H5edit Atomicity Feature

Proposed by Albert Cheng

1 Purpose

This describes the requirements and design of the Atomicity feature of the H5edit tool. Section 2 shows a user case of the need of the atomicity feature. Section 3 lists the requirements of the feature. Section 4 describes the implementation design. Section 5 shows the validation requirements and implementation. Section 6 lists the documentation needed for each stage of implementation. Section 7 is a summary of time estimate of implementation.

2 A User Story of the Need of Atomicity

When a user uses the H5edit tool to modify an HDF5 data file, the file is opened and modifications are made according to the edit commands specified by the user. There are two kinds of potential failures.

2.1 User command errors

The user specifies some incorrect edit commands. For example, he CREATE a new attribute and DELETE another attribute of a dataset successfully. He then attempts to CREATE a new attribute that exists already. The H5edit tool will issue an error and exit the tool with a failure status code. At that point, the original data file is already partially changed. The user may want to have his edit commands be done all or none, not partially. Unless he has made a backup copy of the data file before starting the H5edit tool, he cannot undo the partial changes. For example, he cannot recover the original attribute that has been deleted.

2.2 System failures

Sometimes computer resources fail. E.g., a disk runs out of space. If this happens while the H5edit tool is running, the tool has to exit and leaves behind an unstable data file that is not accessible by the HDF5 library any more. If the user has not made a backup copy of the original data file, all is lost.

3 Requirement Specifications

In this section, I will describe the functionality of the Atomicity option of the H5edit tool that will provide a means to restore the original data files in the failures described before.

3.1 Function Definition

The H5edit tool should create and maintain a backup copy of the original data file being edited by the tool. The Atomicity option controls the manner the backup copy is managed. In case of user commands errors or system failures, the data file can be recovered from the backup copy by replacing the data file with the backup copy.

3.2 Atomicity Option

Commands are applied to the data file in an atomic manner, that is, the file will not be modified only partially if H5edit encounters error during the execution.
The Atomicity option will first make a backup copy of the data file before applying the input command. If the tool encounters any error, the user may recover the data file from the backup copy.
The Atomicity option, --atomic, supports 3 levels, *yes*, *no*, and *inc*.

- --atomic=yes applies the input commands in an all or none manner. That is, either all input commands are applied to the data file, or none is applied.
- --atomic=no will not make the backup copy and apply the input commands as much as possible. The user may use this if he is sure of the validity of the input commands or he is not concern if the data file is partially modified or corrupted.
- --atomic=inc (default) will apply the changes in an all or none manner but at a command level. If H5edit encounters an error during execution, the data file can be restored back to the last command applied successfully.

4 Implementation Design

4.1 Backup File Name Convention

A backup copy of the data file shall have a file name that is composed of the file name of the data file but proceeded with a period and appended with ".bck". For example, if the data file name is "2010_10_01_data.h5", the backup file name will be ".2010_10_01_data.h5.bck".

4.2 When and How the Backup File is Generated and Removed

The backup file will be generated from the original data file before any change is applied to the data file. Upon the successful execution of the H5edit tool, the backup file will be deleted before the tool exits. If there is any error encounter during the H5edit tool execution, the backup file will not be deleted.

4.3 Implementation stage

The complete implementation of the Atomicity option will be delivered in three stages.

4.3.1 Stage 1: Basic backup support

The --atomic option is implemented with support for the *yes* and *no* levels but not the *inc* level.
For the atomic *yes* level, the backup file is generated right after the data file is opened successfully. No more modification is applied to the backup file during the

execution of the H5Edit tool. Upon the successful execution of the H5edit tool, the backup file is deleted before the tool exits. If there is any error during the H5edit tool execution, the backup file is not deleted.
For the atomic *no* level, no backup file is generated nor deleted.

4.3.2 Stage 2: Support for Incremental Atomicity

In addition to the functionality of Stage 1, --atomic=*inc* will be supported in the following manner:
During the execution of the H5edit tool, whenever an edit command is completed successfully, H5edit issues an H5Fflush if the atomicity level is *inc*. The H5Fflush will trigger the update of the backup file by copying the data file to the backup file before H5edit will continue to execute the next edit command.

4.3.3 Stage 3: Performance Enhancement

In addition to the functionality of Stage 2, the update of the backup file is enhanced in the following manner:
The data file is divided into segments or regions of 1 MB each. Changes made to the data file are recorded according to segments touched. When the backup file should be updated, only segments that have been touched are copied from the data file to the backup file. After the update, the record of segments touched is reset for the next round of changes.

4.4 Implementation time estimate

The time estimate for design, implementation and testing of each stage is listed below. Documentation time estimate is listed in a later section.

4.4.1 Stage 1

1. Design and implement a simple mechanism to generate backup copy at file open and to remove backup file at file close: 12 hours.
2. Set up testing for backup feature: 20 hours.
3. (If time permits) Perform comparison of execution speeds between *yes* and *no* levels of atomicity: 15 hours.
4. Contingency: in case there are requirements changes or extra tasks come up, reserve 25% more time for their implementation. If no contingency arises, the reserved time can be applied to do the option work. The reserved estimate is 8 hours.

