedy such situation, the library will repeatedly read the piece of metadata until the check passes or finally fails the read when the allowed number of attempts is reached.

The number of read attempts used by the library will depend on how the file is opened and whether the user sets the number of read attempts via this routine:

- For a file opened with SWMR access:
  - If the user sets the number of attempts to N, the library will use N.
  - If the user does not set the number of attempts, the library will use the default for SWMR access (100).
- For a file opened without SWMR access, the library will always use the default for non-SWMR access (1). The value set via this routine does not have any effect.

**Parameters:**

| | |
|---|---|
| *hid_t* `plist_id` | IN: Identifier for a file access property list. |
| *unsigned* `attempts` | IN: The number of read attempts which is a value greater than 0. |

**Returns:**

Returns a non-negative value if successful; otherwise returns a negative value.

**Example Usage:**

The first example illustrates the case in setting the number of metadata read attempts for a file opened with SWMR access.

```
/* Create a copy of file access property list */
fapl = H5Pcreate(H5P_FILE_ACCESS);

/* Set the # of metadata read attempts */
H5Pset_metadata_read_attempts(fapl, 20);

/* Open the file with SWMR access and the non-default file access
   property list */
fid = H5Fopen(filename, (H5F_ACC_RDONLY | H5F_ACC_SWMR_READ), fapl);

/* Get the file's file access property list */
file_fapl = H5Fget_access_plist(fid);

/* Retrieve the # of read attempts from the file's file access
   property list */
H5Pget_metadata_read_attempts(file_fapl, &attempts);

/*
 * The value returned in "attempts" will be 20.
 * The library will use 20 as the number of read attempts
 * when reading check-summed metadata in the file
 */

/* Close the property lists */
H5Pclose(fapl);
H5Pclose(file_fapl);

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

The second example illustrates the case in setting the number of metadata read attempts for a file opened without SWMR access.

```
/* Create a copy of file access property list */
fapl = H5Pcreate(H5P_FILE_ACCESS);

/* Set the # of metadata read attempts */
H5Pset_metadata_read_attempts(fapl, 20);

/* Open the file without SWMR access and the non-default file access
   property list */
fid = H5Fopen(filename, H5F_ACC_RDONLY, fapl);

/* Get the file's file access roperty list */
file_fapl = H5Fget_access_plist(fid);

/* Retrieve the # of read attempts from the file's file access
```

```
   property list */
H5Pget_metadata_read_attempts(file_fapl, &attempts);

/*
 * The value returned in "attempts" will be 1
 * (default for non-SWMR access).
 * The library will use 1 as the number of read attempts
 * when reading check-summed metadata in the file
 */

/* Close the property lists */
H5Pclose(fapl);
H5Pclose(file_fapl);

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

## 4.2 H5Pget_metadata_read_attempts

**Name:**

H5Pget_metadata_read_attempts

**Signature:**

*herr_t* H5Pget_metadata_read_attempts *(hid_t* plist_id*, unsigned **attempts*)*

**Purpose:**

Retrieves the number of metadata read attempts from a file access property list.

**Description:**

*H5Pset_read_attempts* retrieves the number of metadata read attempts that is set in the file access property list *plist_id*.

For a default file access property list, the value retrieved will depend on whether the user sets the number of read attempts via H5Pset_metadata_read_attempts:

- If the number of attempts is set to N, the value returned will be N.
- If the number of attempts is not set, the value returned will be the default for non-SWMR access (1).

For the file access property list of a specified HDF5 file, the value retrieved will depend on how the file is opened and whether the user sets the number of read attempts via H5Pset_metadata_read_attempts:

- For a file opened with SWMR access:
  - If the number of attempts is set to N, the value returned will be N.
  - If the number of attempts is not set, the value returned will be the default for SWMR access (100).
- For a file opened without SWMR access, the value retrieved will always be the default for non-SWMR access (1). The value set via H5Pset_metadata_read_attempts does not have any effect.

**Parameters:**

| | |
|---|---|
| *hid_t* plist_id | IN: Identifier for a file access property list. |

*unsigned* *`attempts` OUT: The number of read attempts.

**Returns:**

Returns a non-negative value if successful; otherwise returns a negative value.

