RFC: h5watch and 3 public routines for dataset manipulation

Vailin Choi

This RFC proposes a new HDF5 command line utility *h5watch* for users to dynamically monitor data elements appended to a dataset. It also introduces three high-level pubic routines for users to manipulate dataset operations.

# Introduction

The new command line utility *h5watch* allows users to see new data elements appended to a dataset as it grows. It has similar functionality like the Unix user command *tail –f* (--*follow*) option, which outputs appended data as the file grows.

The three new high-level public routines are: *H5LD\_get\_dset\_type\_size()*, *H5LDget\_dset\_dims()*, and *H5LDget\_dset\_elmts()*. They are convenient functions that condense user operations needed to handle a dataset.

# Approach

The development of *h5watch* will be in phases. The initial phase develops the tool’s basic features to dump appended data from one dataset. Subsequent phases will expand the dumping of data elements from multiple datasets and other desirable features. The command line syntax as well as the output format from the tool will follow the existing HDF5 utility *h5ls*. Development will be done in the *revise\_chunks* branch as it depends on the SWMR (single write/multiple readers) feature.

For the initial phase, this new tool applies only to chunked dataset with unlimited dimension(s) or fixed dimension with maximum dimension setting. Also, while monitoring a dataset, user needs to ensure data written to the dataset is flushed to the file.

The development of the three high-level public routines will also be done in the *revise\_chunks* branch and will be available in the initial phase together *h5watch*.

# h5watch

**The command line syntax for the tool is: *h5watch [options] [object]***

## Options

--help Print a usage message and exit

--version Print version number and exit.

--label Label members of compound typed dataset.

--simple Use a machine-readable output format.

--dim Monitor changes in size of the dataset dimensions only.

--width=N Set the number of columns to N for output.

A value of 0 sets the number of columns to the maximum (65535).

The default width is 80 columns.

--polling=N Set the polling interval to N (in seconds) when the dataset will be checked for appended data. The default polling interval is 1.

--field=<list\_of\_fields>

Display data for the field(s) specified in <list\_of\_fields> for a compound data type. <list\_of\_fields> can be specified as follows:

* A comma-separated list of field names in a compound data type.
* A single field name in a compound data type. Can use this option multiple times to specify multiple field names.

Note the tool’s default separators:

* ‘,’ is the separator for fields
* ‘.’ is the separator for a nested field
* ‘\’ is the escape character to avoid character(s) in field names that conflict with the tool’s separators.

## Object

*[object]* is the dataset to be monitored and is specified with the following format:

*[<filename>/<path\_to\_dataset>/<dataset\_name>]* **where**

* *<filename>*

**The name of the HDF5 file. It may be preceded by path separated by slashes to the specified HDF5 file.**

* *<path\_to\_dataset>*

**The path separated by slashes to the specified dataset**

* *<dataset\_name>*

**The name of the dataset to be monitored**

# Examples for h5watch

The dataset described in the following examples is a chunked dataset with unlimited dimension(s) or fixed dimensions with maximum dimension settings.

## Case A: monitor a one-dimensional dataset

*dsetA* is a one-dimensional dataset with 3 data elements—

* When the user increases the dimension size of *dsetA* from 3 to 5 and writes to *dsetA*:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

h5watch example.h5/dsetA will output the following:

dims[0]: 3->5

Data:

(3): *data*

(4): *data*

* When the user decreases the dimension size of *dsetA* from 3 to 2 and writes to *dsetA*:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
|  |  |

h5watch example.h5/dsetAwill output the following:

dims[0]: 3->2

## Case B:monitor a two-dimensional dataset

*dsetB* is a two-dimensional dataset with 3x4 data elements—

* When the user changes the dimension size of *dsetB*—increases dims[0]from3 to 6 and writes to *dsetB*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

h5watch example.h5/dsetB will output the following:

dims[0]: 3->6

dims[1]: unchanged

Data:

(3, 0): *data*, *data*, *data*, *data*

(4, 0): *data*, *data*, *data*, *data*

(5, 0): *data*, *data*, *data*, *data*

* When the user changes the dimension size of *dsetB*—increasesdims[1]from4 to 5 and writes to *dsetB*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

h5watch example.h5/dsetB will output the following:

dims[0]: unchanged

dims[1]: 4->5

Data:

(0, 4): *data*

(1, 4): *data*

(2, 4): *data*

* When the user changes the dimension size of *dsetB*—increases dims[0]from3to *5*, increases dims[1] from *4* to5, and then writes to *dsetB*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

h5watch example.h5/dsetB will output the following:

dims[0]: 3->5

dims[1]: 4->5

Data:

(0, 4): *data*

(1, 4): *data*

(2, 4): *data*

(3, 0): *data*, *data*, *data*, *data*, *data*

(4, 0): *data*, *data*, *data*, *data*, *data*

(5, 0): *data*, *data*, *data*, *data*, *data*

* When the user changes the dimension size of *dsetB*—increases dims[0] from 3 to 5,decreasesdims[1] from 4 to 3, and then writes to *dsetB*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

h5watch example.h5/dsetB will output the following:

dims[0]: 3->5

dims[1]: 4->3

Data:

(3, 0): *data*, *data*, *data*

(4, 0): *data*, *data*, *data*

* When the user changes the dimension size of *dsetB*—decreases dims[0] from 3 to 2, increases dims[1] from 4 to 5, and then writes to *dsetB*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

h5watch example.h5/dsetB will output the following:

dims[0]: 3->2

dims[1]: 4->5

Data:

(0, 4): *data*

(1, 4): *data*

* When the user changes the dimension size of *dsetB*—decreases dims[0] from 3 to 2, decreases dims[1] from4 to 2,and then writes to *dsetB*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |

h5watch example.h5/dsetB will output the following:

dims[0]: 3->2

dims[1]: 4->2

## Case C:monito*r* adataset with compound data type

*dsetC1* is a two-dimensional dataset of 3x4 data elements with compound data type defined as:

DATATYPE "ctype1" H5T\_COMPOUND {

H5T\_STD\_I32BE "c1";

H5T\_STD\_I32BE "c2";

H5T\_STD\_I32BE "c3";

H5T\_STD\_I32BE "c4";

}

* When the user changes the dimension size of *dsetC1*—increases dims[0] from 3 to 5and writes to *dsetC1*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

h5watch example.h5/dsetC1 will output the following:

dims[0]: 3->5

dims[1]: unchanged

Data:

(3, 0): {{*data* for c1,c2,c3,c4}, {*data* for c1,c2,c3,c4},

{*data* for c1,c2,c3,c4}, {*data* for c1,c2,c3,c4}}

(4, 0): {{*data* for c1,c2,c3,c4}, {*data* for c1,c2,c3,c4},

{*data* for c1,c2,c3,c4}, {*data* for c1,c2,c3,c4}}

* When the user changes the dimension size of *dsetC1*—increases dims[1] from 4 to 5 and writes to *dsetC1*:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

h5watch –-fields=c2,c4 example.h5/dsetC1

or

h5watch –-fields=c2 –-fields=c4 example.h5/dsetC

will output the following:

dims[0]: unchanged

dims[1]: 4->5

Data:

(0, 4): {{*data* for c2, c4}}

(1, 4): {{*data* for c2, c4}}

(2, 4): {{*data* for c2, c4}}

In the output above, *h5watch* displays data only for the two selected fields: the secondfield and the fourth field of the dataset’s compound data type.

*dsetC2*  is a one-dimensional dataset of 3 data elements with nested compound data type defined as below. The tool’s default separators “,’ and “.’ are used as part of the field names.

DATATYPE "ctype2" H5T\_COMPOUND {

H5T\_STD\_I32BE "c,1";

H5T\_STD\_I32BE "c,2";

H5\_COMPOUND {

H5T\_STD\_I32BE "sub.1";

H5T\_STD\_I32BE "sub.2";

} c3;

}

* When the user changes the dimension size of *dsetC2*  from 3 to 5 and writes to *dsetC2*:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

h5watch –-fields=c\,1 –-fields=c3.sub\.2 example.h5/dsetC2 will output the following:

dims[0]: 3->5

Data:

(3): {{*data* for “c,1”, “sub.2 of c3”}}

(4): {{*data* for “c,1”, “sub.2 of c3”}}

In the output above, *h5watch* displays data only for the two selected fields: the firstfield and the second subfield of the third field. Backslash is used to escape “.” and “,” in the field names.