Total for stage 1 implementation is: 32--40 hours + optional 15 hours.

4.4.2 Stage 2

1. Design and implement the *inc* level of atomicity: 16 hours
2. Add tests to validate --atomic=inc is working correctly: 12 hours
3. (If time permits) Perform comparison of execution speeds between *inc* and *yes* levels of atomicity: 15 hours.
4. Contingency: in case there are requirements changes or extra tasks come up, reserve 25% more time for their implementation. If no contingency arises, the reserved time can be applied to do the option work. The reserved estimate is 7 hours.

Total for stage 2 implementation is: 28--35 hours + optional 20 hours.

4.4.3 Stage 3

The segment management is needed to enhance the performance of the *inc* level of atomicity. This requires access at the level of an HDF5 Virtual File Driver (VFD). The Backup VFD proposal will meet the need of the optimization via segments management.

1. Design and implement the Backup VFD: 30 hours.
2. Validate the Backup VFD can generate, update and remove the backup copy correctly: 20 hours.
3. Design and implement performance enhancement by deploying the Backup VFD in H5edit: 20 hours.
4. Validate the enhancement supports the backup correctly: 20 hours.
5. Perform comparison of execution speeds between previous and the new versions of h5edit tools to show speed up. 15 hours.
6. Contingency: in case there are requirements changes or extra tasks come up, reserve 25% more time for their implementation. The reserved estimate is 26 hours.

Total for stage 3 implementation is: 105--131 hours.

5 Validation Requirements and Implementation

Setup tests to validate the H5edit tool is meeting all the requirements specified in the Requirement Specification section. In this section, *data file* means the original version or an incrementally modified version of the data file argument.

5.1 Recover Data File

Requirement: show the data file can be recovered after user command input errors or system errors are encountered. This validates requirement 3.1.

5.1.1 Recover after user command input error

Requirement: show the data file can be recovered after user command input errors.
Implementation: a test with the following steps:

1. Setup a command input file containing bad commands;
2. Run h5edit tool with the bad command input file;
3. H5edit tool should terminate with error;
4. Verify data file is unstable (e.g., h5dump cannot display it);
5. Recover data file from the backup file;
6. Verify data file is valid now. (e.g., h5dump can display it.)

5.1.2 Recover after System Errors

Requirement: show the data file can be recovered after system errors.
Implementation: There are many types of system errors such as disk quotas, disk filled up, execution limit, system crashes, … Most are hard to generate at will.
The plan is to implement one of the following tests.

5.1.2.1 test it against an execution time limit

The test is setup by the following steps:

1. Prepare a big HDF5 file that has many datasets and attributes such that it takes a long time to find an attribute of a specific dataset.
2. Setup a good command input file containing the several DELETE attributes and several more ADD attributes of different data types. (This ensure changes of metadata and rawdata.)
3. Run the h5edit command using the good command input file against the big HDF5 file, expecting it will take at least over 10 seconds to run.
4. Meanwhile, launch a separate command to send a kill signal to the running h5edit process, terminating it.
5. Verify that the original data file is unstable (e.g., h5dump cannot display it).
6. Recover the data file from the backup file.
7. Verify that the data file is valid now. (e.g., h5dump can display its contents.)

5.1.2.2 test it against a disk full error

The test is set up by the following steps:

1. Setup a good command input file containing the CREATE a large attribute;
2. Copy the original data file to a file system that has free space exactly for another copy of the data file. This would allow the backup copy file to be generated but nothing else.
3. Run the h5edit command in this directory using the good command input file from step 1.
4. H5edit is expected to fail because of the creation of the large attribute.
5. Verify that the original data file is unstable (e.g., h5dump cannot display it).
6. Recover the data file from the backup file.
7. Verify that the data file is valid now. (e.g., h5dump can display its contents.)

5.2 Validate Atomicity Option Levels

Requirements: show the h5edit tool functions according to the atomicity level as specified in the --atomic option. This validates requirement 3.2.

5.2.1 Validate Atomicity level of yes

Requirement:

When --atomic=yes is used, all or none of input commands are applied.

Implementation:

Setup a test in the following steps:

1. Setup a bad command input file containing a correct command of DELETE an existing attribute and an incorrect command of DELETE a non-existing attribute.
2. Make an extra copy of the data file.
3. Run the h5edit --atomic=yes command with the bad command input file above. H5edit will fail with messages complaining trying to delete a non-existing attribute.
4. Compare the data file with the extra copy created in step 2. E.g., use h5diff to compare them. The comparison should fail.

5. Recover the data file from the backup file.
6. Compare the data file with the extra copy created in step 2. This time, the comparison should pass.

5.2.2 Validate Atomicity level of no

Requirement:
When --atomic=no is used, the backup copy is not created.
Implementation:
The requirement has two implications. First of all, the backup copy file is not created nor deleted when the tool ends. This also means it will not check if the backup file already exists when it starts. The second implication is that if the H5edit tool fails, the original data file becomes unstable and there is no backup copy file for recovery.

