An Evaluation of HDF Support for NCL, GrADS, PyHDF, GDL and GDAL

Choonghwan Lee (clee83@hdfgroup.org)
MuQun Yang (myang6@hdfgroup.org)
The HDF Group

This document evaluates five freely available software packages that can access HDF4 and/or HDF5 files.

1 Prerequisite

This document assumes that users have a basic knowledge of the following data formats and the corresponding software packages:

- HDF4 (1)
- HDF5 (2)
- HDF-EOS2 (3)
- netCDF-4 (4)
- CF Conventions (5)

2 Introduction

NCL (6), GrADS (7), PyHDF (8), GDL (9) and GDAL (10) are five open source software packages that can manipulate either HDF4 or HDF5 data. This document provides a limited evaluation of HDF support for these packages from the user's point of view. The evaluation criteria include installation, features and documentation. A summary of the evaluation is listed at the end. A separate document, *Usage of NCL, GrADS, PyHDF, GDL and GDAL to Access HDF Files* (11), describes how to use these packages to access HDF files.

For the evaluations discussed in this document, we used NCL Version 5.0.0 (released in November 2007), GrADS 2.0a2 (published in April 2008), PyHDF 0.7-3 (published in July 2005), GDL 0.9rc1 (published in April 2008), and GDAL 1.5.2 (published in May 2008).

3 NCL

3.1 Overview

The *NCAR Command Language (NCL)* is an interpreted language designed for scientific data analysis and visualization. NCL supports HDF4 SDS and netCDF (12). NCL can read netCDF-4 classic model (13) files.

3.2 Installation

NCL is easy to install with precompiled binaries. Both source code and precompiled binaries for AIX, IRIX, Linux, MacOSX, Solaris and Windows under cygwin are available from http://www.ncl.ucar.edu/Download/index.shtml.

3.3 Documentation

NCL is well documented. The NCL website provides many examples with screenshots, which are particularly helpful for users learning to utilize NCL to access data.

3.4 Features

3.4.1 HDF4

Visualizing HDF4 SDS that has dimension scales can be easily done. If an HDF4 SDS has dimension scales such as longitude and latitude, NCL automatically maps data to the correct location. Also, NCL can automatically retrieve “fill value” and “unit” information if they are provided in the HDF4 file.

NCL only supports HDF4 SDS. Other HDF4 objects such as Vdata are not supported.

NCL does not automatically recognize an HDF-EOS2 file when its file name has an HDF4 extension, such as `.hdf`. One has to rename the file extension to the HDF-EOS2 file extension, such as `.he2` or `.eos`, or append a string `.he2` or `.eos` to the actual HDF-EOS2 file name.

Access to HDF-EOS2 grid files is easy because NCL calculates longitude and latitude. For both one-dimensional and two-dimensional longitude and latitude, drawing a plot overlapped with maps for HDF-EOS2 grid files is easily accomplished.

HDF-EOS2 swath without a dimension map may be easily visualized with only a few lines of code. When dimension maps exist, users need to adjust latitude or longitude fields by interpolation or subsampling, which can be done inside NCL.

3.4.2 HDF5

HDF5 is not supported yet; therefore, the current version of NCL cannot read a general HDF5 file. However, the current version of NCL does support the netCDF-4 classic model and can access an HDF5 file that complies with the netCDF-4 classic model.

3.4.3 Others

When the fill value of a variable is 0, NCL is not able to generate a plot for that variable. (This is a known problem that is documented.) To draw a contour plot when the fill value is 0, users should manually set the fill value with a nonzero value.

4 GrADS

4.1 Overview

The *Grid Analysis and Display System (GrADS)* is an interactive desktop tool that is used for easy access, manipulation, and visualization of earth science data. GrADS can draw various types of plots, including contours overlapped with a geographic map, and draw a series of plots continuously.

GrADS supports HDF4 SDS and netCDF-3. NetCDF-4 and HDF5 are not supported.

4.2 Installation

GrADS is easy to install with precompiled binaries, which worked without any problems under GNU/Linux.

Unlike NCL, GrADS does not support netCDF-4, which means that GrADS cannot read any HDF5 file. If a user builds GrADS from source code, GrADS can be linked with netCDF-4 instead of netCDF-3, but this approach requires a few undocumented tricks.

4.3 Documentation

Compared with NCL's documentation, GrADS' documentation has fewer screenshots that accompany its examples. For this reason, learning how to make comprehensive plots is more difficult.

4.4 Features

4.4.1 HDF4

Visualizing HDF4 SDS can be easily done. As NCL does, GrADS can detect dimensions, units and fill values. Also, GrADS can generate animated plots. For example, a series of plots based on the 'time' dimension can be generated.

GrADS can only access SDS objects among the various HDF4 objects.

HDF-EOS2 is not specially handled. GrADS can read HDF-EOS2 files, but we could not find any special rules related to the handling of these files. Geolocations were neither calculated nor associated. Users should provide values of longitude and latitude.

