Ensuring Long-Term Access to Remotely Sensed Data With Layout Maps

Ruth E. Duerr, *Member, IEEE*, Peter Cao, Jonathan Crider, Mike Folk, Christopher Lynnes, *Member, IEEE*, and Mu Qun Yang

***Abstract*—The Hierarchical Data Format (HDF) has been a data format standard in National Aeronautic and Space Administration (NASA)'s Earth Observing System Data and Information System since the 1990s. Its rich structure, platform independence, full-featured application programming interface (API), and internal compression make it very useful for archiving science data and utilizing them with a rich set of software tools. However, a key drawback for long-term archiving is the complex internal byte layout of HDF files, requiring one to use the API to access HDF data. This makes the long-term readability of HDF data for a given version dependent on long-term allocation of resources to support that version. Much of the data from NASA's Earth Observing System have been archived in HDF Version 4 (HDF4) format. To address the long-term archival issues for these data, a collaborative study between The HDF Group and NASA's Earth Science Data Centers (ESDCs) is underway. One of the first activities was an assessment of the range of HDF4-formatted data held by NASA to determine the capabilities inherent in the HDF format that were used in practice and for use in estimating the effort for full implementation across NASA's ESDCs. Based on the results of this assessment, methods for producing a map of the layout of the HDF4 files held by NASA were prototyped using a markup-language-based HDF tool. The resulting maps allow a separate program to read the file without recourse to the HDF API. To verify this, two independent tools based solely on the map files were developed and tested.**

***Index Terms*—Archiving, data conversion, data management, data structures, preservation.**

I. Introduction

THE VAST majority of the data created today are recorded in digital form, but digital data are, by their very nature, impermanent. One cannot assume that, simply because data are in a digital format, they will be accessible and understandable into the indefinite future. Nowadays, there are countless news reports about digital data that either have been lost or are no longer recoverable. The National Aeronautic and Space Administration (NASA)'s loss of data from the 1976 Viking mission to Mars [1] and the near loss of the BBC's 1986 digital Domesday project [2] are two of the most frequently cited incidents. An entertaining yet sobering description of the variety of ways that data can be lost has recently been published [3]. One of the mechanisms discussed, the mechanism ultimately at the core of both the Viking and Domesday problems, is loss of the ability to read the data archived. As data age, this risk to readability arises from two sources: the ability to read the bit stream from the storage medium and the ability to interpret the bit stream. From an archivist's standpoint, the former problem is tractable (although expensive and time-consuming) through media migration and backup strategies. On the other hand, preserving the interpretability of the data is more problematic, because it is not completely in the archivist's control. Consider, for example, a data file written in a format that can be read only with specific read software. Should that software no longer be supported, there arises a significant risk that, at some point, it will no longer be possible to deploy systems to read the data.

Hierarchical Data Format Version 4 (HDF4) [4], [5] is an example of such a software-dependent format. Strictly speaking, HDF4 does have a published format specification [6]; therefore in theory, one could read an HDF4 file by referring to the specification. However, the specification is sufficiently rich and complex that such an effort would be almost tantamount to reconstructing the HDF4 application programming interface (API) and library, a task that consumed an estimated 20 person-years to develop. This complexity arises from the flexible hierarchical structure as well as the use of chunking and several algorithms for internal compression that result in data being organized in a variety of ways. These features have the effect of complicating the bit layout on disk. Thus, from a practical standpoint, the archivist must consider HDF4 to be a software-dependent format.

The Earth Observing System Data and Information System (EOSDIS) of NASA stores more than 1 PB of remote sensing data in HDF4 at eight archive centers in the United States. The HDF Group (THG) (http://www.hdfgroup.org/) currently supports the HDF4 format [7]. THG also supports HDF Version 5, a format with similar properties but which is not backward compatible with HDF4 [8]. That is, the HDF5 software cannot read HDF4 files. The existence of the more advanced HDF5 format exerts a subtle, but palpable, pressure on the HDF community to migrate data to HDF5 so that HDF4 can be deprecated and eventually dropped. Such a migration would be expensive, making it likely that some of the less popular data sets would not be migrated, resulting in data sets that eventually will be unreadable.

Manuscript received March 12, 2008; revised June 13, 2008. First published November 17, 2008; current version published December 17, 2008. This work was supported in part by the National Aeronautics and Space Administration under DAAC Contract NAS5-03099.

R. E. Duerr is with the National Snow and Ice Data Center, University of Colorado, Boulder, CO 80309 USA (e-mail: rduerr@ nsidc.org).

P. Cao, M. Folk, and M. Q. Yang are with The HDF Group, Champaign, IL 61820 USA (e-mail: xcao@hdfgroup.org; mfolk@hdfgroup.org; ymuqun@hdfgroup.org).

