RFC: A Tool to Handle HDF5 File Format Compatibility for Chunked Datasets

Vailin Choi

The HDF5 1.10 and 1.8 libraries differ in the way chunked datasets are indexed. Thus applications built with the HDF5 1.8 library cannot read chunked datasets in a 1.10 HDF5 file generated with the latest format option[[1]](#footnote-1). To mitigate the compatibility issue, we propose a tool, *h5format\_convert*, to do format conversion for files created with different HDF5 library formats. This RFC describes the current implementation of this tool, which handles format conversion for chunked dataset objects.

# Introduction

The HDF5 1.10 and 1.8 libraries differ in the way chunked datasets are indexed. For a 1.10 HDF5 file generated with the latest format option, the 1.10 library uses one of the following indexing types depending on a chunked dataset’s dimension specification and the way it is extended:

* Extensible array indexing for appending along a specified dimension
* Version 2 B-tree indexing for appending along multiple dimensions
* Fixed array indexing for fixed-size datasets
* Implicit indexing for fixed-size datasets with early space allocation and without filters

These indexing types improve performance for writing and reading chunked datasets. Also, the latest file format is required when using SWMR (single-writer/multiple-readers) access.

On the other hand, the HDF5 1.8 library only supports one indexing type—version 1 B-tree indexing type for chunked datasets.

The incompatibility described above will disallow 1.8 library-based applications to read a chunked dataset in a 1.10 library-based file generated with the latest file format. One possible workaround is to use the existing *h5repack* tool. By default the tool rewrites an object in the file using the earliest file format in which the object became available thus making 1.10 library-based file readable by the HDF5 1.8 library. Unfortunately, this is not an efficient solution for large files.

This RFC describes a new tool, *h5format\_convert*, which intends to do format conversion for objects in an HDF5 file. Currently, the tool only handles downgrade conversion for dataset object. It will convert the indexing type of a chunked dataset in an HDF5 file with the latest file format to the indexing type that can be handled by the 1.8 library release. The conversion is performed in place, i.e., without rewriting the file. When conversion is done, the HDF5 1.8 library can access the dataset in the modified file. Please note that further modifications of the dataset could be done according to the 1.8 file format only. For example, a dataset written by a SWMR application cannot be reopened and modified by another SWMR application after conversion was performed.

# The Tool

## Pre-conversion

User creates an HDF5 file with the 1.10 library’s default sec2 driver, using the latest format option. This is a case for files created for SWMR access.

When the user creates and writes a chunked dataset in the file:

* The dataset will employ either one of the following indexing types:
* Extensible array indexing
* Version 2 B-tree indexing
* Fixed array indexing
* Implicit indexing
* The dataset’s layout information in the object header is described by a pair of messages:
* Version 4 *layout* message
* Version 0 *storage* message

The file will be modified as below:

will be larger

## Conversion

The tool will invoke internal library routines to convert the indexing type of the specified chunked dataset to version 1 b-tree indexing type. The routine will traverse the object header of the dataset to perform the following:

* Create version 1 B-tree indexing for the dataset.
* Create version 3 *layout* message to point to the version 1 B-tree.
* Retrieve the set of chunk addresses *K* from the indexing information in the *storage* message
* Insert the set of chunk addresses *K* into the version 1 B-tree.
* Delete the *storage* and the version 4 *layout* messages from the object header without deleting file space for the chunks.

1

## Command line syntax

***h5format\_convert [OPTIONS] file\_name***

The tool will iterate through all the datasets in the file *file\_name* and convert all chunked datasets whose indexing type is not version 1 B-tree. While iterating through the datasets, if a dataset is not chunked or its indexing type is already version 1 B-tree, nothing will be done but just continue with the iteration. When errors are encountered during the iteration, no further conversion is performed and the tool will exit with failure.

The available options are:

|  |  |
| --- | --- |
| *-h [--help]* | The tool will print a usage message and exit with success. |
| *-V [--version]* | The tool will print the version # and exit with success. |
| *-v [--verbose]* | This will enable the verbose mode. The tool will print the steps being done while converting a dataset. |
| *-d <dname> [--dname=<dataset\_name>]* | This is the pathname of the dataset to be converted.When the dataset is not chunked or the indexing type is already version 1 B-tree, the tool will not perform any conversion and will exit with success. |
| *-n [--noop]* | A noop. No file modification.The tool will perform all the steps except the actual conversion and exit with success. When errors are encountered along the way, the tool will exit with failure. |

# Internal Library Routines

The library provides internal routines to support the tool in doing the conversion. It is uncertain at this point whether they will become public routines.

## H5Oformat\_convert

**Name:**

H5Oformat\_convert

**Signature:**

herr\_tH5Oformat\_convert *(hid\_t loc\_id, const char \* objectname, hid\_t object\_convert\_plist)*

**Purpose:**

Converts the format of an object in an HDF5 file.

