HDF5 text formats: current capabilities

Robert E. McGrath
27 February, 2005

While the HDF5 a binary format, we do have some text-based technology. This note is a summary of what has been done to date. These text formats may or may not turn out to be relevant to what we want to do for long term archiving.

1. Format specification

I don't want to overlook the obvious: the HDF5 storage format is documented in a formal document [1]. The format document could be included as a preamble in a binary file, for example.

In principle, the format specification can be used to define a text encoding at the level of the stored file structures. I am not aware of any attempt to do so to date.

2. Text representations of the logical objects in the file

The *h5dump* utility provides a rather complete capability to transform binary HDF5 to a formally defined text format. [2] This utility analyses a single HDF5 file and produces a description of the objects in the file, including the names and contents of Groups, Datasets, Attributes, along with key information about the binary file, such as compression used and chunking.

This text representation is based on the HDF5 Abstract Data Model ([3]), and has a formal grammar, the HDF5 Data Definition Language (DDL) [4].

The *h5dump* utility has an option to dump the same information in XML. The format of the XML output is defined in an XML schema [5].

In principle, either form of text output from the *h5dump* utility can be parsed to create a binary file that is *logically equivalent* to the original. "Logically equivalent" means that all the objects have the same names, attributes, and data values (within the precision of the computing platform). The *h5diff* utility tests this logical equivalence, although there is no formal specification of the equivalence [6].

The transformation from text to HDF5 binary has been done with the XML output of h5dump , In several experiments, we were able to successfully dump to text and then re-generate a binary that was logically identical to the original [7]. The XML file was typically ten times the size of the binary, although common data compression techniques were observed to reduce the XML files to almost the same size as the binary HDF5. The text-to-binary tool is no longer available.

The DDL and/or XML text representation might be of interest for long term archiving, depending on the goals of the archive. There are several points about these formats I would like to note:

1. The text is a representation of the logical contents of the files, not the storage layout of the file. A regenerated file would not be a bit-for-bit restoration of the original.
2. Following from this fact, the dump/gen is done external to the HDF5 library. It is far from clear how or if it would be possible to make the HDF5 library read/write these text formats, if that was a requirement.

In addition to the fundamental limitations, it should be noted that both the DDL and XML schema specifications are incomplete. Key missing features are:
1. there is no markup for the format for atomic data elements, including numbers, and strings
2. the markup for arrays is not fully specified,
3. the markup for names and addressing specific elements is incomplete

These deficiencies are not a problem for the current use of the tool, but become critical for a 'gen' tool.

There is no reason these specifications cannot be created. However, it is quite possible that the existing implementation would have to be significantly modified to implement these formal markups. (The current implementation uses "printf" to format numbers, and arranges the array elements to fit on a printed page or screen.)

3. HDF4

Given the enormous amount of data already stored in HDF4, it is likely that archives will need a text format for HDF4 files, as well. One option would be to convert the file to HDF5, and then to text. This double transformation might be problematic for long term preservation: it would be more challenging to state how precisely the original data was preserved.

The older HDF4 binary file format has its own text representation, HDF Configuration Record (HCR) [8]. Like the HDF5 DDL, the HCR defines a formal grammar for describing the objects in an HDF4 or HDF-EOS2 file. The HCR grammar is defined in ODL.

The HCR has similar limitations to the HDF5 DDL, it is incomplete in similar ways. Unlike the HDF5 DDL, the HCR does not define a text representation for the values of arrays, i.e., the "problem sized data" in the file.[1]

At one time, there was a set of tools that wrote and read this format. This software has not been tested in many years, most likely it doesn't work without some effort to update it.

To my knowledge, there was no HCR to binary conversion.

An XML schema based on the ODL was created. No tools are available to write or read the XML, so the schema has not yet been published.

4. Discussion

[1] This statement is based on my incomplete understanding of HCR. I stand to be corrected on this point.

We have developed some text representation for HDF5, including a fairly complete XML schema. We have also demonstrated the feasibility of transforming from binary to text, and from text to binary. At this time, we have a robust ability to transform from binary to text, but not back again. We know it is feasible to develop these tools, if this would meet the requirements of long term archives.

The existing text formats (excluding the format specification) are representations of the logical objects in the file. The text representations have very little relationship to the stored binary file. This may or may not be the representation wanted for long term archiving.

The experiments with XML raised some fundamental issues that we will need to deal with. I will briefly note some of them here.

First, as already noted, there was a question as to what should be represented. In the DDL and XML, we not only focused on the logical objects, we specifically attempted to describe the *properties of the stored object*. However, an HDF object can be viewed as having "behaviors" as well, i.e., data transfer and transformation actions. For example, we report the stored data type of the numbers, but print the values in a native memory format (i.e., we present the result of the "read this data here" operation, not fact of "what is on disk"). This is appropriate for human readers, but might not be the correct design for a formal archive format.

Second, we have not seriously attempt to address the issue of what would be a "faithful rendition" of a given object or file. Given that we represent the logical object, it is not trivial to ground this definition of faithfulness in the format specification itself. If we want to use this kind of logical representation for a formal archive format, I think we will need to define the correctness and conformance standards. For instance, we will need to define a text format for the data elements, and specify the correct mapping between a stored data type and the text string to represent it.

Third, I would observe that one advantage of representing the logical (rather than the physical) file is that the logical view is much easier to map to other formats. We are confident that formats such as HDF4, GIF, FITS, or CDF can be mapped to HDF5 objects (because we have done it), and therefore could be stored in, say HDF5 XML. In my opinion, it is less clear how easy it would be to map from another format to the HDF5 storage specification.

References

1. http://hdf.ncsa.uiuc.edu/HDF5/doc/H5.format.html
2. http:.. h5dump
3. http://hdf.ncsa.uiuc.edu/HDF5/papers/presentations/ADM/ADM_EOS_Sep99/EOSpresentation/index.html
4. http://hdf.ncsa.uiuc.edu/HDF5/doc/ddl.html
5. http://hdf.ncsa.uiuc.edu/HDF5/XML/dtd-info.html
6. h5diff
7. http://hdf.ncsa.uiuc.edu/HDF5/XML/JSPExperiments/index.html
8. http://hdf.ncsa.uiuc.edu/hcr.html