4.4.2 HDF5

Neither HDF5 nor netCDF-4 is supported. If users manage to build GrADS with netCDF-4, netCDF-4 files that conform to the netCDF-4 classical model can be accessed via GrADS, but this feat requires undocumented tricks.

4.4.3 Others

Following CF Conventions does not help when accessing HDF data with GrADS. Even if a file conforms to the CF Conventions, the user still needs to provide an additional file for GrADS. For more information, refer to "PDEF BILIN Option in GrADS" in the "Appendix" section in *Usage of NCL, GrADS, PyHDF, GDL and GDAL to Access HDF Files*.

5 PyHDF

5.1 Overview

PyHDF, developed by Andre Gosselin, is a Python interface to the HDF4 library. PyHDF has corresponding Python wrappers for HDF4 APIs such as Scientific Data Sets (SD API), Vdatas (VS API), and Vgroups (V API). PyHDF is not merely a wrapper of HDF4 C API; it exploits Python features such as the OOP concept and exception handling to make PyHDF more attractive to users.

The latest version, as of July 2008, was released in July 2005, and it was built with HDF4.2r1. However, HDF4.2r3 could be successfully linked.

5.2 Installation

Like most Python packages, `setup.py` is provided, and PyHDF is easy to install. PyHDF requires the *Numeric* package, which is a predecessor of *numpy* (14).

5.3 Documentation

The PyHDF document is succinct. PyHDF users can easily follow the HDF4 reference manual for additional information.

5.4 Features

HDF4 users can easily learn PyHDF. For those who have experience using object-oriented language binding for a procedural language, the transition from HDF4 C API to PyHDF API should be smooth and straightforward.

Regarding OOP, HDF4 C API functions are categorized into a few classes. The separation is intuitive, and each class encapsulates data and behaviors well.

It is important to note that PyHDF raises an exception of *HDF4Error* type if the underlying C API returns an error. The exception mechanism may improve readability.

PyHDF only supports HDF4 Scientific Data Sets (SD API), Vdatas (VS API), and Vgroups (V API). PyHDF supports compression without chunking, but it does not support a chunked dataset and an n-bit dataset.

6 GDL

6.1 Overview

GNU Data Language (GDL) is a free clone of *Interactive Data Language (IDL)* (15). IDL is an interpreted language that is used to manipulate scientific data and draw plots. GDL partially supports both HDF4 and HDF5. Additionally, it supports netCDF.

6.2 Installation

GDL assumes that HDF4 and HDF5 are built without SZIP (16). If HDF4 or HDF5 is built with SZIP, building GDL will fail during the configuration step. To fix this problem, one may need to manually patch the `configure` script.

Another problem arises from GDL's dependency on *PLplot* (17) and *GNU Scientific Library (GSL)* (18). Building PLplot is not simple. It requires manual modification of a script to fix the related problems.

6.3 Documentation

GDL lacks sufficient or comprehensive documentation. Users can refer to IDL documents, but they may have difficulty determining what features GDL does not implement and what features GDL implements differently.

6.4 Features

GDL users need to have some knowledge of the HDF4 programming model and APIs. For example, to read an SDS object from its name, users need to know a set of interfaces, including `HDF_SD_START()`, `HDF_SD_NAMETOINDEX()`, `HDF_SD_SELECT()`, and `HDF_SD_GETDATA()`. If a user is familiar with HDF4, using GDL may be easy because one GDL interface is mapped to one HDF4 C API. Since GDL keeps the same interfaces as IDL exposes, IDL users can easily use GDL, although GDL did not fully implement all IDL interfaces.

HDF4 is partially supported. GDL supports only SDS, Vgroup and Vdata.

Handling HDF5 requires some knowledge of HDF5. Like the HDF4 interface, one GDL interface is mapped to one HDF5 C API, which means that users need to know how to use HDF5 C API.

HDF5 is partially supported. Only a few basic dataset interfaces are supported. The HDF5 Group Interface and the HDF5 Attribute Interface are missing.

Features that generate plots are limited. IDL provides interfaces such as `MAP_SET` for mapping points on the earth's surface, but GDL does not implement this (as of June 2008). GDL can draw contours and surfaces without mapping, but it cannot draw shaded surfaces.

The official website states that graphical output is partially implemented. Missing features can be checked at http://aramis.obspm.fr/~coulais/IDL_et_GDL/Matrice_IDLvsGDL.html.

7 GDAL

7.1 Overview

The Geospatial Data Abstraction Library (GDAL) is a translator library for raster geospatial data formats. As of July 2008, GDAL supports about 100 formats, including netCDF, HDF4 and HDF5. This library was written in C++, but it also provides bindings for other languages.

7.2 Installation

GDAL 1.5.2 has one known problem resulting in compile errors. A patch (19) from the GDAL website is required to build it.

