RFC: Chunk query functionality in HDF5

Binh-Minh Ribler

Elena Pourmal

Quincey Koziol

This document proposes implementation of the new HDF5 functions for discovering chunk sizes, their locations in HDF5 file and information about filters applied to each data chunk (chunk query functions). The functions will become part of the HDF5 C library and will be released in the HDF5 maintenance release in Fall 2018.

# Introduction

HDF5 applications developers expressed a lot of interest in reading a data chunk from HDF5 file without HDF5 library APIs. The requests are documented in the JIRA database (see [HDFFV-9935](https://jira.hdfgroup.org/browse/HDFFV-9935), [HDFFV-10089](https://jira.hdfgroup.org/browse/HDFFV-10089), [HDFFV-8487](https://jira.hdfgroup.org/browse/HDFFV-8487)). To enable this functionality an application should know the size of the chunk and the address at which it 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 server (see Appendix 2, “[Web Accessible APIs in the Cloud Trade Study](https://confluence.hdfgroup.org/pages/viewpage.action?pageId=47253862)” 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 on 2018.

Requested chunk query functionality would also very useful for the HDF Cloud HSDS solution as was indicated by John Readey.

The rest of the document is organized as follows. 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 discusses new proposed functions and Section 4 summarizes the recommendations.

# Exiting functionality and prototype implementations

During the last few years some chunk query functionality was added to the HDF5 library and was prototyped under Task 28. In this section, we will provide an overview of the previous work.

# 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 the functionality to read chunk with optimized function [H5DOread\_chunk](https://portal.hdfgroup.org/display/HDF5/H5DO_READ_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 [H5Dget\_chunk\_storage\_size](https://portal.hdfgroup.org/display/HDF5/H5D_GET_CHUNK_STORAGE_SIZE). The third parameter chunk\_nbytes returns the size of the chunk in bytes as it is stored in the file.

There is no public API that returns chunk’s address in the file, or an API that tells how many actual chunks are written in the file and what are their logical coordinates. The prototype created for Task 28 addresses some of those issues.

# 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/browse?at=refs%2Fheads%2Fhdf5_1_8_17_storage_info) <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][[1]](#footnote-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](https://bitbucket.hdfgroup.org/users/myang6/repos/hdf5/browse/examples/h5dstoreinfo.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](https://confluence.hdfgroup.org/display/HDF5/File+Format+Specification) 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.

The functions allow to query the number of chunks in a dataset selection, chunk’s storage size, filter mask, address in the file, and logical coordinates.

The H5Dget\_chunk\_info\_by\_coord function is designed to return information about an existing chunk and or provide a hint that chunk doesn't exist.

The H5Dget\_num\_chunks function should be used to find number of the existing chunks in a specified selection and the H5Dget\_chunk\_info function 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.

The iterator function H5Dchunk\_iterate is added to go along with general HDF5 paradigm and can be very useful for applications developers, for example, to implement statistics on the data chunks without passing data to the users to perform the same task.

The last function 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, *hsize\_t* \*coord, *uint32\_t* \*filter\_mask, *hsize\_t* \*size, *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 is set to 0 and address to HADR\_UNDEF. Value pointed by filter\_mask is not modified.

**Parameters:**

|  |  |
| --- | --- |
| *hid\_t* dset\_id | IN: Dataset identifier |
| *hsize\_t* \*coord | IN: Pointer to a one-dimensional array of the size equal to the dataset’s rank. The array’s elements contain 0-based logical coordinates of the chunk. |
| *uint32\_t* \*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. |
| *haddr\_t* \*addr | OUT: Chunk address in the file; if chunk doesn’t exist the address is set to HADR\_UNDEF. |

**Returns:**

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

**History:** To beintroduced in HDF5 1.10.4; 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 beintroduced in HDF5 1.10.4; 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* inde*x, hsize\_t* \*coord, *uint32\_t* \*filter\_mask, *hsize\_t* \*size, *haddr\_t* \*addr)

**Purpose:**

Retrieves information about the chunk specified by chunk index.

**Description:**

The function retrieves logical 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* \*coord | 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 logical coordinates of the chunk |
| *uint32\_t* \*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 |
| *haddr\_t* \*addr | OUT: Chunk address in the file |

**Returns:**

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

**History:** To beintroduced in HDF5 1.10.4; 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 beintroduced in HDF5 1.10.4; 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 beintroduced in HDF5 1.10.4; may be backported to 1.8 if requested by the customer.

# Recommendation

Summary with proposed work (functions implementation, testing, tools updates will be added in the next version)

# 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 |

1. Chunk will have the same dimensions as datasets with the maximum rank 32. It will not hurt to have a bigger array, but it is not necessary. It is very possible that +1 was introduced to accommodate an extra dimension for a chunk that is the size of the datatype (see file format spec for more info; h5debug does show extra dimension for the chunk, but it will be very confusing for the users in the public structure). [↑](#footnote-ref-1)