**Example Usage:**

The first example illustrates the two cases for retrieving the number of metadata read attempts from a default file access property list.

```
/* Get a copy of file access property list */
fapl = H5Pcreate(H5P_FILE_ACCESS);

/* Retrieve the # of read attempts from the file access property list */
H5Pget_metadata_read_attempts(fapl, &attempts);

/*
 * The value returned in "attempts" will be 1
 * (default for non-SWMR access).
 */

/* Set the # of read attempts to 20 */
H5Pset_metadata_read_attempts(fapl, 20);

/* Retrieve the # of read attempts from the file access property list */
H5Pget_metadata_read_attempts(fapl, &attempts);

/*
 * The value returned in "attempts" will be 20 as set.
 */

/* Close the property list */
H5Pclose(fapl);
```

The second example illustrates the two cases for retrieving the number of metadata read attempts from the file access property list of a file opened with SWMR access.

```
/* Open the file with SWMR access and default file access
   property list */
fid = H5Fopen(filename, (H5F_ACC_RDONLY|H5F_ACC_SWMR_READ), H5P_DEFAULT);

/* Get the file's file access roperty list */
file_fapl = H5Fget_access_plist(fid);

/* Retrieve the # of read attempts from the file's file access property
   list */
H5Pget_metadata_read_attempts(file_fapl, &attempts);

/*
 * The value returned in "attempts" will be 100
 * (default for SWMR access).
 */
```

```
/* Close the property list */
H5Pclose(file_fapl);

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

/* Create a copy of file access property list */
fapl = H5Pcreate(H5P_FILE_ACCESS);

/* Set the # of metadata read attempts */
H5Pset_metadata_read_attempts(fapl, 20);

/* Open the file with SWMR access and the non-default file
   access property list */
fid = H5Fopen(filename, (H5F_ACC_RDONLY|H5F_ACC_SWMR_READ), fapl);

/* Get the file's file access roperty list */
file_fapl = H5Fget_access_plist(fid);

/* Retrieve the # of read attempts from the file's file access
   property list */
H5Pget_metadata_read_attempts(file_fapl, &attempts);

/*
 * The value returned in "attempts" will be 20.
 */

/* Close the property lists */
H5Pclose(file_fapl);
H5Pclose(fapl);

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

The third example illustrates the two cases for retrieving the number of metadata read attempts from the file access property list of a file opened without SWMR access.

```
/* Open the file without SWMR access and default file access
   property list */
fid = H5Fopen(filename, H5F_ACC_RDONLY, H5P_DEFAULT);

/* Get the file's file access roperty list */
file_fapl = H5Fget_access_plist(fid);

/* Retrieve the # of read attempts from the file's file access
   property list */
H5Pget_metadata_read_attempts(file_fapl, &attempts);

/*
 * The value returned in "attempts" will be 1
 * (default for non-SWMR access).
 */
```

```
/* Close the property list */
H5Pclose(file_fapl);

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

/* Create a copy of file access property list */
fapl = H5Pcreate(H5P_FILE_ACCESS);

/* Set the # of metadata read attempts */
H5Pset_metadata_read_attempts(fapl, 20);

/* Open the file without SWMR access and the non-default file
   access property list */
fid = H5Fopen(filename, H5F_ACC_RDONLY, fapl);

/* Get the file's file access roperty list */
file_fapl = H5Fget_access_plist(fid);

/* Retrieve the # of read attempts from the file's file access
   property list */
H5Pget_metadata_read_attempts(file_fapl, &attempts);

/*
 * The value returned in "attempts" will be 1
 * (default for non-SWMR access).
 */

/* Close the property lists */
H5Pclose(file_fapl);
H5Pclose(fapl);

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

## 5 Statistics for Read Retries

The library will track the number of read retries for each metadata entry of a cache client that has checksum. Statistics for the 21 metadata entries of the 8 cache clients are collected and provided to the user via a new public routine *H5Fget_metadata_read_retries_info* as described below. When the current number of read attempts is unable to remedy the reading of inconsistent metadata on a system, the user can assess the information obtained via this routine to derive a different maximum value. The information can also be helpful for debugging purposes to identify potential issues with metadata flush dependencies and general SWMR implementation.

**Name:**

H5Fget_metadata_read_retries_info

**Signature:**

*herr_t* H5Fget_metadata_read_retries_info*(hid_t* file_id*, H5F_retries_info_t **info*)*

**Purpose:**

Retrieves the collection of read retries for metadata entries with checksum.

**Description:**

*H5Fget_metadata_read_retries_info* retrieves information regarding the number of read retries for metadata entries with checksum for the file *file_id*. The information is reported with a data structure *H5F_retries_info_t* defined in *H5Fpublic.h*:

```
#define NUM_METADATA_READ_RETRIES     21

typedef struct H5F_retries_info_t {
      unsigned nbins;
      uint32_t *retries[NUM_METADATA_READ_RETRIES];
};
```

*nbins* is the number of bins for each *retries[i]* of metadata entry *i*. It is calculated based on the current number of read attempts used in the library and logarithmic 10.

If there are read retries occurred for metadata entry *i*, the library will allocate memory for *retries[i]* (*nbins * sizeof(uint32_t)*) and store the collection of retries there. If there are no retries for metadata entry *i*, *retries[i]* will be NULL. After each call to *H5Fget_metadatda_read_retries_info*, user should free each *retries[i]* that is non-NULL, otherwise resource leak will occur.

For the library default read attempts of 100 for SWMR access, *nbins* will be 2 as depicted below:

- *retries[i][0]* is the number of 1 to 9 read retries.
- *retries[i][1]* is the number of 10 to 99 read retries.