J. Crider is with the University of Colorado, Boulder, CO 80309 USA (e-mail: jonathan.crider@nsidc.org).

C. Lynnes is with the National Aeronautics and Space Administration, Greenbelt, MD 20771 USA (e-mail:Chris.Lynnes@nasa.gov).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2008.2004626

In order to mitigate this risk, we have begun a pilot study to develop a means of reading HDF4 data without recourse to the HDF4 API. The basic concept is to develop an HDF4 utility to construct a human- and machine-readable "map" of the data layout on disk. This map would be archived along with the data as extra metadata. It would allow the creation of read programs that use the map but not the HDF4 API.

II. Project Description

This project comprises four main activities. First, NASA's holdings of HDF4 products have been assessed and categorized to determine which HDF4 capabilities are actually in use at NASA's Earth Science Data Center (ESDC) archives, as well as to help scope the effort for full implementation. Second, maps of the contents of HDF files are being prototyped, including specification of the syntax for such maps. Third, in order to test the utility of the maps developed, software is being developed which uses the maps generated to read actual HDF files. Last, the usefulness of this effort will be evaluated, and the scope of the effort for full implementation across all NASA ESDCs will be determined. These tasks are described in more detail in the sections that follow.

A. Assessing and Categorizing NASA Holdings

An early concern was that it might not be possible to map all types of HDF4 data given the variety and complexity of the files that could be generated using the HDF library's complete suite of capabilities. Thus, determining the structures that are in actual use and their frequency of use was deemed key to prioritizing capabilities to prototype. Moreover, one of the goals of this project is to provide a high-level evaluation of the effort for full implementation across all of NASA ESDC archives. This requires understanding of the holdings of each data center. Consequently, a catalog of HDF4 products in the archives of each of NASA's ten ESDCs was compiled, and samples of each product were obtained for analysis.

A combination of interactive tools (HDFView [9] and its HDF-Earth Observing System (EOS) extension [10]) and command line tools (e.g., hdp [11]) were used to determine which of the data structures and associated capabilities had been used to create each data file examined. The complete set of information recorded for each data product is given in Table I.

TABLE I
Product Characteristics Examined

- Product Identification
  - Product Name
  - Data Level
  - Archive Location
  - Product Version
- Whether the product was a multi-file product
- For HDF-EOS products
  - HDF-EOS Version
  - For point data
    - Number of point data sets
    - Maximum number of levels within the point data
  - For swath data
    - Number of swaths
    - Maximum number of dimensions within a swath
    - Organized by time, space, both, or other
    - Whether dimension maps were used
  - For gridded data
    - Number of grids
    - Maximum number of dimensions within a grid
    - Number of projections used
    - Whether any grids were indexed
- HDF Version
- For raster data
  - Number of 8-bit rasters
  - Number of 24-bit rasters
  - Number of general rasters
  - Whether any rasters had attributes
  - Whether any of the rasters were compressed
  - Whether any of the rasters were chunked
  - Whether there were any palettes
- For SDS data
  - Number of SDSs
  - Maximum number of dimensions of an SDS
  - Did any SDS have attributes
  - Was any SDS annotated
  - Were dimension scales used
  - Was compression used and if so what kind
  - Was chunking used
- For Vdata
  - Number of Vdata structures
  - Did any Vdata have attributes
  - Did any Vdata fields have attributes
  - Was compression used and if so what kind
  - Was chunking used

B. Constructing Maps

1) About HDF4: The HDF4 format is a multifile, hierarchical, and physical file format for organizing data on disk. The data structures that are supported include raster images (standard 8-b, 24-b, or self-defined), color palettes, tables (Vdatas), multidimensional arrays [scientific data sets (SDSs)], and groups of any of these structures (Vgroups) [5]. In addition, most of these structures can have associated metadata called attributes.

Some HDF4 structures can be stored on disk in a variety of ways. For instance, the data in an SDS can be stored in a compressed format or can be stored as a collection of chunks or tiles. This feature is important for creating maps to HDF4 data, for it means that a map must not only say *where* data reside in the file but also *how they are organized.*

Taken together, the HDF data structures can represent nearly any kind of data. However, the HDF4 structures and library do not impose any particular scientific meaning on the data they represent. For instance, an HDF4 SDS may be used to store data associated with a particular geographic projection, but the HDF4 library has no awareness that the SDS is being used for that purpose. It is up to the application accessing the SDS to know that the SDS contains a geographic projection and, if so desired, to know the geolocation of points in the projection.

