RFC: Support Reg. Ref. and Bitfield in HDFView

Peter Cao

The purpose of this RFC is to solicit ideas and request inputs on how to display data pointed region references and how to show bitfield data in HDFView. Related RFCs on higher level APIs handling region references and bitfield data include:

- RFC: High-Level Functions for Handling Region References and Hyperslab Selections by M. Scot Breitenfeld and Elena Pourmal
- RFC: Reading Quality Flags from NPOESS Product File by Elena Pourmal and M. Scot Breitenfeld

1 Introduction

NPOESS data users need convenient tools to view and export packed quality flag values, as well as data point to by region references. HDFView, a graphical tool for displaying and editing the contents of HDF5 files, is one of the HDF5 tools, which are particularly important to NPOESS in managing and using this kind of data. However, the current HDFView does not support bitfield and doesn't show data pointed by region references.

Hence, the following tasks are proposed:

i). Modify HDFView to display in spreadsheet format a rectangular region of data in a dataset corresponding to the set of region references.

ii). Modify HDFView to display packed bits corresponding to the NPOESS quality flags.

2 Region References

The NPOESS Program delivers the official deliverable data products (RDR, SDR/TDR, EDR/ARP/IP) and dynamic ancillary data and auxiliary data in HDF5 Files. Official deliverable data products are organized by reference objects (aggregations), which contain one or more reference regions (granules). A region reference points to a dataset and a region within that dataset.

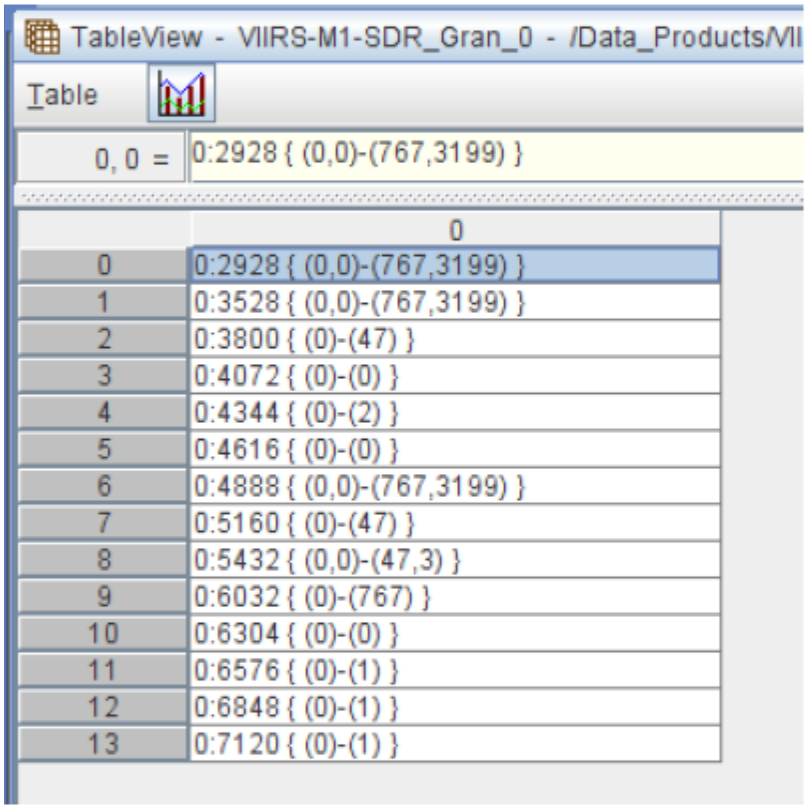
2.1 Use Cases

- A user starts HDFView and using a special icon sets properties to show datasets with region references. He opens the HDF5 file and inspects the datasets. The user finds the dataset of interest and clicks the right button to display a menu with data display options. He chooses to display a region reference data in a spreadsheet with each element shown as a path to a dataset and corner coordinates of a hyperslab. The user opens the dataset with region references, examines the data and clicks on one of the elements. HDFView spawns a spreadsheet window with the referenced data displayed.

