HDF5 File and Object Differences Specification

Peter Cao
Frank Baker

This document (or specification) describes the differences or equivalence relation of two HDF5 files or two HDF5 objects. The specification only handles HDF5 files and HDF5 objects. It does not apply to non-HDF5 files and non-HDF5 objects, which include a user block in an HDF5 file. The intended purpose of the document is to provide guidelines or information for developing tools or APIs for comparing HDF5 files or objects.

1 Introduction

The differences or equivalence relation of two HDF5 files or objects have not been defined. Lack of such a specification caused confusion in applications such as h5diff. The purpose of this document is to provide a clear definition on how two HDF5 files or objects should be compared.

An HDF5 file appears to the user as a directed graph. There are three higher-level objects that are exposed by the HDF5 APIs: groups, datasets, and named datatypes. The simplicity of the HDF5 model provides a great flexibility on what things can be put in a file. In the mean time, it also creates challenges on how to compare two files or two objects. The intention of this specification is to clarify the confusion and provide information for developing tools or APIs for comparing HDF5 files or objects.

The specification described in this document is intended for HDF5 files or objects only. It does not apply to non-HDF5 files and objects, which include a user block in an HDF5 file.

2 Descriptions of HDF5 files and objects

This section briefly describes high level HDF5 objects and their metadata. For details, please read the HDF5 File Format Specification at http://www.hdfgroup.org/HDF5/doc/H5.format.html and other related documents at http://www.hdfgroup.org/HDF5/doc/index.html.

2.1 Primary objects

HDF5 files are organized in a hierarchical structure, with two primary structures: groups and datasets.

- ***HDF5 group***: a grouping structure containing instances of zero or more groups or datasets, together with supporting metadata.

- ***HDF5 dataset***: a multidimensional array of data elements, together with supporting metadata.

An object in HDF5 is identified by its full path (path + name) when two objects are compared. Although each object has a unique identification number, which is used to identify objects under the lower level, the object is recognized by its full path not the ID as to users.

2.2 Metadata

HDF5 files generally contain two types of metadata: structural metadata and application metadata.

Structural metadata is generated by the HDF5 Library to describe the structure of the file and structure and contents of objects in the file. For example, structural metadata includes information such as:

- A header block (superblock) that sets up the file, sets up the initial structures, and identifies the file as a valid HDF5 file
- B-trees that describe the location of and provide access to groups and members of groups
- Datatype, current and maximum array dimensions, and other features of a dataset
- Dataset properties such as storage layout, fill value, allocation time, or the use of filters

HDF5 natively interprets and understands structural metadata. Structural metadata is always present; even an otherwise-empty file must contain certain metadata to be a valid HDF5 file.

Application metadata is defined and provided by a user application, is often stored in an HDF5 attribute, and may describe virtually anything. For example:

- Minimum and maximum valid values in a dataset
- Conditions under which data was collected
- Data history and/or provenance
- Relationships among datasets
- Scales or other interpretive information

HDF5 does not natively understand application metadata; it must be understood and interpreted by the application. Application metadata is technically optional but commonly used.

In both cases, these are just a few examples illustrating the range of possibilities.

3 Definition of equivalence

In this section, we will start a general definition of equivalence relation and its properties. Later, HDF5 equivalence relation will be introduced. Two files or objects are different if they are not equivalent.

3.1 Mathematic equivalence

"A given binary relation "~" on a set A is said to be an equivalence relation if and only if it is reflexive, symmetric and transitive." based on the definition of mathematical equivalence relation from Wikipedia. Equivalence relation has three basic properties:

- Reflexivity: “a” ~ “a”
- Symmetry: if “a” ~ “b” then “b” ~ “a”
- Transitivity: if “a” ~ “b” and “b” ~ “c” then “a” ~ “c”

3.2 HDF5 equivalence

A single definition of equivalence or differences of two HDF5 files or two HDF5 objects is not adequate for all situations. Two definitions of equivalence are proposed here: strictly equivalent and loosely equivalent.

3.2.1 Strictly equivalent

Two objects or files are strictly equivalent if their content (all values or members) and metadata are the same. The three properties of the equivalence relation should be applied under strictly equivalent.

Under strict equivalence, two files are equal if their file structures, objects (all groups and dataset), and metadata are the same. Two groups are equal if structures, objects (all groups and dataset), and metadata are the same. Two datasets are equal if their data values and metadata are the same.

3.2.2 Loosely equivalent

Two objects or files are loosely equivalent if they are equivalent under certain given conditions or options. The three properties of the equivalence relation may not be applied to loosely equivalent.

Below are some examples of comparison options for loosely equivalent. Other comparison options can be specified by applications.

Option A): Excluding certain objects when comparing two files or two groups

Option B): Comparing only common objects

Option C): Ignoring attributes or other metadata when comparing two objects

Option D): Comparing only attributes or other metadata

4 Comparing HDF5 objects

This section describes how two HDF5 objects should be compared. Options may be applied for loosely equivalent.

4.1 Group hierarchy

When two files are compared, the file structure will be compared along with all the objects in the file. Since all HDF5 files start at the root group, comparing two files would be the same as comparing the root groups of the two files. Based on the HDF5 file's group hierarchy five general cases for comparing two files are given below:

A) Itself: there should be no difference if a file is compared to itself.

B) Identical files: there should be no difference if two identical files are compared. This case is the same as case A) except that the two file are two separate physical files in the system.

C) Empty files: a file is empty if it contains only a root group. There should be no differences if two empty files are compared. If an empty file is compared with a non-empty file, the result varies according the comparison options. Under strictly equivalent, an empty file and a non-empty file should be different. Under certain loosely equivalent, such as comparing only common objects, there will be no difference.

