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

Vailin Choi

To improve performance for writing and reading chunked datasets and to enable new features like SWMR, the new chunk indexing types were introduced in the HDF5 1.10 library when the file is generated using the latest format option. Thus applications built with the HDF5 1.8 library cannot read chunked datasets that use new indexing types. To mitigate the compatibility issue, we propose a tool, *h5format_convert*, to convert a chunked dataset's indexing type in a 1.10 HDF5 file to the indexing type that can be handled by the HDF5 1.8 library.

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 file format, 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

The dataset layout information in the object header is described by a pair of messages:

- Version 4 *layout* message
- Version 0 *storage* message

The HDF5 1.8 library only supports version 1 B-tree indexing type for chunked datasets. The dataset layout information in the object header is described by a single message: version 1, 2, or 3 *layout* message.

We refer the reader to the latest File Format Specification [1] for description of the messages and layouts.

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 library format. One of the possible workarounds is to use the *h5repack* tool. By default the tool rewrites an object in the file using the earliest file format version 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 the large files.

This RFC describes a tool to convert the chunk indexing types for datasets in 1.10 library-based files generated with the latest format, to version 1 B-tree indexing types. The conversion is performed in place, i.e., without rewriting the file. When conversion is done, the datasets in the modified file can be accessed by the HDF5 1.8 library. Please notice that further modifications of those datasets 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

The new tool is called *h5format_convert*. It will change a chunked dataset's indexing type to version 1 B-tree for an HDF5 file.

Usage

```
h5format_convert [OPTIONS] file_name
```

-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.
-a [--all]	The tool will iterate through all the datasets in the file and convert all chunked datasets whose indexing type is not version 1 B-tree. If both *-a* and *-d* options are given on the command line, *-a* option will take precedence. When errors are encountered during the iteration, no further conversion is done and the tool will exit with failure.
-n [--dry]	A dry run. 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.

1)

Two Internal Library Routines

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

H5Dformat_convert

Name:

`H5Dformat_convert`

Signature:

herr_t `H5Dformat_convert` (*hid_t* `did`)

Purpose:

Converts a dataset's chunk indexing type to version 1 B-tree.

Description:

`H5Dformat_convert` 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_t `H5Dget_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 the dataset identifier, `did`. It will return error if the dataset is not chunked.

Parameters:

hid_t `did`	IN: Dataset identifier.
*H5D_chunk_index_t *idx_type*	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.

Example Usage

The example below illustrates the usage of these two internal routines:

```
hid_t did;              /* Dataset identifier */
H5D_chunk_index_t type; /* Chunk indexing type */

/* Open a file */
:
:
:
/* Open a dataset in the file */
did = H5Dopen(fid, dname, H5P_DEFAULT);

/* Convert the dataset’s chunk indexing type */
if(H5Dformat_convert(did)) < 0)
      printf(“ERROR in converting\n”);

/* Retrieve the dataset’s chunk indexing type */
H5Dget_chunk_indexing_type(did, &type);
/* type is H5D_CHUNK_IDX_BTREE */
:
:
:
```

Future Enhancement

A new option *-u [--upgrade]* might be a useful addition. 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.

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, 2015	*Version 2 sent to DLS with the source code.*
May 5, 2015	Version 3 circulated within the HDF Group.

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/