## Case D: monitor changes in size of dataset dimensions via -d option

*dsetD1* is a one-dimensional dataset with 3 data elements—

* When the user increases the dimension size of *dsetD1* from 3 to 5:

h5watch –-dim example.h5/dsetD1 will output the following—

dims[0]: 3🡪5

* When the user decreases the dimension size of *dsetA* from 3 to 2:

h5watch –-dim example.h5/dsetD1 will output the following—

dims[0]: 3🡪2

*dsetD2* is a two-dimensional dataset with 3 x 4 data elements—

* When the user changes the dimension size of *dsetD2*—increases dims[0]from3 to 6:

h5watch –-dim example.h5/dsetD2 will output the following—

dims[0]: 3🡪6

dims[1]: unchanged

* When the user changes the dimension size of *dsetD2*—increases dims[0] from 3 to 5*,* andincreasesdims[1] from 4 to 5:

h5watch –-dim example.h5/dsetD2 will output the following—

dims[0]: 3🡪5

dims[1]: 4🡪5

# New public routines

This RFC adds three new public routines to the HDF5 High Level Interface. These routines perform operations for a dataset object based on a specified selection.

## size\_t H5LDget\_dset\_type\_size(hid\_t did, const char \* fields)

This routine retrieves the size in bytes of the data type for the dataset *did*.

If the parameter *fields* is NULL, this routine just returns the size of the dataset’s data type*.* Otherwise, this routine returns the size of the data type(s) for the selected fields specified in *fields* of a compound data type. Note that “,” is the separator for fields while “.” is the separator for a nested field. Users can use backslash to escape characters in field names that conflict with the tool’s default separators.

The parameters for this routine are**:**

*hid\_t did*:

IN: The dataset identifier.

*char \* fields*:

IN: A string containing a comma-separated list of fields for a compound datatype.

## herr\_t H5LDget\_dset\_dims(hid\_t did, hsize\_t \*cur\_dims)

This routine retrieves the current dimension sizes for the dataset *did* through the parameter *cur*\_*dims*. It will return failure if *cur\_dims* is NULL.

The parameters for this routine are:

*hid\_t did*:

IN: The dataset identifier.

*hsize\_t \*cur\_dims*:

OUT: The current dimension size of the dataset.

## herr\_t H5LDget\_dset\_elmts(hid\_t did, const hsize\_t \*prev\_dims, const hsize\_t \*cur\_dims, const char \* fields, void \*buf)

This routine retrieves selected data from the dataset *did* andstores the data in the parameter *buf.*  The difference between the parameters *prev\_dims* and *cur\_dims* indicates the dimension size of the data to be selected from the dataset. Note that *cur\_dims* must have at least one dimension whose size is greater than the corresponding dimension in *prev\_dims*.

The parameter *fields* is a string containing a comma-separated list of fields. If *fields* is NULL, this routine returns data for the whole dataset element. Otherwise, each name in *fields* selects a field in a compound data type and returns data for the selected fields. Note that ‘,’ is the separator for fields while ‘.’ is the separator for a nested field. Users can use backslash to escape characters in field names that conflict with the tool’s default separators.

Users can determine the size of *buf* by multiplying the size of the dataset’s data type by the number of selected data elements.

The parameters for this routine are:

*hid\_t did*:

IN: The dataset identifier.

*hsize\_t \*prev\_dims*:

IN: The previous dimension size of the dataset.

*hsize\_t \*cur\_dims:*

IN: The current dimension size of the dataset.

*char \*fields:*

IN: A string containing a comma-separated list of fields for a compound data type.

*void \*buf:*

OUT: Buffer containing the selected data elements in the dataset.

# Examples for the three new public routines

## Example 1

The dataset *DSET1* is a two-dimensional chunked dataset with atomic type defined as follows:

DATASET "DSET1" {

DATATYPE H5T\_STD\_I32LE

DATASPACE SIMPLE { ( 4, 13 ) / ( 60, 100 ) }

:

:

}

The following coding sample illustrates the reading of data elements appended to the dataset *DSET1*:

/\* open the HDF5 file \*/

fid = H5Fopen(FILE, H5F\_ACC\_RDWR, H5P\_DEFAULT);

/\* open the dataset \*/

did = H5Dopen2(fid, "DSET1", H5P\_DEFAULT);

:

:

/\* define hsize\_t dims[2]; \*/

/\* define hsize\_t new\_dims[2]; \*/

/\* get the dataset's current dimension sizes \*/

H5LDget\_dset\_dims(did, dims);

/\* extend the dataset by 2 \*/

new\_dims[0] = dims[0] + 2;

new\_dims[1] = dims[1] + 2;

H5Dset\_extent(did, new\_dims)

/\* write data to the extended dataset \*/

:

:

/\* get the size of the dataset's data type \*/

type\_size = H5LDget\_dset\_type\_size(did, NULL);

:

:

/\* allocate buffer for storing selected data elements from the dataset \*/

/\* calculate # of selected elements from dims & new\_dims \*/

/\* buffer size = type\_size \* number of selected elements \*/

:

:

/\* read the selected elements from the dataset into buf \*/

H5LDget\_dset\_elmts(did, dims, new\_dims, NULL, buf);

:

:

H5Dclose(did);

H5Fclose(fid);

The buffer *buf* will contain data elements selected from *DSET1* as follows:

data for elements (0, 13), (0, 14)

data for elements (1, 13), (1, 14)

data for elements (2, 13), (2, 14)

data for elements (3, 13), (3, 14)

data for elements (4, 0), (4, 1), (4, 2)…………………………………………………………(4, 13), (4, 14)

data for elements (5, 0), (5, 1), (5, 2)…………………………………………………………(5, 13), (5, 14)

## Example 2

The dataset *DSET2* is a one-dimensional chunked dataset with compound type defined as follows:

DATASET "DSET2" {

DATATYPE H5T\_COMPOUND {

H5T\_STD\_I32LE "a";

H5T\_STD\_I32LE "b";

H5T\_ARRAY { [4] H5T\_STD\_I32LE } "c";

H5T\_STD\_I32LE "d";

H5T\_STD\_I32LE "e";

H5T\_COMPOUND {

H5T\_STD\_I32LE "a";

H5T\_STD\_I32LE "b";

H5T\_ARRAY {[4] H5T\_STD\_I32LE} "c";

H5T\_STD\_I32LE "d";

H5T\_STD\_I32LE "e";

} "s2";

}

DATASPACE SIMPLE { ( 5 ) / ( 5 ) }

:

:

}

The following coding sample illustrates the reading of data elements appended to the dataset *DSET2* with compound data type—select only 2 fields: the fourth field “*d*” and a subfield of the sixth field “*s2.c*” from the compound type:

/\* open the HDF5 file \*/

fid = H5Fopen(FILE, H5F\_ACC\_RDWR, H5P\_DEFAULT);

/\* open the dataset \*/

did = H5Dopen2(fid, "DSET2", H5P\_DEFAULT);

/\* define hsize\_t dims[1]; \*/

/\* define hsize\_t new\_dims[1]; \*/

:

:

/\* get the dataset's current dimension size \*/

H5LDget\_dset\_dims(did, dims);

/\* extend the dataset by 2 \*/

new\_dims[0] = dims[0] + 2;

H5Dset\_extent(did, new\_dims);

:

:

/\* write data to the extended part of the dataset \*/

:

:

/\* #define fields “d,s2.c” \*/

/\* get the size of the dataset's data type for the selected fields \*/

type\_size = H5LDget\_dset\_type\_size(did, fields);

:

:

/\* allocate buffer for storing selected data elements from the dataset \*/

/\* calculate # of selected elements from dims & new\_dims \*/

/\* buffer size = type\_size \* number of selected elements \*/

:

:

/\* read the selected elements from the dataset into buf \*/

H5LD\_get\_dset\_elmts(did, dims, new\_dims, fields, buf);

:

:

H5Dclose(did);

H5Fclose(fid);

The buffer *buf* will contain data for “*d*” and “s2.c” selected from *DSET2* as follows:

Data for element (5): integer value for “d”, 4 integer values for array “s2.c”

Data for element (6): integer value for “d”, 4 integer values for array “s2.c”

# Reference Manual

Reference manual entries for *h5watch* and the three new public routines will be added to the HDF5 High-Level Interface.

# Acknowledgements

This work is supported by a commercial client of the HDF group.

# Revision History

|  |  |
| --- | --- |
| *November 30, 2009:*  *February 24, 2014:* | Version 1 circulated for comment within The HDF Group.  Version 2 updated to reflect current implementation. |