RFC: Chunk query functionality in HDF5

Elena Pourmal

Quincey Koziol

This document proposes of HDF5 function chunk sizeslocations inHDF5 file and information about filters applied to each data chun he functions will become part of the HDF5 C library and will be released in the HDF5 maintenance release .

Introduction

HDF5 application developers expressed interest in reading a data chunk from HDF5 file without HDF5 library APIs. The requests are documented in the JIRA database (see HDFFV-9935, HDFFV-10089, HDFFV-8487) To enable this an application know the size of the chunk and the address at which can be found in the HDF5 file along with the flag that indicates which filters were applied to the chunk

This chunk query functionality can be very useful for building HDF5 map files like implemented for HDF4 (see https://support.hdfgroup.org/projects/h4map. Map files allow access to all data stored in the HDF5 files without using the HDF5 library.

HDF5 map file concept was employed in the implementation of the Architecture#2 prototype for accessing HDF5 data in Cloud via OPeNDAP serverAppendix 2, "Web Accessible APIs in the Cloud Trade Study" report performed under the ESDIS/Raytheon Task 28 EED-2 project Task 28 studies were supported by a prototype implementation of chunk query functions discussed in Section 2.

After completion of Task 28, ESDIS program POC asked for more studies on Architecture #2 and requested that The HDF Group EED2 team implements the HDF5 chunk query functionality in the mainstream HDF5 library. New functions should be provided to the OPeNDAP developers no later than Fall 2018 and released in the maintenance release by the end 2018.

equested chunk query functionality would also very useful the HDF Cloud HSDS solution as indicated by John Readey

Section 2 describes the existing functionality to find chunk sizes and filter flags and the work done by

Kent Yang to support chunk queries. Section 3 functions and Section 4 summarizes the recommendations.

Exiting functionality and prototype implementations

years some chunk query functionality was added to the HDF5 library and was prototypedunder Task 28. In this section, we will provide an overview of th work.

0.1 Getting the size of the stored chunk and filter flags

We added two of the requested queries to the HDF5 library when we introduced the functionality to read chunk with optimizedfunction `H5Dread_chunk`.Its parameter `filter_mask` indicates which filters are used with the chunk when written. A zero value indicates that all enabled filters are applied on the chunk. A filter is skipped if the bit corresponding to the filter's position in the pipeline (0 ≤ position < 32) is turned on.

The size of the chunk stored in the file can be obtained by using . The third parameter `chunk_nbytes` returns the size of the chunk in bytes as it is stored in the file.

There no public API chunk's address. The prototype created for Task 28 addresses some of .

0.2 Prototype implementation of chunk query functions

Under Task 28 sponsored by the EED2 project, Kent Yang implemented several functions to locate the chunks in the file. The source code is available in the hdf5_1_8_17_storage_info branch in the Bitbucket reporsitory https://bitbucket.hdfgroup.org/users/myang6/repos/hdf5/.

See the functions

herr_t H5Dget_dataset_chunk_storage_info(hid_t dset_id,H5D_chunk_storage_info_t chunk_st_array[],
unsigned int *num_chunk_dims_ptr)
and
herr_t H5Dget_dataset_storage_info(hid_t dset_id,uint8_t* layout_typeptr,hsize_t* num_chunkptr,uint8_t* storage_status_ptr)

in the H5Ddbg.c file and a data structure

```
typedef struct H5D_chunk_storage_info_t {
        uint32_t    nbytes;                     /* Size of stored data
*/
        hsize_t     chunk_offset[H5S_MAX_RANK+1]; /* Logical offset to
start */
        unsigned    chunk_filter_mask;          /* Excluded filters
*/
```

```
        haddr_t     chunk_addr;                 /* Address of chunk in
file */
} H5D_chunk_storage_info_t;
```

in the H5Dpublic.h file in that branch.

The example h5dsoreinfo.c in the examples directory of the branch shows the usage of the functions.

The prototype was implemented against 1.8.17. In HDF5 1.8.* series B-trees version 1 are used for chunks indexing. In HDF5 1.10.0 new chunk indexing schemas were introduced for datasets with unlimited and fixed size dimensions. See File Format Spec, Appendix C for more information.