2) EOS Data and Metadata: To provide scientific meaning to HDF structures, NASA has developed conventions for storing data from the EOS program as well as an HDF-EOS software library that implements these conventions. An HDF-EOS file is an HDF file that follows those conventions. For example, an HDF-EOS 2 file is an HDF4 file that uses Version 2 of the HDF-EOS libraries. HDF-EOS defines three types of data: data that are stored on a regular grid based on a geographic

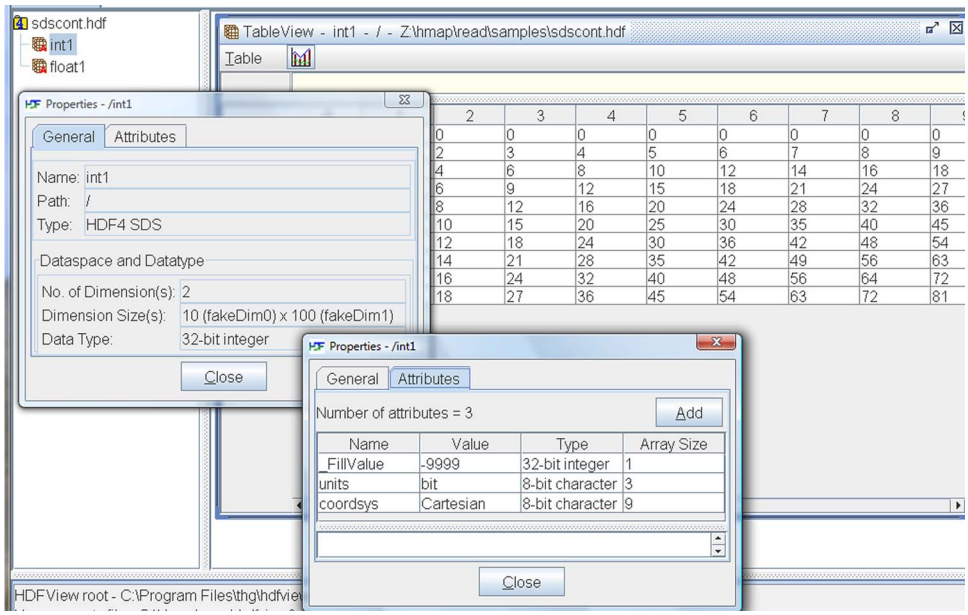

Fig. 1. Simple HDF4 file containing an SDS.

projection (grid data), data that are irregularly spaced in time and/or space (point data), and time-ordered data arrays (swath data) [12].

NASA's EOS data are also characterized by "processing level," a term used to describe how much processing has been done. Level 1A data are unprocessed instrument data at full resolution, with georeference information (swath coordinates). Level 1B data are Level 1A data processed to sensor units. Level 2 data are derived geophysical variables at the same resolution and location as the Level 1 source data. Level 3 data are mapped on uniform space–time grid scales, usually with some completeness and consistency. While the organization of data products varies, Level 1 and Level 2 data are typically stored in HDF-EOS swath structures, while Level 3 data are typically stored as grids.

Clearly, in developing preservation requirements, it is also important to preserve the annotations (metadata) that give EOS data its meaning. The following discussion describes mappings of pure HDF4 structures, but it is important to understand that these structures often represent HDF-EOS objects and are likely to be accessed in that context.

3) Mapping Requirements: Fig. 1 shows a simple HDF4 file with a single SDS. This example will be used to explain how the HDF4 mapping works. The SDS contains a 10×100 array whose elements are of type 32-b integer. The SDS is stored without chunking and is uncompressed, which means that the SDS is stored in the HDF4 file as a contiguous block of 1000 (10×100) 32-b integers (4000 B).

This example illustrates the fact that three types of information are needed to describe an object: structural metadata, application metadata, and the data of the object itself (object data). *Structural metadata* are information that describes the structure of the object as a data structure (a 10×100 array of integers), as well as how the data are organized on disk (contiguous, uncompressed). *Application metadata* are any interpretive information that is meaningful to an application, such as information shown in the attributes window. *Object data* are the data that make up the contents (elements) of the SDS array.

Fig. 2. Information needed to describe the location of the object data for the SDS example in Fig. 1.

Of these three types of information needed to describe an object, the mapping file must provide the structural and application metadata. Structural information is needed to retrieve the object data from the HDF4 file and application metadata to interpret the object data.

The structural information needed to retrieve object data includes the offset and size of each block of object data. With this information, other programs can use routines such as C's fseek() and fread() to extract the data from an HDF4 file without the HDF4 library. Structural metadata for an SDS include data type, number of dimensions, dimension sizes, and storage layout. Other objects (Vgroups, Vdata, raster images, color palettes, and attributes) have similar structural metadata.

Application metadata refer directly to the object data content and include information such as time and geolocation information. In this phase of the project, mapping files contain any application metadata that occur in the HDF4 file, but relationships between application metadata and object data are not described.

Fig. 2 shows this requirement for the simple example in Fig. 1. In this example, a C program could access the object data by seeking to the offset 2502, then reading the next 4000 B. In this case, the first 2502 B of the file contain file header information and storage space for metadata and header information about the data object.