5.2.2.1 Verify backup copy is not created

Setup a test in the following steps:
1. Create an empty backup copy of the data file.
2. Run the h5edit --atomic=no command with a good command input file (from previously generated legal command input file). H5edit is expected to complete without any message complaining "not able to generate backup file because it already exists". This is because --atomic=no does not generate the backup file, nor verify if one already exists.
3. Verify the empty backup file is still present.
4. Verify that the data file is stable. (E.g., use h5dump to display it.)

5.2.2.2 Verify backup copy is not created and there is no means to recover

1. Setup a bad command input file containing a correct command of DELETE an existing attribute and an incorrect command of DELETE a non-existing attribute.
2. Create an empty backup copy of the data file.
3. Run the h5edit --atomic=no command with the command input file from step 1. H5edit is expected to fail but without any message complaining "not able to generate backup file because it already exists".
4. Verify that the data file is unstable (e.g., h5dump cannot display it.)
5. Verify that the empty backup copy file is still around but it cannot be used to recover the data file.

5.2.3 Validate Atomicity Level of inc

Requirement:
When --atomic=inc is used, the data file can be recovered back to the last successful command in case of user or system errors.
Implementation:
There are two cases to test. First case is that h5edit --atomic=inc runs without error. Verify that there is no backup file left behind, that is, the backup file is deleted before the h5edit exits without error. Second case is that h5edit --atomic=inc encounters some errors and exits. Verify there is a backup file left behind and it can be used to recover the data file.

5.2.3.1 Verify successful run
Setup a test in the following steps:

1. Setup a good command input file containing a correct command of DELETE an existing attribute.
2. Make an extra copy of the data file.
3. Verify that the backup file does not exist.
4. Run the h5edit --atomic=inc command with the good command input file from step 1. H5edit is expected to complete without any message.
5. Verify that the backup file does not exist.
6. Verify that the data file is stable and has one fewer attribute than the extra copy of data file (e.g., use h5diff to compare the two files.)

5.2.3.2 Verify fail run
Setup a test in the following steps:

1. Setup a bad command input file containing a correct command of DELETE an existing attribute and an incorrect command of DELETE a non-existing attribute.
2. Make an extra copy of the data file.
3. Verify the backup file does not exist.
4. Run the h5edit --atomic=inc command with the bad command input file from step 1. H5edit is expected to fail.
5. Verify that the data file is unstable (e.g., h5dump cannot display it.)
6. Verify that the backup file is present and can be used to restore the data file.
7. Verify the data file is stable and has one fewer attribute than the extra copy of data file (e.g., use h5diff to compare the two files.)

6 Documentation Requirements
There are two documents, H5edit User Reference Page and H5edit User Guide that need update when the Atomicity feature is implemented. The required updates are listed according to the 3 stages of implementation.

6.1 Stage 1

6.1.1 Initial updates
Update H5edit User Reference Page with the new Atomicity option, with notes marking --atomic=inc is accepted but not supported yet.
Update H5edit User Guide with the explanation and examples of what the Atomicity option support and the current limit (*inc* level is not implemented yet). Time estimate is 12 hours.

6.1.2 Performance comparison (if time permits)
May present performance difference between the two *yes* and *no* levels of atomicity when handling big data files. Time estimate is 8 hours.

Total document time is 12 hours + optional 8 hours.

6.2 Stage 2

6.2.1 Update with inc level support

Update H5edit User Reference Page that --atomic=inc is supported.
Update H5edit User Guide with the explanation and examples of what the *inc* level of atomicity provides. Time estimate is 8 hours. .

6.2.2 Performance comparison (if time permits)

May present performance difference between the two *yes* and *inc* levels of atomicity when handling big data files. The *inc* level is expected to take more time with the added benefit. Time estimate is 8 hours.

Total document time is 8 + optional 8 hours.

6.3 Stage 3

6.3.1 Update with performance enhancement

No updates needed for the User Reference Page.
Update the H5edit User Guide with performance enhancement description and some performance measurements showing the speedup in execution speed of the *inc* level of atomicity. Time estimate is 16 hours.

Total document time is 16 hours.

7 Summary

The atomicity feature will be implemented and delivered in 3 stages (releases) together with updated documents. The table below shows a summary of time estimate of each stage for DIV (Design, Implementation, Validation) and Doc (Documentation). The optional work of stage 1 and stage 2 can be postponed to stage 3 if releases 1 and 2 are needed sooner. By stage 3, the optional work must be done in order to demonstrate the overall performance in execution time. The total time estimate to implement the feature is 288 hours.

	DIV	Doc	Optional DIV	Optional Doc	Total (DIV, Doc, optional work)	Total minus optional work
Stage 1	40	12	15	8	75	52
Stage 2	35	8	15	8	66	43
Stage 3	131	16	0	0	147	147
Total	206	36	30	16	288	242

Table 1: Summary of Time Estimate (hours)

Revision History

Date	Revisions
2012-10-30	Rev 1: Function requirements drafted.
2012-11-09	Rev 2: Requirements for validation and document added.