Prototype implementation addressed the needs of Task 28. They were not created to be generalized enough to work with other use cases; for example, query the chunks that cover a specified hyperslab selection (i.e., a number of chunks that have nonempty intersection with a specified selection, and their logical coordinates, sizes, filter masks and addresses), or find out a selection that only contains chunks that exist in the file.

The next section introduces the functions that address those requirements too.

New functions to query chunks

This section proposes the functions to be added to the HDF5 C library

`H5Dget_chunk_info_by_coord` return information about an existing chunk or that chunk doesn't exist.

`H5Dget_num_chunks` number of the existing chunks in a specified selection

`H5Dget_chunk_info`returns information about the existing chunk specified by its index in the set of the existing chunks. The APIs signatures are “optimized” for creation of Fortran, Java and low-level h5py wrappers

`H5Dchunk_iterate` is added to go along with general HDF5 paradigm and can be very useful for application developers, for example, to implement statistics on the data chunks without passing data to the users to perform the same task.

`H5Dget_chunk_selection`will be handy when finding a region with all data written to a chunked dataset

Name: `H5Dget_chunk_info_by_coord`
Signature:

```
herr_t H5Dget_chunk_info_by_coord (hid_t dset_id, size_t *,
*filter_mask, haddr_t *addr)
```

Purpose:

Retrieves information about a chunk specified by its coordinates

Description:

The function retrieves filter mask, size and address in the file of a chunk specified by its coordinates. If chunk doesn't exist in the file, the size set to 0 and address to HADR_UNDEF. Value pointed by filter_mask not modified.

Parameters:

`hid_t dset_id`	IN: Dataset identifier
`hsize_t *`	IN: Pointer to a one-dimensional array of the size equal to the dataset's rank. element .
`*filter_mask`	OUT: The parameter indicates which filters were used with the chunk when written. A zero value indicates that all enabled filters are applied on the chunk. A filter is skipped if the bit corresponding to the filter's position in the pipeline (0 ≤ position < 32) is turned on. If chunk doesn't exist the value at the pointer location is not modified.
`hsize_t *size`	OUT: Chunk size in bytes; if chunk doesn't exist the size is set to 0.

Returns:

Returns a non-negative value if successful. Otherwise returns a negative value.

History: To be introduced in HDF5 1.10.; may be backported to 1.8 if requested by the customer.

Name: `H5Dget_num_chunks`

Signature:

```
herr_t H5Dget_num_chunks(hid_t dset_id, hid_t fspace_id,
hsize_t *nchunks)
```

Purpose:

Retrieves number of chunks that have nonempty intersection with a specified selection.

Description:

The function retrieves a number of chunks that have a nonempty intersection with the set of the selected elements (i.e. selection specified by `fspace_id`). If `fspace_id` is H5S_ALL, the function will retrieve the total number of the chunks stored for the dataset.

Parameters:

hid_t dset_id	IN: Dataset identifier
hid_t fspace_id	IN: File dataspace selection identifier; H5S_ALL if the selection is the current extent of the dataset
hsize_t * nchunks	IN/OUT: Number of chunks in the selection

Returns:

Returns a non-negative value if successful. Otherwise returns a negative value.

History: To be introduced in HDF5 1.10.; may be backported to 1.8 if requested by the customer.

Name: H5Dget_chunk_info

Signature:

herr_t H5Dget_chunk_info (*hid_t* dset_id, *hid_t* fspace_id, *hsize_t* index, *hsize_t* *, *filter_mask, *haddr_t* *addr)

Purpose:

Retrieves information about the chunk specified by chunk index.

Description:

The function retrieves coordinates, filter mask, size and address in the file of the chunk specified by index. The chunk belongs to a set of chunks that have nonempty intersection with dataset's file space selection specified by fspace_id.

Parameters:

hid_t dset_id	IN: Dataset identifier
hid_t fspace_id	IN: File dataspace selection identifier; H5S_ALL if the selection is the current extent of the dataset
hsize_t index	IN: Chunk index in the selection. Index value may have the value 0 to number of chunked stored in the file that have nonempty intersection with the filespace selection.
hsize_t *	OUT: Pointer to a one-dimensional array of the size equal to the dataset's rank. When function returns the array's elements contain 0-based
u *filter_mask	OUT: The parameter indicates which filters were used with the chunk when written. A zero value indicates that all enabled filters are applied on the chunk. A filter is skipped if the bit corresponding to the filter's position in the pipeline (0 ≤ position < 32) is turned on.
hsize_t *size	OUT: Chunk size in bytes

Returns:

Returns a non-negative value if successful. Otherwise returns a negative value.