Fig. 3 shows a more complex example of HDF4 file. There are three data objects in this example file: a Vdata, an SDS, and

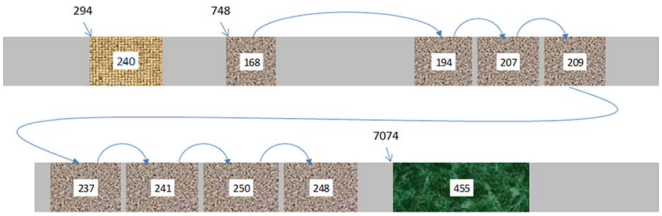

Fig. 3. More complex example HDF4 file structure.

a raster image. The object data of the Vdata table are located at offset 294 with size of 240 B. The SDS is a 10 × 100 2-D array with a chunk size of 5 × 25. The offset of the first chunk is 748. Because the chunks are compressed (gzip compression in the example), each chunk has different lengths in the file—168 B, 194 B, 207 B, etc. The last object is a 46 × 23 2-D 8-b raster image compressed with JPEG. The image is located at offset 7074 with size of 455 B. The gaps between object data blocks represent the storage space for metadata and header information.

C. Reading Maps

The true test of the efficacy of this mapping process is whether a non-HDF-expert programmer can develop code that uses the maps to read actual data files without using the HDF libraries. In order to demonstrate this, two tests are underway. Using maps provided by THG, both the National Snow and Ice Data Center (NSIDC) and the Goddard Earth Sciences Data and Information Services Center (GES-DISC) have independently built software utilities that use these maps to read data files from their archives. In the case of NSIDC, an undergraduate majoring in computer science, unfamiliar with science data in general and HDF data in particular, was tasked to develop the read utilities. The accuracy of the read utilities has been assessed by comparing the data read to the equivalent data extracted from the original data file using the HDF libraries.

III. Plans and Early Results

A. Product Assessment

One of the first activities undertaken was to collect and analyze samples of each of the HDF4 and HDF-EOS 2 products archived by any of NASA's ESDCs. This task was complicated by the factor that the ESDCs maintain separate archive catalogs—no definitive NASA-wide compilation exists, making it necessary to elicit the support of each center to complete the task. As a starting point, archive lists were compiled from the NASA Earth Science Data and Information System's Data Gathering and Reporting System (EDGRS) and EOS ClearingHOuse (ECHO) systems. The EDGRS and ECHO lists are not 100% accurate and represent only a snapshot in time of NASA's ESDC data holdings. Many of the data products archived by the ESDCs come from active missions, where reprocessing and/or replacement of a product stream is expected to occur every few years. Given these caveats, however, the catalog does provide a reasonably comprehensive look at NASA's currently available publicly accessible Earth science data holdings.

All of the centers have indicated either which products in their archive are in HDF4/HDF-EOS 2 format or that none of the products in their archive are in those formats (Table II). Currently, analysis of all readily accessible products from all of the ESDCs is complete. Roughly 200 products that are not readily accessible remain to be obtained and analyzed. Initial analysis suggests that many of these may not actually be in HDF format, as the data are primarily Level 0 products, whereas others have not been produced yet or may be erroneous entries in NASA's EDGRS and ECHO lists. If samples of any of these products should become available over the course of this project, they will be analyzed and added to the catalog if relevant.

All but two of NASA's ESDCs archive HDF data. However, the fraction of HDF-formatted data sets in the archive at each ESDC vary widely, from identically zero to nearly one. Not quite 20% of NASA's Earth science data products are archived in HDF4 format, considerably less in number if not in volume than anticipated at the beginning of this project. Of the 586 products analyzed, a bit more than half (327) are archived in HDF-EOS format. Grids are the most common HDF-EOS data structures in use, with some 227 products using them. One-hundred products contain swath structures, while only two products contain point data. Moreover, no product uses a combination of grid, swath, and point data structures.

Nearly all swath products contain a single swath (96), with 2- or 3-D swaths being most common. The maximum number of swaths in a single file is 11, and the highest swath dimension is seven. Most swaths are organized spatially (88), but of these, 37 are also organized by time. A few products are organized by neither time nor space. Dimension maps are used in 38% of the swath products.

Unlike the swath products where most products contain a single swath data structure, many grid products contain multiple grids. Fully half of the products contain more than five grids, 16% of the products contain more than 30 grids per file, and seven products contain more than 200 grids in a single file. The vast majority of the grid products contain 2- or 3-D grids, with only nine products containing grids with more than four dimensions. Very nearly all grid products include a single projection, and no products contain multiple projections in a single file. Indexes are not used.