For the library default read attempts of 1 for non-SWMR access, *nbins* will be 0 and each *retries[i]* will be NULL.

The following table lists the 21 metadata entries of *retries[]*:

**Table 2. Metadata entries with tracking for read retries**

| Index for *retries* | Metadata entries[+] |
|---|---|
| 0 | Object header (version 2) |
| 1 | Object header chunk (version 2) |
| 2 | B-tree header (version 2) |
| 3 | B-tree internal node (version 2) |
| 4 | B-tree leaf node (version 2) |
| 5 | Fractal heap header |
| 6 | Fractal heap direct block (optional checksum) |
| 7 | Fractal heap indirect block |
| 8 | Free-space header |
| 9 | Free-space sections |
| 10 | Shared object header message table |
| 11 | Shared message record list |
| 12 | Extensive array header |
| 13 | Extensive array index block |

| | |
|---|---|
| 14 | Extensive array super block |
| 15 | Extensive array data block |
| 16 | Extensive array data block page |
| 17 | Fixed array header |
| 18 | Fixed array data block |
| 19 | Fixed array data block page |
| 20 | File's superblock (version 2) |

[+]All entries are of version 0 except indicated otherwise.

**Parameters:**

| | |
|---|---|
| *hid_t* `file_id` | IN: Identifier of a currently-open HDF5 file. |
| *H5F_retries_info_t* *`info` | OUT: Struct containing the collection of read retries for metadata entries with checksum. |

**Returns:**

Returns a non-negative value if successful; otherwise returns a negative value.

**Example Usage:**

This example illustrates the case on retrieving the collection of read retries for a file opened with SWMR access.

```
H5F_retries_info_t info;

/* Get a copy of file access property list */
fapl = H5Pcreate(H5P_FILE_ACCESS);

/* Set to use the latest library format */
H5Pset_libver_bounds(fapl, H5F_LIBVER_LATEST, H5F_LIBVER_LATEST);

/* Create a file with the latest format */
fid = H5Fcreate(filename, H5F_ACC_TRUNC, H5P_DEFAULT, fapl);

/* Perform operations on groups/datasets etc. in the file */
:
:
:

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

/* Perform operations via a writer */
fidw = H5Fopen(filename, H5F_ACC_RDWR |H5F_ACC_SWMR_WRITE, fapl);
:
:
:
/* Perform operations via multiple readers */
/* For example: open a file with SWMR access */
fidr = H5Fopen(FILE, H5F_ACC_RDONLY|H5F_ACC_SWMR_READ, fapl);

/* Perform operations on group/datasets etc. in the file */
```

```
:
:
:
/* Retrieve the collection of read retries for the file */
H5Fget_metadata_read_retries_info(fidr, &info);

/* Print the collection of read retries */
for(i = 0; i < NUM_METADATA_READ_RETRIES; i++) {
      if(info. retries[i] != NULL) {
            printf("Read retries for entry %u:\n", i);

            /* info.nbins will be 2 */
            for(j = 0; j < info.nbins; j++)
            /*
             * Print the following if nonzero:
             * info.retries[i][0] for # of 1-9 read retries.
             * info.retries[i][1] for # of 10-99 read retries
             */
      } /* end if */
} /* end for */

/* Free the array of retries */
for(i = 0; i < NUM_ METADATA_READ_RETRIES; i++)
      if(info.retries[i] != NULL)
            free(info.retries[i]);

:
:
:
```

## 6 Future enhancements

Future enhancement can be added to the collection of statistics in *H5Fget_metadata_read_retries_info* as follows:

- Collect the number of read retries according to metadata size.
- Collect the number of read retries according to object type.

## Acknowledgements

This work was supported by a customer of The HDF Group, Diamond Light Source.

## Revision History

| | |
|---|---|
| *October 4, 2013:* | Version 1 circulated for comment within The HDF Group SWMR team. |
| *October 8, 2013* | Version 2 incorporated comments from SWMR team |
| *October 10, 2013* | Version 3 incorporated comments from SWMR team |

## Appendix A:

The atomicity test is developed by the HDF group to check atomic operations on a system. It validates whether a reader will read either all or none of any write by a writer. When inconsistent data is obtained, the reader will perform re-reads up to a defined number of attempts until valid data is obtained or finally fails the read at hand. The test is run with the writing and reading of different sized buffers. The atomicity test is available in the HDF5 library source: *test/atomic_reader.c* and *test/atomic_writer.c*.

Based on the results of the test runs, the atomicity test usually succeeds within 20 attempts in getting consistent data. But there are test runs that fail to obtain valid data beyond 20 attempts. To take a conservative estimate, the value of 100 is chosen for the library's default number of read attempts.

## References

1. The HDF Group. "RFC: SWMR Requirements and Use Cases," http://confluence.hdfgroup.uiuc.edu/pages/viewpage.action?pageId=25100365 (February 19, 2013).