**Description:**

This routine is currently not implemented but is provided here for future enhancement. More work is required to describe the full functionality of this routine.

H5Oformat\_convert will convert the object *objectname* associated with *loc\_id* which will be a file or group identifier. Objects in an HDF5 file are groups, datasets and named datatypes.

object\_convert\_plist is the object conversion property list with properties to determine the type of conversion to be performed. Properties might be as follows:

* H5O\_OBJ\_CONVERT\_DOWNGRADE
	+ To downgrade the format of the object, e.g. from 1.10 library release to 1.8 release
* H5O\_OBJ\_CONVERT\_UPGRADE
	+ To upgrade the format of the object, e.g. from 1.8 library release to 1.10 release

**Parameters:**

|  |  |
| --- | --- |
| *hid\_t* loc\_id*const char \**objectname*hid\_t* object\_convert\_plist | IN: Location identifier where the object resides.IN: The object nameIN: The object conversion property list |
|  |  |

**Returns:**

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

## H5Dconvert\_chunk\_index\_type

**Name:**

H5Dconvert\_chunk\_index\_type

**Signature:**

herr\_tH5Dconvert\_chunk\_index\_type*(hid\_did)*

**Purpose:**

Converts a chunked dataset’s indexing type to version 1 B-tree.

**Description:**

This routine is currently invoked by the tool *h5format\_convert* to perform downgrade conversion for a chunked dataset. It will be incorporated later into *H5Oformat\_convert()*.

H5Dconvert\_chunk\_index\_type converts the chunk indexing type for the dataset associated with *did* to version 1 B-tree indexing type. The dataset has to fulfill the following conditions:

* The dataset is chunked.
* The dataset’s chunk indexing type is not version 1 B-tree. That is, the indexing type is one of the following:
	+ Extensible Array
	+ Version 2 B-tree
	+ Fixed Array
	+ Implicit

If the above conditions are not satisfied, the routine will not perform any conversion but will return SUCCESS.

**Parameters:**

|  |  |
| --- | --- |
| *hid\_t* did | IN: Dataset identifier. |
|  |  |

**Returns:**

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

## H5Dget\_chunk\_index\_type

**Name:**

H5Dget\_chunk\_index\_type

**Signature:**

herr\_tH5Dget\_chunk\_index\_type *(hid\_t* did*, H5D\_chunk\_index\_t \**idx\_type*)*

**Purpose:**

Retrieves a dataset’s chunked indexing type.

**Description:**

H5Dget\_chunk\_index\_type retrieves the chunked indexing type for the dataset associated with *did*. It will return error if the dataset is not chunked.

**Parameters:**

|  |  |
| --- | --- |
| *hid\_t* did*H5D\_chunk\_index\_t \*idx\_type* | IN: Dataset identifier.OUT: The chunk indexing type. Possible types are: H5D\_CHUNK\_IDX\_BTREE H5D\_CHUNK\_IDX\_NONE H5D\_CHUNK\_IDX\_FARRAY H5D\_CHUNK\_IDX\_EARRAY H5D\_CHUNK\_IDX\_BT2 |
|  |  |

**Returns:**

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

# Future Enhancement

## Upgrade

Add a new option *-u [--upgrade]* to the tool. This will upgrade a chunked dataset’s indexing type to the latest format type like extensible array, version 2 B-tree, fixed array or implicit.

The work involved in adding this option:

* Decide on the indexing type to upgrade based on the dataset’s dimension specification
* Handle the actual conversion in the internal library routine based on the indexing type
* Debugging, testing and adding new tests
* Update reference manual entry

The estimated time for doing the above work is roughly 40 hours.

## Conversion for other objects

Expand the tool to perform upgrade or downgrade for other HDF5 objects in the file, i.e. groups and named data types. More time is needed to investigate the full extent of this work.

Acknowledgements

This work was supported by The HDF Group internal maintenance project (GMQS).

Revision History

|  |  |
| --- | --- |
| *March 26, 2015:* | Version 1 circulated for comment within The HDF Group.  |
| March 31, 2015May 5, 2015May 13, 2015 | Version 2 sent to DLS with the source code.Version 3 circulated within the HDF Group.Version 4 incorporated review comments. |
|  |  |
|  |  |
|  |  |
|  |  |

References

1. HDF5 File Format Specification, The HDF Group, <http://www.hdfgroup.org/HDF5/docNewFeatures/NewFeaturesReferenceDocs.html>
2. RFC: Options to handle compatibility issues for HDF5 files,

<http://svn.hdfgroup.uiuc.edu/hdf5doc/trunk/RFCs/HDF5/tools/compat_tool/>

1. See the H5Pset\_libver\_bounds function <http://www.hdfgroup.org/HDF5/doc/RM/RM_H5P.html#Property-SetLibverBounds> [↑](#footnote-ref-1)