Only eight of the 259 purely HDF4 products analyzed contain raster data, and all eight of these products contain 8-b rasters. No general or 24-b raster data were found. None of the raster products contained either SDS or VData structures, and perhaps surprisingly, none contained a palette either.

With the exception of 18 products, all products analyzed contain at least one SDS, with 86% of the products containing 50 or fewer SDSs. A few products contain very large numbers of SDSs, with two products containing 975 SDS data structures each. As expected, the distribution of SDS dimensions parallels that of the gridded HDF-EOS products, with the majority of products containing 2- or 3-D SDS data. Roughly 80% of the products containing SDS data structures include attributes for at least some of the SDSs in the file. Nearly 20% of the products contain SDSs that are chunked, and nearly 40% are compressed, typically using gzip compression, although occasionally using run length encoding compression.

TABLE II
NASA DATA CENTER HOLDINGS

EARTH SCIENCE DATA CENTER	HOST INSTITUTION	TOTAL DATA SETS	HDF4 DATA SETS
Alaska Satellite Facility DAAC	Geophysical Institute, University of Alaska, Fairbanks, AK	52	0
Goddard Earth Science Data and Information Services Center	NASA Goddard Space Flight Center, Greenbelt, MD	1280	236
Global Hydrology Resource Center	Global Hydrology and Climate Center, NASA Marshall Space Flight Center, Huntsville, AL	110	54
Atmospheric Science Data Center	NASA Langley Research Center, Hampton, VA	850	63
Land Processes DAAC	Earth Resources Observation and Science (EROS) Data Center, U.S. Geological Survey (USGS), Sioux Falls, SD	183	67
National Snow and Ice Data Center DAAC	Cooperative Institute for Research in the Environmental Sciences, University of Colorado, Boulder, CO	283	47
Oak Ridge National Laboratory DAAC	DOE Oak Ridge National Laboratory, Oak Ridge, TN	93	2
Physical Oceanography Distributed Active Archive Center	Jet Propulsion Lab, Pasadena, CA	259	22
Socioeconomic Data and Applications Center	Center for International Earth Science Information Network, Lamont-Doherty Earth Observatory, Columbia University, Palisades, NY	6	0
Moderate Resolution Data Center	NASA Goddard Space Flight Center, Greenbelt, MD	97	95
Total		**3213**	**586**

Three-hundred-thirty products (56%) contain VData structures, with roughly two thirds containing six or fewer VData structures. Like the products containing SDS structures, these products can contain very large numbers of VData structures with ten products containing more than 150 VData structures and three products containing more than a thousand VData structures. Unlike SDS data structures, attributes for VData structures are extremely rare, as is both compression and chunking. Finally, Vdata and SDS data structures are quite often, but not always, found within the same file. This is not surprising since point and swath EOS-HDF 2 data structures both use Vdata and SDS HDF4 data structures.

```
<?xml version="1.0" encoding="utf-8"?>
<hdf4:HDFMap xmlns:hdf4="http://www.hdfgroup.org/HDF4/HDF4Map">
  <hdf4:RootGroup>
    <hdf4:SDS objName="data1" objPath="/" objID="xid-DFTAG_NDG-2">
      <hdf4:Attribute name="data range" ntDesc="32-bit signed integer">
        0 255
      </hdf4:Attribute>
      <hdf4:Datatype dtypeClass="INT" dtypeSize="4" byteOrder="BE" />
      <hdf4:Dataspace ndims="2">
        10 100
      </hdf4:Dataspace>
      <hdf4:Datablock nblocks="1">
        <hdf4:BlockOffset>
          2502
        </hdf4:BlockOffset>
        <hdf4:BlockNbytes>
          4000
        </hdf4:BlockNbytes>
      </hdf4:Datablock>
    </hdf4:SDS>
  </hdf4:RootGroup>
</hdf4:HDFMap>
```

Fig. 4. Example of a mapping file for the simple HDF4 file in Fig. 1.

B. Prototype Map Generation

To fulfill the purpose of retrieving the HDF4 data independently, the file format of the map should be both human and machine readable, portable, standard, and independent of any binary format. XML fulfills these requirements for many reasons. XML is plain text, a desirable feature for data archiving because it allows users to read mapping information with any standard text editor and it is highly unlikely that technology will make plain text unreadable any time soon. Second, XML documents are hierarchical and hence match most HDF4 file structures well. Third, other important schemas in the realm of data archiving, such as the PREMIS preservation metadata schema [13], [14], are also written in XML. Therefore, by adopting XML as the language for the map, compatibility with these schemas can be maintained. For these reasons, an XML-based prototype schema for mapping HDF4 files has been developed and is undergoing review by the project team. The intent is to publish the draft specification for review by the user community.