History: To be introduced in HDF5 1.10.; may be backported to 1.8 if requested by the customer.

Name: H5Dchunk_iterate

Signature:

```
herr_t H5Dchunk_iterate(hid_t dset_id, hid_t fspace_id,
H5D_chunk_order_t order, hsize_t *idx, H5D_chunk_iter_cb_t *op, void
*op_data)
```

Purpose:

Iterates through the datasets chunks.

Description:

The function iterates through the chunks that have nonempty intersection with the dataset selection specified by `fspace_id`, in the order of the specified order, `order`, using a user-defined callback routine `op`. Passing H5S_ALL for `fspace_id` will iterate over all chunks in the dataset.

`Order` can be one of three values:

H5_CHUNK_ITER_ORDER_NATIVE	Native order of chunks as they stored in the traversed HDF5 data structure (see Appendix C); native order is the fastest one
H5_CHUNK_ITER_ORDER_COORD	Order of chunks in the linearized chunk coordinate space (we can give here formula or better a reference where it is described in UG? Tutorial?)
H5_CHUNK_ITER_ORDER_ADDR	Order of chunks sorted by their addresses in the file

The prototype of the callback function `op` is as follows:

int(*H5D_chunk_ iter_cb_t)(*const H5D_chunk_info_t* *info, *void* *op_data)

The parameters of this callback function have the following values or meanings:

info — *H5D_chunk_info_t* structure containing information regarding the chunk:

```
typedef struct H5D_chunk_info_t {
        hsize_t     offset[H5S_MAX_RANK]; /* Chunk
coordinates*/
        unsigned    filter_mask; /* Excluded filters*/
        uint32_t    size; /* Size of stored data */
        haddr_t     addr; /* Address of chunk in file */
} H5D_chunk_info_t;
```

op_data — User-defined pointer to data required by the application in processing the chunk; a pass-through of the `op_data` pointer provided with `H5Dchunk_iterate` function call

The return value should be H5_ITER_ERROR, H5_ITER_CONT, H5_ITER_STOP.

Parameters:

hid_t dset_id	IN: Dataset identifier
hid_t fspace_id	IN: File dataspace selection identifier; NULL if the selection is the current extent of the dataset
H5_chunk_order_t order	IN: Chunk iteration order; can be H5_CHUNK_ITER_ORDER_NATIVE, H5_CHUNK_ITER_ORDER_COORD, and H5_CHUNK_ITER_ORDER_ADDR.
hsize_t *idx	IN: Iteration index position at which to start OUT: Position at which an interrupted iteration may be restarted
H5D_chunk_iter_cb_t *op	IN: Callback function passing data regarding the chunk to the calling application
void *op_data	IN: User-defined pointer to data required by the application for its processing of the chunk

Returns:

Returns a non-negative value if successful. Otherwise returns a negative value.

History: To be introduced in HDF5 1.10.; may be backported to 1.8 if requested by customer.

Name: `H5Dget_chunk_selection`
Signature:

```
hid_t H5Dget_chunk_selection (hid_t dset_id)
```

Purpose:

Returns a dataspace identifier with a selection for all existing chunks in the dataset.

Description:

The function constructs a dataspace with selection that is a union of the selections. Each selection in the union contains an existing chunk.

Parameters:

`hid_t dset_id`	IN: Dataset identifier

Returns:

Returns an identifier for a dataspace with a selection for all existing chunks in the file if successful. Otherwise returns a negative value.

History: To be introduced in HDF5 1.10.; may be backported to 1.8 if requested by the customer.

Test plan

Tests are to be added per Elena's test outline in

Recommendation

Acknowledgement

This work was supported by NASA/GSFC under Raytheon Co. contract number NNG15HZ39C.

Revision History

June 25, 2018:	Version 1 circulated for comment.
June 29, 2018	Version 2 contains Quincey's suggestions and edits; sent to the group