- A user starts HDFView and sets properties to show datasets with region references using a special icon. He opens the HDF5 file and inspects the datasets. The User finds the dataset of interest and clicks the right button to display a menu with data display options. He chooses to display the region reference data graphically. The User opens the dataset with region references. HDFView spawns another window with the file tree showing all referenced datasets (i.e., datasets referenced by the elements of the dataset with region references). The User clicks on one of the datasets. HDFView displays data from the referenced region in a spreadsheet.
- A User opens a dataset as an image in HDFView. He selects a sub-region by clicking and dragging the mouse. Then he uses one of the menus to create a region reference and store it in a buffer. The User opens another dataset, selects a region, creates the second reference, etc. When all desired region references are created the user saves them in a dataset in the file. Now he can use the dataset with region references to find the datasets and regions of interest.

2.2 Current Behavior

HDFView shows the values of a dataset with region references as obj_ref {coordinates_of_reg, ...}. For example, 2928{ (0,0)-(767, 3199)} is a 2D rectangle section of (0,0)-(767, 3199) in the dataset with a unique object identifier of 2928. HDFView currently does not display the data pointed by the region selection. We want to add the capability to do so. The following snapshot is an example of showing a dataset of region references in the current HDFView



0, 0 = 0:2928 { (0,0)-(767,3199) }

	0
0	0:2928 { (0,0)-(767,3199) }
1	0:3528 { (0,0)-(767,3199) }
2	0:3800 { (0)-(47) }
3	0:4072 { (0)-(0) }
4	0:4344 { (0)-(2) }
5	0:4616 { (0)-(0) }
6	0:4888 { (0,0)-(767,3199) }
7	0:5160 { (0)-(47) }
8	0:5432 { (0,0)-(47,3) }
9	0:6032 { (0)-(767) }
10	0:6304 { (0)-(0) }
11	0:6576 { (0)-(1) }
12	0:6848 { (0)-(1) }
13	0:7120 { (0)-(1) }

Figure 1 -- Region references displayed in HDFView

2.3 Recommendation

Below are some tentative ways of showing data pointed by a region reference in HDFView:

i). Display data in a popup window as you move the mouse over a cell.

a. Pros: easily navigate data pointed by region references.

b. Cons: performance can be a problem if the region is large or you move your mouse frequently.

ii). Show data in a separate spreadsheet by double click a cell. Also, you can use a right mouse click to select options to display data in image or spreadsheet.

iii). Allow users to highlight multiple cells of the region reference dataset and show the data pointed by the region references in separate windows.

iv). Create dataset of region references and update values of region references.

2.4 Other optional features

i). Displaying point data and data of multiple regions selected by a single region reference can be challenging. One solution is to open each selection in a separate sheet.

ii). Distinguishing between the data pointed by region reference and the data of regular dataset can be tricky. One way to is to use different background.

3 Bitfield Data

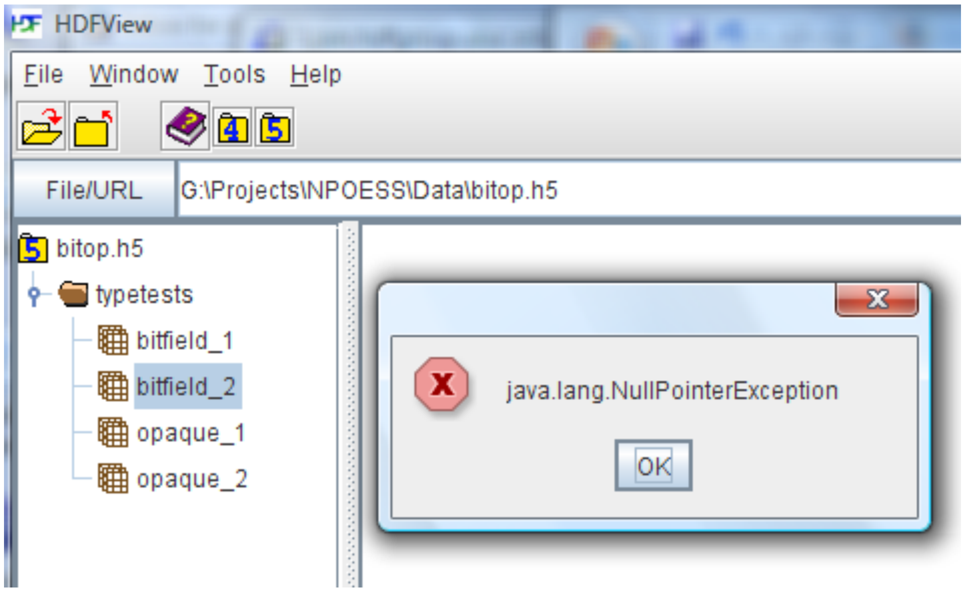
NPOESS quality flag data is to provide information about the quality of the delivered data on an element-by-element basis. To reduce storage size, several quality flags associated with a data product are packed in one byte and stored as an 8-bit unsigned integer in HDF5 file. Our goal is to show specific quality flag(s) or field(s) in HDFView.