The goal of this pilot project is to demonstrate that we can extract data values from HDF4/HDF-EOS files without using the HDF4 library. Consequently, there are limitations in the current implementation. First, the mapping writer will not be able to extract information for all data objects and metadata supported by HDF4. It is not the intention of this project to support all use cases, although the data objects and metadata corresponding to the predominate use cases observed during the assessment have been addressed.

Second, the mapping file may not provide enough information to fully interpret user data. For example, the HDF-EOS library defines three types of data: grid data, point data, and swath data. Although HDF4 mapping file will provide information to extract the data values from an HDF-EOS file, it may not provide enough information to construct the grid, point, or swath data structure. In other words, HDF-EOS objects will be described as HDF objects in the map file, not as EOS objects.

A prototype mapping tool, *hmap*, has been developed to create an XML mapping file for a given HDF4 file according to the XML mapping schema. Since the SDS is the primary HDF4 object used in HDF-EOS data, *hmap* maps SDS objects. The map file for an SDS includes its data type, dimensions and rank, data storage type, and offset and length. SDS attributes are stored as ⟨name, value⟩ pairs. The *hmap* prototype also supports Vdatas, Vgroups, and other necessary information in the prototype. Fig. 4 shows a mapping file for the simple HDF4 file in Fig. 1.

C. Reading Maps

The key test of this mapping approach is to read an HDF4 data file using the map file, but with no reference to HDF

```
. . .
# Import XML and Inflate packages
use XML::LibXML;
use IO::Uncompress::Inflate qw(inflate);

# Parse the XML mapping file $mapfile
my $parser = XML::LibXML->new();
my $root = $parser->parse_file($mapfile)->getDocumentElement();

# Find XML node for specific dataset
my ($node) = $root->findnodes("//hdf4:SDS[\@objName='$dataset']");
my ($datatype) = $node->findnodes('.//hdf4:Datatype');

# Reverse bytes if endian-ness is not the same as this machine
my $byte_order = $datatype->getAttribute('byteOrder');
my $revbytes = (($byte_order eq 'BE') xor (pack('n',42) eq pack('s',42)));

# Determine type of data and convert to a pack template
my %pack_templates = ( 'FLOAT' => {4 => 'f',8 => 'd'},
    'INT' => {1 => 'c', 2 => 's', 4 => 'i', 8=> 'q'},);
my $dtype_class = $datatype->getAttribute('dtypeClass');
my $dtype_size = $datatype->getAttribute('dtypeSize');
my $pack_template = $pack_templates{$dtype_class}{$dtype_size};

# Extract Datablock info for dataset
my ($datablock) = $node->findnodes('.//hdf4:Datablock');
my $nblocks = $datablock->getAttribute('nblocks');

my ($i, $j, $buf, $newbuf, @data);
open (HDF, "<:raw", $datafile);   # Open HDF file $datafile
foreach $i(1..$nblocks) {         # Loop through blocks
  # Seek to start of block and read
  seek(HDF, $datablock->findvalue("./hdf4:BlockOffset[$i]"), 0);
  read(HDF, $buf, $datablock->findvalue("./hdf4:BlockNbytes[$i]"));

  inflate \$buf => \$newbuf;      # Uncompress data block
  for ($j = 0; $j < length($newbuf); $j += $dtype_size) {
    # Extract, reverse if necessary, and convert to number
    my $val = substr($newbuf, $j, $dtype_size);
    $val = reverse $val if ($revbytes);  # Reverse bytes?
    push @data, unpack($pack_template, $val);
  }
}
. . .
```

Fig. 5. Sample PERL routine reading an SDS from an HDF4 file without using the HDF libraries.

libraries or utilities. Two prototype read programs have been written, one in C and the other in Perl. The Perl read program was tested with Level 2 standard retrieval files from the atmospheric infrared sounder [15]. HDF maps were created for existing files. Variables were then extracted to external binary files using the Perl read program and compared with binary files extracted with the HDF hdp utility. Similarly, the C program was tested with a variety of snow and ice products from the Moderate Resolution Imaging Spectroradiometer [16]. For brevity's sake, we show the key subroutine in the Perl case (without error handling) in Fig. 5.

In this example, the program locates the XML section for the variable to be read using a common XML parsing package. It then extracts the salient information about the data type and byte layout: FLOAT versus INT, size in bytes, byte order, compression information, and block location and size relative to the start of the file. From here, it is straightforward, through calls to seek() and read(), to extract the blocks of interest. Decompression (inflation) is handled using another external Perl package, after which the bytes are reversed if necessary and then converted to numeric values. The current prototypes handle relatively simple cases, but work is underway to read data sets with complex "chunking" of multidimensional data.

D. Evaluating the Effort Required for Full Implementation

The last major task of this project is to evaluate the efficacy of this process and develop an estimate of the effort required to map all of the holdings in NASA's ESDC archives, a task that is still in its preliminary phase. Given that the fraction of the holdings archived in HDF4 format vary dramatically from data center to data center, it should be noted that any effort to implement full-scale mapping in NASA's ESDC archives would need to encompass the vast majority of NASA EOSDIS data centers, yet the effort required at any particular center would vary substantially from center to center.

