**The current design implementation**

**of the filter for modifying data set’s data type**

by **Raymond Lu**

8 October 2008

For the background information and basic design of this filter, please read the RFC document at ????? The implementation of this filter is based on v1.8.0 of the library. It is in the SVN branch at http://svn.hdfgroup.uiuc.edu/hdf5/branches/modify\_dtype\_filter/.

1. **The Filter**

The filter basically performs two functions – reading and writing data (see the function H5Z\_filter\_dtype\_modify in H5Zdtype\_modify.c). During write, the information of the current data set’s data type is encoded in the beginning of the data chunks. Whether the data type is the original or a new one being changed to through H5Dmodify\_dtype is not a concern here. To save the space, the filter only allows the committed, shared, or predefined data types to be written to the chunks. Somehow in the new file format, the library makes a copy of the predefined data type and uses the copy as the data set’s type (see H5D\_init\_type in H5Dint.c). Fortunately, all predefined data types are either integers or floating-point numbers. Their copied data types do not occupy much space, either. Thus, the filter allows them, too.

During read, the information of the data type encoded in the beginning of the chunk is decoded.

This data type is compared with the current data set’s data type. The current data set’s data type is saved in the client data (passed in to the function H5Z\_filter\_dtype\_modify in the parameter cd\_values). If these two data types are different, the filter will convert the data to the current data type and pass it up to the next level.

The filter uses a skip list to save the information of the data types of the data chunks. The use of the skip list is only for managing the reference count of the data types for the data chunks. It only resides in the memory. When the data chunk is being read, the filter inserts the data type for this chunk into the skip list. The searching key for the skip list node is the offset of the chunk. This offset has been translated from multiple dimensions to single dimension. When the data chunk is written to the file, the filter first search in the skip list to see whether there is a data type saved in the skip list for this chunk. If there is one, the filter will decrement its reference count and close it. The filter will also insert the new data type and increment its reference count in the skip list. If there is no data type saved for this chunk in the skip list, the filter will simply insert the new data type and increment its reference count. This skip list is created in H5Z\_set\_local\_dtype\_modify when the data set is created. Its address is saved in the client data (passed in to the function H5Z\_filter\_dtype\_modify in the parameter cd\_values).

The filter uses a B-tree to store the information of the data types and their reference counts for the data chunks. This B-tree is saved in the file for the convenience of deleting the data set. When the data set is deleted, the filter will go through the B-tree to decrement the reference counts for the data types. The insertion of the data types and their reference counts happens during data writing. The filter first finds the old data type for the chunk in the skip list. Then it decrement the reference count for the old data type in the B-tree. Finally, it inserts the new data type into the B-tree and increments its reference count. If the chunk information is not even in the skip list, the filter will simply insert the new data type into the B-tree and increment its reference count. This B-tree is also created in H5Z\_set\_local\_dtype\_modify when the data set is created. Its address is saved in the client data (passed in to the function H5Z\_filter\_dtype\_modify in the parameter cd\_values).

There are a few new private functions for the filter pipeline as follows:

H5Z\_reset\_local

H5Z\_change\_local

H5Z\_evict\_local

H5Z\_delete\_local

H5Z\_close\_local

They were added mainly for this filter of modifying data set’s data type. The filter’s corresponding callback functions to them are as follows:

H5Z\_reset\_local\_dtype\_modify

H5Z\_change\_local\_dtype\_modify

H5Z\_evict\_local\_dtype\_modify

H5Z\_delete\_local\_dtype\_modify

H5Z\_close\_local\_dtype\_modify

Please see the detailed descriptions for these functions in the comments in H5Zdtype\_modify.c.

1. **The Functions**

Two public functions have been added to the library - H5Pset\_dtype\_modifiable and H5Dmodify\_dtype. Please see the reference manual for how to use them. When the function H5Dmodify\_dtype is called, if the dataset is chunked with the filter disabled or if the dataset is contiguous or compact, the library will change the data type and convert the data in the file immediately. The way that the library does it (in H5D\_modify\_dtype in H5Dio.c) is to create a bogus dataset, read and convert and write the data from the original dataset to this bogus dataset, and assign the ownership of the raw data of this bogus dataset to the original dataset. Then the library will delete the metadata of the bogus dataset and modify the metadata of the original dataset. In the process of reading, converting, and writing the data, the library uses a buffer of limited size to avoid large data operations in the memory (see H5D\_contig\_copy\_conv in H5Dcontig.c and H5D\_istore\_copy\_conv in H5Distore.c). If the filter is enabled and the dataset is chunked, the library will postpone the data conversion until the data is read or written. But the data chunks already in the cache will be converted when H5Dmodify\_dtype is called (see H5D\_modify\_dtype in H5Dio.c).

1. **The Issue Left**

There is one issue left to implement in the library. In the filter file H5Zdtype\_modify.c, the reference count for the data type is supposed to be adjusted for both committed and shared data type. The current implementation only adjusts for committed type. The shared data type should also be adjusted.