RFC

Addressing HDF5 file corruption issue

Quincey Koziol, Elena Pourmal
November 11, 2007

This document describes data corruption problem in early releases of HDF5 and how it affected some AURA data files produced by the MLS science team (JPL), and proposes the way of fixing the problem on the level of the HDF5 library.

Background

Versions of the HDF5 library prior to the HDF5 1.6.6 release had a bug that could cause corruption of object header messages.

Generally, the sequence of actions to generate this bug looks like this:

- Create an object.
- Close the file.
- Re-open the file.
- Add 2 (or more) attributes to the object.
- Close the file.
- Re-open the file.
- Delete one of the attributes on the object.
- Add a smaller attribute to the object.
- Delete the smaller attribute on the object.
- Add a larger attribute on the object.

After this, the number of header messages stored for the object will be off by one. Other sequences of modifying attributes on an object could also trigger this bug. If application opens an object and the bottom few messages of the HDF5 error stack resembles the following, the object has been affected by this bug:

#007: ../../hdf5_v1.6/src/H5C.c line 3887 in H5C_load_entry():
  unable to load entry
  major(08): Meta data cache layer
  minor(40): Unable to load metadata into cache
#008: ../../hdf5_v1.6/src/H5Ocache.c line 332 in H5O_load():
  corrupt object header - too few messages
  major(12): Object header layer
  minor(40): Unable to load metadata into cache

Specifically, "*corrupt object header*" is the best string to search for in the HDF5 error stack output.

The bug was fixed in HDF5-1.6.6-snap0 and in HDF5-1.8.0-alpha4
(For more details see the HDF Newsletter # 88
http://www.hdfgroup.org/newsletters/newsletter88.html).

As a result of the fix, current library doesn't allow creation of corrupted files but it also fails on the read operations on the corrupted files producing an error message similar to the message shown above.

One year ago The HDF Group provided a tool to detect and fix corrupted files ftp://hdfgroup.org/HDF5/special_tools/. THG also issued a newsletter #88 to notify HDF5 users.

Please note that the tool has to be compiled with the HDF5 library version 1.6.5 or earlier. The h5check tool (ftp://ftp.hdfgroup.uiuc.edu/pub/outgoing/h5check/) that assures the consistency of the HDF5 files, finds the problem but doesn't fix it.

Recently it was discovered that there were corrupted files that had been made already available to the users of the AURA MLS data in spite of the rare event of the HDF5 library sequence of calls that triggered the bug.

Examples of the corrupted files include (please see http://disc.gsfc.nasa.gov/data/datapool/MLS/; files are also available on the THG machines under /mnt/scr1/NASA/corrupted)

MLS-Aura_L2AUX-Cloud_v01-51-c01_2004d245.h5
MLS-Aura_L2AUX-Cloud_v01-51-c02_2005d182.h5
MLS-Aura_L2AUX-Cloud_v01-52-c02_2006d139.h5

MLS-Aura_L2AUX-Cloud_v01-52-c01_2007d059.h5

along with some internal files generated by the MLS team. THG notified AURA science teams, DAACs, HDF-EOS developer and management, and vendors about the corrupted data. Since that time THG has been working on the solution to the problem.

Interaction with NASA

The HDF group discussed with the NASA AURA science teams and DAACs the possible approaches to the problem:

1. Ask AURA data users to run the tool provided by THG
2. Fix data at DAACs level
3. Switch applications to use HDF5 1.6.6 and later versions to stop production of the corrupted files along with implementing steps 1 - 2.
4. Fix the HDF5 library to be able to read corrupted files in spite of recognition that corruption can be triggered by some other event.

After consideration of all options listed above, THG received the following response from AURA MLS team (JPL):

Paul Wagner (MLS, JPL) in email to THG (Elena Pourmal), AURA team members and DAACs representatives sent on November 2, 2007:

Our views are based on the overriding need to assure our users uninterrupted access to all our scientific data.

Therefore we favor that the next and any future public releases of the hdf5 library be indifferent to the kinds of corrupt headers created by earlier versions of the library. This indifference should not require any special instructions or optional arguments by the users.

What we hope is that users of our data, the SDP Toolkit including HDF-EOS5, and tools such as idl and matlab, when built against the latest public version of hdf5 would automatically obtain a built-in "immunity" to the corrupt header bug.

Even if future hdf5 library releases add stricter filters to catch evidence of file corruption, the need for backward compatibility should allow them to read successfully these files written with the looser standards of their time.

We do not plan on using the special repair tool on files identified as having the corrupt headers. As a policy we don't approve of "correcting" existing data, and if files in some archives are repaired while those in other archives are left as is they will no longer be identical.

We do not plan to immediately switch to using the current v1.6.6 release of the hdf5 library in our processing. Older v1.51 and v1.52 (our own version numbers, not hdf5 numbers) of our software were the ones that created files with the corrupt headers. Our current processing is with v2.21 and the files it creates do not appear to have this problem.

Next section of this document describes changes to the HDF5 library necessary to address JPL's request. After implementing the changes, THG will need to come up with an official release to be included in the releases of the HDF-EOS Toolkit, and the next versions of Matlab and IDL.

Proposed solution

In keeping with the desire from NASA to have the default installation read these files without additional flags, we should arrange for that to happen. This can have two behaviors, depending on whether the file is open read-only or read-write:

- Files opened for read-only access should accept files with this sort of corruption, making the necessary adjustments to determine the correct number of object header messages for the object.
- Files opened for read-write access should accept files with this sort of corruption, the same as files opened for read-only access. Additionally, they should re-write the object header for a corrupted object, in order to write out the correct information.

In addition to these two changes, we should add an additional configure flag which enables strict format checking and returns an error when a file with this sort of corruption is detected (the current behavior).