One factor that will substantially impact the total cost of implementation across NASA's ESDCs is whether data maps can be generated at the data set level or they must be generated for each and every file within a data set. Clearly, generating a single map instead of a map for each of the hundreds, thousands, or even hundreds of thousands of files that comprise a single product would require significantly less effort. However, this mechanism can only work with simple products where the structures within a file do not change from file to file. Any product where the files have variable length data structures or attributes or where some fields have been compressed or chunked would require a different map for each data file. Initial discussions led to the hypothesis that it was highly unlikely that providing a single map for an entire product set would work for any of NASA's data. However, after analyzing samples of each data set, there is a significant fraction of the products that have fixed size data structures, and either have no attributes or have attributes with a set length, no matter their content. To demonstrate that a single data set level map will work for these products, maps will be generated for each file of an identified candidate product. The resulting sets of maps will be compared to determine if differences were noted. The results of this test will be folded into the overall estimate of implementation across NASA's Earth science data holdings.

Given these considerations, we anticipate that it will be possible to estimate the development costs for a full implementation of the mapping software as well as to develop high-level operation concepts and requirements for implementing this activity across the suite of NASA ESDCs. Complete estimates will be provided for both GES-DISC and NSIDC. An estimate will be requested from the contractor currently responsible for the maintenance of the EOS Core System. These point estimates will be extrapolated to the other ESDCs that archive HDF4-formatted data products. Detailed estimates will require coordination with each of the other data centers that archive data in HDF4 format, something that is beyond the scope of this project.

Automating the process of running the generator against the holdings of the ESDCs is likely to be feasible with minimal cost. However, depending on the architecture of each ESDC's data management system, it is likely that file system and database structural changes will be required. Initial testing shows that the additional storage volume required for the map files is a small fraction, typically less than 0.5%, of the original file size. However, this percentage varies greatly depending on the number and size of the objects within a file. In addition, once mapping for the existing archive is complete, it would be ideal to incorporate mapping as part of the ingest process for new data.

IV. Conclusion

Preliminary indications are that the mapping concept is feasible. We have successfully demonstrated that the most used features of the HDF4 library can be mapped and that the maps created can be used to write programs that can read data in HDF4 format without using the HDF library. We have yet to complete the evaluation of the effort required to implement this

strategy across all of NASA's Earth science data holdings. One of the expected outcomes of this will be the identification of the processes needed for other centers to implement these maps on their own.

Clearly, an assessment of the risks of not providing data maps, as well as the costs and benefits of implementing these data maps, needs to be made on a case-by-case basis. However, this strategy could be extended to other formats such as HDF 5 and beyond. HDF5 supports a smaller number of object types and a simpler metadata structure than HDF4, which would seem to make it even easier to map than HDF4. On the other hand, HDF5 has a richer set of array element types than HDF4, which adds some complexity. On balance, it is expected that HDF5 would be similarly amenable to a mapping strategy as HDF4.

At some point, it may be useful to extend the HDF data format itself to automatically generate maps such as those described here. THG has made preliminary forays into improving the long-term survivability of data formatted in HDF formats by developing the HDF-Archive Information Package (HDF-AIP format) [17]. This HDF5-based format associates the Metadata Encoding and Transmission Standard (METS) metadata commonly used within the digital library and archives community with the HDF5 data file [18]. The METS standard is designed to incorporate metadata from other standards, including the PREMIS metadata discussed earlier and the Content Standard for Digital Geospatial Metadata, Version 2 (FGDC-STD-001-1998) which was mandated for all federally funded geospatial metadata in 1995 [19]. The PREMIS metadata standard explicitly includes the ability to describe bit streams within a file, which is effectively what the maps described here accomplish. Consequently, future efforts will endeavor to ensure that the maps generated here are compatible with these metadata standards. Demonstrating that usability of the HDF-AIP format with NASA EOS data is the topic of another ongoing research project.

Finally, one test of the efficacy of this strategy is a measure of how much more survivable data archived in HDF4 format with these maps are, as compared to the same data without the maps. Unfortunately, direct measures of format survivability do not exist. However, recently, a new methodology for assessing the suitability of formats for long-term archive usage has been developed [20]. Using the INFORM methodology to assess the HDF4 format with and without the maps could provide one possible means for assessing the efficacy of this process.

ACKNOWLEDGMENT

The authors would like to thank H. K. Ramapriyan and D. Marinelli for their support and guidance in making this study possible, E. Pourmal of THG for her advice and expertise, and the management and staff at each of NASA's Earth Science Data Centers for making the product data and information used in this paper available.

