RFC: h5diff - Ignoring object from comparing

Jonathan Kim

This RFC proposes to add a feature to exclude one or more objects from comparing one file to the other with h5diff tool. This feature will allow user to ignore object(s) selectively and verify the equality for the rest of object(s).

# Introduction and background

The h5diff tool has ability to compare either by files or by objects.

When comparing by objects, h5diff compare the two given objects.

When comparing by files, h5diff compares objects with matching path and name from a source file to a target file. If there is an object and its path and name doesn’t match from other file, h5diff will not compare the object, thus h5diff doesn’t display the difference related to the object. However the exit code will indicate that the two files are different.

However there is a situation that a user wants to ignore some objects from comparing intentionally and verify the equality for the rest in the files.

Mark Linda, a NASA customer, has two HDF files and knows the two files contain same content of datasets except a few datasets with just different time stamp. He wants to verify if the two files are same. However without the ignoring object feature, he has to work around to verify the equality.

If a file contains small number of objects, he may be able to just compare by each object manually to figure out. However if the file contains many objects, comparing by each object would not be realistic.

This could be one example to work around and what would involve:

1. Collecting all the objects in full path from two files.
2. Remove the object path to be ignored.
3. Come up with scripting to compare by each object by using the above list as input. This script should be able to alarm if any difference is found.

Even this could be possible conceptual procedure to work around; it requires multiple passes of comparison which lead performance issue. It also causes inconvenience to the user.

In such cases, ignoring object feature would provide effective solution to verify the equality of the two files.

An object can be a dataset, group or named datatype.

# User interface

## Name of the new argument

* ***–ignore-path*** <path\_to\_object>  <FILE1> <FILE2>

The specified object <path\_to\_object> would be ignored from comparing <FILE1> and <FILE2>.

If <FILE1> **and**/**or** <FILE2> contain the object, h5diff will skip the object as if it doesn’t exist.

So if everything else is same except the ignored object, the exit code will indicate the two files are same.

## Excluding multiple objects

It’s necessary to provide ignoring multiple objects at once.

So for ignoring multiple objects at once could be:

* h5diff  ***--ignore-path*** “/g1/dset2”   ***--ignore-path*** “/g2/dset3”  … <FILE1>  <FILE2>

It may seem hassle to type the arguments repeatedly, but this will make simpler to construct a string line for automated scripting.

Specifying multiple objects followed by single “—ignore-path” could be considered, but it causes complications due to the complexity of naming convention and separating to individual name.

# Use cases

## Case1: Excluding object(s) from identical files

A user has two HDF files and knows the two files contain same datasets. However one dataset has different values and the user knows the particular dataset is design to be different. The user wants to verify if the two files are same. The user can use this feature to specify ignoring object(s) through a new argument(s).

Assumptions:

Both files contain same number, name and structure of objects. All the objects are expected to be identical except two objects.

* Objects in FILE1 : A, B, C, D, E
* Objects in FILE2 : A, B, C, D, E
* Object C and E is known as different

To verify all other objects are identical except the object C and E.

>> h5diff ***–ignore-path*** “C” ***--ignore-path*** “E” FILE1 FILE2

The h5diff will return exit code indicates both files are identical.

## Case2: Excluding object(s) from one file

A user has two HDF files. Assume one file contains extra object(s). Besides the extra object(s), the rest objects are expected to be identical. The user wants to compare only the identical objects by excluding the extra object(s) from one file to verify the equality of the two files.

Assumptions:

Both files contain same name and structure of objects. However one file contains extra objects compare to the other. Overlapping objects are expected to be identical.

* Objects in FILE1: A, B, C, D, E
* Objects in FILE2: A, B, C, D, E, F

To verify all other objects are identical except F

>> h5diff ***–ignore-path*** “F” FILE1 FILE2

The h5diff will return exit code indicates both files are identical.

## Case3: Excluding object(s) from both file

A user has two HDF files. Assume one file contains object A,B,C,D and the other file contains object B,C,D,E. The B,C and D are expected to be identical. The user wants to compare only the B,C and D by excluding A and E from both files to verify the equality of the two files.

Assumptions:

Both files contain same name and structure of objects. However both files contain extra objects with different name and values. Overlapping objects are expected to be identical.

* Objects in FILE1: A, B, C, D
* Objects in FILE2: B, C, D, E

To verify all other objects are identical except A and F

>> h5diff ***–ignore-path*** “A” ***--ignore-path*** “E” FILE1 FILE2

The h5diff will return exit code indicates both files are identical.

## Case3: Excluding hard link(s)

A user has two HDF files. Assume one file contains object A,B,C,D and the other file contains object A,B,C,D,E. However ‘E’ was added as a hard link to the object ‘D’. The user wants to compare the two file by excluding ‘E’ to verify the equality of the two files.

Assumptions:

Both files contain same name and structure of objects. However a file contains extra objects as hard link to one of the same objects.

* Objects in FILE1: A, B, C, D
* Objects in FILE2: A, B, C, D, E (hard line to D)

To verify all other objects are identical except E

>> h5diff ***–ignore-path*** “E” FILE1 FILE2

The h5diff will return exit code indicates both files are identical.

# Functional details

The ‘—ignore-path’ option will ignore only the given ‘path\_to\_object’.

To be clear, handling with hard link and symbolic link are described below in detail.

## Handling with hard link

Assume an object ‘A’ is pointed by has HARD link ‘B’ and ‘C’. If a user specify object ‘A’ to be ignored with ‘—ignore-path’ option, the h5diff will ignore only the object ‘A’ but not ‘B’ or ‘C’.

Also if a user specify object ‘C’ to be ignored with ‘—ignore-path’ option, the h5diff will ignore only the object ‘C’, not ‘A’ or ‘B’.

## Handling with symbolic link

Symbolic link means soft or external link.

If an object is pointed by a symbolic link(s), and the (source) object name is given to be ignored, h5diff will only ignore the object not the symbolic link(s). In this assumption, if the symbolic link is given to be ignored, h5diff will only ignore the symbolic link not the source object.

However a user can use the ‘—follow-links’ option to prevent the h5diff from comparing the symbolic link(s) by content, so the user can manipulate the options for the expected result.

# Optional features to consider in the future

These are gathered from internal meeting and feedbacks. Even these will not be considered as the initial targets to be implemented, it’s recorded here to be referred in the future as necessary.

* ‘—ignore-obj’: if an object relate to hard link(s) is specified, all the objects will be ignored.
* Use configuration file for specifying multiple paths, which can be too long to fit in one command line.
* Support multiple paths with one invocation of the flag. This feature will need its own plan among tools and other assumption to be made. (ex: reserve special characters)
* Allow wild card in given name – The issue is currently any special character can be part of name. This feature will need its own plan among tools and other assumption to be made. (ex: reserve special characters)
* An argument option to indicate equal exit code, if the all the candidated objects are same. (this feature is subset of ignoring object feature, so if the code path are same we may consider it)

# Revision History

|  |  |
| --- | --- |
| *May 13, 2010:* | Version 1 circulated for comment within The HDF Group. |
| *May 17, 2010:* | Version 1.2 updated from the internal meeting and feedbacks within the HDF Group. |
|  |  |
|  |  |