D) Files with identical structure: all objects will be compared if two files have identical group hierarchy. The result will be determined by the comparison of individula objects.

E) Files with different hierarchical structure: all common objects will be compared. Under strictly equivalent, other objects will be reported as different objects. Under loosely equivalent, such as option B) of section 3.2.2, other objects will be ignored.

Metadata will also be compared when two groups are compared unless ignoring metadata options is given. Metadata of a group includes:

- Creation order, group layout, and other structural metadata
- Attributes attached to the group

4.2 Datasets

An HDF5 dataset is an object that contains raw data and metadata that describes the data elements, data layout, and all other information necessary to write, read, and interpret the stored data. For details, read the HDF5 user's guide at http://www.hdfgroup.org/HDF5/doc/UG/.

Comparing datasets means comparing both the raw data and metadata unless certain option is given.

4.2.1 Metadata

The metadata of a dataset includes:

- Datatype, dataspace, storage layout, compression filters, and other structural metadata
- Attributes attached to the datasets

Options can be given to ignore any or all the metadata when comparing two datasets.

4.2.2 Raw data

When two dataset are compared, the values of data arrays will be examined and compared. Special values will take special account.

- Floating points: to determine if two floating point values, float1 and float2, are different, one cannot use the simple comparison of (float1 == float2). Two floating point values can be the same while (float1 == float2) may show different because of the precision of floating points. Limit of precision needs to be set when comparing floating point values. For details, see RFC at https://www.hdfgroup.uiuc.edu/RFC/HDF5/tools/h5diff/RFC_h5diff_default_epsilon.pdf.
- Not-a-Number (NaN): two NaNs are equal. A NaN and a regular number are different.
- Infinity: infinity should be treated as a regular number. Two infinity numbers with the same sign (+/-) are equal. Two infinity numbers with different signs (+/-) are different. An infinity number and a regular number are different.

- Special datatypes: comparing values of special datatypes such as opaque and variable length can be complicated and need to be handled case by case.

4.2.3 Non-Comparable datasets

Two dataset may be non-comparable. Examples of non-comparable datasets include:

- Different datatype classes, e.g. H5T_TIME and H5T_COMPOUND
- Different dataspace (ranks and dimension sizes)
- Different order properties
- Different sign properties
- Invalid numeric operation in relative error calculation

4.3 Named datatypes

HDF5 allows one to define many different kinds of datatypes. There are two categories of datatypes: atomic datatypes and compound datatypes. Named datatypes are either atomic or compound datatypes that have been specifically designated to be shared across datasets. For more information, see the HDF5 user's guide at http://www.hdfgroup.org/HDF5/doc/UG/ .

The HDF function, H5Tequal(…), is sufficient to determine if two datatypes are equal. If two datatypes are different, it will require other library functions calls to retrieve the details of the two datatypes.

4.4 Attributes

An HDF5 attribute is a small metadata object describing the nature and/or intended usage of a primary data object. A primary data object may be a dataset, group, or named datatype. Attributes of two objects will be compared according to the following:

- Attributes will be compared by their names and values
- Dataspace and datatype of two attributes will be compared unless option of ignoring metadata is given.

5 h5diff

h5diff is a command line tool that compare two HDF5 files or objects and report the differences. The tool provides options on what to compare and how to compare. For details, see the online guide at http://www.hdfgroup.org/HDF5/doc/RM/Tools.html#Tools-Diff.

5.1 Major issues of h5diff

This section briefly describes some major issues regarding h5diff. It is not meant to be comprehensive.

5.1.1 Performance

Performance problem has been a known issue for h5diff. It is under investigation. One of the causes is from handling NaN. Current h5diff checks NaN values by default. The operation of checking and

comparing special values is very time consuming. For datasets without special values, the time was totally wasted. "-N" option is provided so that users can skip checking NaN.

Another performance issues in h5diff is retrieving and checking information of datatypes for each data point. This can be a major problem for datasets withlarge number of data points since checking datatypes information, such as H5Tequal() and H5Tget_member_type() are very expensive.

5.1.2 File structure

Current h5diff does not provide options for strictly equivalent and loosely equivalent. h5diff compares objects ONLY if it finds common objects in both files. If one of the files is an empty hdf5 file, h5diff concludes there is nothing to compare and since no difference has been found, it exits with 0. This is against common practice. E.g., the Unix tool, diff, considers the two files are different if ONLY one of them is an empty file. The h5diff behavior will also hide potential error in the testing of the h5copy or h5repack tools since if h5copy generates an empty hdf5 file by mistake, h5diff, as is, will report the error empty file no different from the original file. Therefore, h5diff should report an empty hdf5 file is different from a non-empty hdf5 file.

A proposal was made to add a new option "-c" as "Contents mode. Objects in both files must match". In view of the common practice of other comparison tools (e.g., the cmp and diff tools in Unix systems), it is better to fix h5diff as described in the above paragraph than introducing a new flag.

5.1.3 Non-Comparable datasets

h5diff does not compare some dataset objects, for a variety of reasons. When this happens, h5diff prints "Some objects are not comparable" at the end of the program execution. In verbose mode, h5diff also prints the reason(s) why it did not perform the comparison. The following are examples of non-comparable datasets listed in the RFC on non-comparable objects. For details, see the RFC at https://www.hdfgroup.uiuc.edu/RFC/HDF5/tools/h5diff/RFC_h5diff_NonComparable.pdf.

- Empty datasets
- Different datatype classes or H5T_TIME class or H5T_COMPOUND class
- Different dataspace ranks
- Different dataspace dimensions
- Different order properties
- Different sign properties
- Invalid numeric operation in relative error calculation

Revision History

October 8, 2010: Version 1 circulated for comment within The HDF Group.