REFERENCES

[1] A. Hesseldahl, *Saving Dying Data,* Sep. 12, 2002, Forbes. [Online]. Available: http://www.forbes.com/2002/09/12/0912data_print.html

[2] A. Jesdanun, "Digital memory threatened as file formats evolve," *Houston Chronicle*, Jan. 16, 2003. [Online]. Available: http://www.chron.com/cs/CDA/story.hts/tech/1739675

[3] R. Duerr, M. A. Parsons, R. Weaver, and J. Beitler, "The international polar year: Making data available for the long-term," in *Proc. Fall AGU Conf.*, San Francisco, CA, Dec. 2004. [Online]. Available: ftp://sidads.colorado.edu/pub/ppp/conf_ppp/Duerr/The_International_Polar_Year:_Making_Data_and_Information_Available_for_the_Long_Term.ppt

[4] M. Folk, "HDF today," *Dr. Dobbs Journal*, May 1998.

[5] NCSA, *HDF4 User's Guide*, May 1999. [Online]. Available: http://hdfgroup.org/old/UG41r3_html/

[6] NCSA, *HDF4.1r5 Specification and Developer's Guide*, Nov. 2001. [Online]. Available: ftp://ftp.hdfgroup.org/HDF/Documentation/HDF4.2r0/HDF41r5_SpecDG.pdf

[7] The HDF Group. [Online]. Available: http://hdfgroup.org

[8] M. Folk, R. E. McGrath, and N. Yeager, "HDF: An update and future directions," in *Proc. IGARSS*, 1999, vol. 1, pp. 273–275.

[9] *HDFView, a Viewer and Editor for HDF Files*. [Online]. Available: http://www.hdfgroup.org/hdf-java-html/hdfview/

[10] HDF EOS. [Online]. Available: http://hdfeos.org/software.php#HDFViewHDF-EOSplug-in

[11] *HDP, a Command Line Utility for Quick Display of Contents and Data of HDF4 Objects*. [Online]. Available: http://www.hdfgroup.org/hdp.html

[12] D. Wynne and E. M. Taaheri, *An HDF-EOS and Data Formatting Primer for the ECS Project*, Mar. 1, 2001. [Online]. Available: http://hdfeos.org/software/HDFEOS_2_5/HDFEOS_Primer_mar_2001.pdf

[13] PREMIS (PREservation Metadata: Implementation Strategies) Working Group, *PREMIS: PREservation Metadata Implementation Strategies [OCLC—Projects]*, May 2005. [Online]. Available: http://www.oclc.org/research/projects/pmwg/

[14] Library of Congress, *PREMIS: Preservation Metadata Maintenance Activity*, Jan. 25, 2008. [Online]. Available: http://www.loc.gov/standards/premis/

[15] H. H. Aumann, M. T. Chahine, C. Gautier, M. D. Goldberg, E. Kalnay, L. M. McMillin, H. Revercomb, P. W. Rosenkranz, W. L. Smith, D. H. Staelin, and J. Susskind, "AIRS/AMSU/HSB on the aqua mission: Design, science objectives, data products, and processing systems," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 2, pp. 253–264, Feb. 2003.

[16] C. O. Justice, E. Vermote, J. R. G. Townshend, R. Defries, D. P. Roy, D. K. Hall, V. V. Salomonson *et al.*, "The moderate resolution imaging spectroradiometer (MODIS): Land remote sensing for global change research," *IEEE Trans. Geosci. Remote Sens.*, vol. 36, no. 4, pp. 1228–1249, Jul. 1998.

[17] P. Cao, *HDF5 Archival Information Package (AIP) a METS Implementation*, Apr. 26, 2006. [Online]. Available: http://www.hdfgroup.uiuc.edu/papers/papers/AIP/HDF5_AIP_White_Paper.html

[18] Library of Congress, *Metadata Encoding and Transmission Standard (METS)*, Jan 25, 2008. [Online]. Available: http://www.loc.gov/standards/mets/

[19] "Federal geographic data committee," in *Content Standard for Digital Geospatial Metadata (FGDC-STD-001-1998)*. Washington, DC: Federal Geographic Data Committee, 1998.

[20] A. Stanescu, "Assessing the durability of formats in a digital preservation environment: The INFORM methodology," *D-Lib Mag.*, vol. 10, no. 11, Nov. 2004. DOI:10.1045/november2004-stanescu.

Ruth E. Duerr (M'07), photograph and biography not available at the time of publication.

Peter Cao, photograph and biography not available at the time of publication.

Jonathan Crider, photograph and biography not available at the time of publication.

Mike Folk, photograph and biography not available at the time of publication.

Christopher Lynnes (M'08), photograph and biography not available at the time of publication.

Mu Qun Yang, photograph and biography not available at the time of publication.