3.1 Use Cases

- A user starts HDFView, opens an NPOESS EDR file and a dataset with quality flags. The browser spawns several windows and displays spreadsheets for each quality flag stored.
- A user opens an NPOESS EDR file in HDFView. He finds a dataset with the IST product data. In a menu the user chooses a tab to show quality flags and clicks on “Active Fire” quality flag. After that he opens the IST dataset and only elements for which the value of the “Active Fire” quality flag is 1 are shown. The rest are filled with a specified fill value.

3.2 Current Behavior

The current HDFView shows the byte data as whole. Users will not be able to see individual bit(or bits packed in the byte. If the datatype of the dataset is bitfield, HDFView only shows an error message when you try to open the dataset.



java.lang.NullPointerException

Figure 2 -- Error message on bitfield dataset

3.3 Recommendation

We are dealing two issues here:

a) Showing specific bit(s) value in integer

b) Supporting HDF5 bitfield data (datatype class is H5T_BITFIELD)

Since NPOESS quality flags are packed in one byte, we will focus on the first issue. Supporting general bitfield data can be optional. We include b) in this RFC just in case we may implement it in the future.

3.3.1 Store bitfield data in Java

For NPOESS data, since the bit fields of quality flags are packed in an unsigned byte, we can use the Java byte to store the data in memory.

Bitfield data is stored in HDF5 file as one of the four basic types: H5T_STD[NATIVE, …]_B8BE[LE], H5T_STD[NATIVE, …]_B16BE[LE], H5T_STD[NATIVE, …]_B32BE[LE], and H5T_STD[NATIVE, …]_B64BE[LE]. The four types of bitfield data can be stored as Java byte, short, int, and long respectively.

Number of bits	**Java Data Type**
H5T_STD[NATIVE, …]__B8BE H5T_STD[NATIVE, …]__B8LE	byte
H5T_STD[NATIVE, …]__B16BE H5T_STD[NATIVE, …]__B16LE	short
H5T_STD[NATIVE, …]__B32BE H5T_STD[NATIVE, …]__B32LE	int
H5T_STD[NATIVE, …]__B64BE H5T_STD[NATIVE, …]__B64LE	long

3.3.2 Retrieve specific bit or bits

When the data is in memory, a simple bitmask operation can easily get the values of a specific bit or bits if the data is stored in primitive data types. The structure of bit allocation of quality flags (bit mask) is defined in NPOESS data profile or in attribute.

3.3.3 Show bit data in HDFView

Below are some tentative ways that we can show bitfield data.

- Showing the whole bit data as a number. We can display the number in one of the following ways (use a 10-bit data, “1111111111", as an example):
    - Decimal, e.g. “1023”
    - Hexadecimal, e.g. “3ff”
    - Binary, e.g. “1111111111"
    - Octal, e.g. “1777”
- Displaying each field in a separate spreadsheet. For example, if a byte contains three fields, the byte will be displayed in three separate spreadsheets. Each spreadsheet shows that one filed extracted from the byte based on the field structure.

- Adding a selection option for bitfield in “Open As” window to allow users to choose which specific field to be displayed.

Acknowledgements

This work is supported by the NPOESS project.

Revision History

April 15, 2009: Peter added use cases from Scot’s and Elena’s RFCs.

April 24, 2009: Peter updated based on Elena’s comments.