7.3 Features

GDAL's abstraction layer provides one unified API for both HDF4 and HDF5. Since GDAL hides HDF4-specific and HDF5-specific details, users do not need to learn HDF4 and HDF5, but they do need to learn the GDAL model.

GDAL recognizes HDF-EOS2-specific attributes. GDAL distinguishes HDF-EOS2 files from pure HDF4 files and detects HDF-EOS2-specific information, including projection code, related attributes, and geolocation fields.

HDF-EOS2 swath assumes that dimension maps exist. The HDF-EOS2 handler has code that detects HDF-EOS2 swath and reads geolocation fields. When we tested it, however, it failed to read geolocation fields if the dimension map does not exist.

HDF5 attribute support is immature. If an attribute is of a numeric type, GDAL converts the numbers into a string. However, it ignores attributes of user-defined types. Also, it may introduce buffer-overrun if an attribute is very long. When the projection information is requested, the HDF5 driver returns predefined values, which may mislead users. Furthermore, GDAL could not detect a fill value.

8 Conclusion

This section summarizes some advantages and disadvantages of NCL, GrADS, PyHDF, GDL, and GDAL based upon selected criteria.

8.1 Installation

Table 1 summarizes the available packages and the difficulties encountered when building a package from source code.

	Precompiled binaries	**Source build**
NCL	AIX, IRIX, Linux, MacOS X, Solaris, Cygwin	Available, but we have not tested
GrADS	Linux, MacOS X, SunOS	Available, but we have not tested
PyHDF		Requires few settings
GDL		Requires several packages; Assumes HDF4 and HDF5 without SZIP
GDAL		Requires one patch

Table 1 Installation

PyHDF provides a setup script like most Python packages, and installation is easy. Both NCL and GrADS distribute precompiled binaries, and both of them worked under Linux. GDAL requires one minor fix, which is documented in their bug tracker, and the instructions are easy to follow. Installing GDL may be difficult mostly because of the libraries that it depends on.

8.2 File Format

Since PyHDF is an HDF4 wrapper, it does not support other APIs inherently. Except for PyHDF, all tools support both HDF4 SDS and netCDF-3 as Table 2 shows.

	HDF4	**HDF5**	**NetCDF**
NCL	SDS	Possible through netCDF-4 API	netCDF-3; netCDF-4 classic model
GrADS	SDS	Possible through netCDF-4 API with undocumented hack	netCDF-3; netCDF-4 classic model with undocumented

			hack
PyHDF	SDS, Vdata, Vgroup		
GDL	SDS, Vdata	Supported	netCDF-3
GDAL	SDS, GR, EOS2	Supported	netCDF-3

Table 2 Supported Formats

GDL and GDAL support HDF5, but GrADS and NCL do not. We believe all applications will support at least the netCDF-4 classic model because this support does not require many changes in the applications. We could make GrADS 2.0a2 read netCDF-4 files that conform to the netCDF-4 classic model by making a few modifications.

At the time of writing this document, none of the tools support netCDF-4 files that have more than one HDF5 group.

8.3 Documentation

Table 3 compares documents and examples that each package offers.

	Documentation	**Examples**
NCL	Sufficient	Many examples with screenshots
GrADS	Sufficient	A few examples
PyHDF	Simple, but sufficient; Users can refer to HDF4 Reference	Several simple examples
GDL	Only a few documents exist; Users can refer to IDL documents	Users need to read IDL documentation.
GDAL	Sufficient	A few examples in Tutorial; Working applications included

Table 3 Documentation

Since PyHDF is a wrapper, its simple documentation is generally sufficient. PyHDF users can easily follow the HDF4 documents for additional information.

GDL does not provide sufficient documentation, but users can refer to IDL documents. One should note that there are missing features and different behaviors inside GDL.

NCL, GrADS and GDAL have many documents. In terms of visualizing the information, we think NCL documents are the most helpful because their website presents many examples that are accompanied by screenshots, which made it easier to learn how to draw elegant plots.

8.4 Features

Table 4 shows a few advantages and disadvantages of each package.

	Advantages	**Disadvantages**
NCL	Supports the netCDF-4 classic model and HDF-EOS2; Easily handles files compliant with CF Conventions	Only the netCDF-4 classic model is supported.
GrADS	Easily draws plots from HDF4 and netCDF-3	NetCDF-4 is not supported; Two-dimensional coordinate variables

		are not easily handled.
PyHDF	Easy for HDF4 users	Only SDS, Vgroup and Vdata are supported.
GDL	Supports HDF5	Many features of HDF4 and HDF5 are missing.
GDAL	Abstraction layer hides HDF4-specific and HDF5-specific features	HDF5 support is not mature.

Table 4 Advantages and Disadvantages

HDF4 C API users can learn PyHDF easily. Also, handling HDF4 files and HDF5 files with GDL is easy for HDF4 C API users and HDF5 C API users because GDL has a very thin abstraction layer.

Due to GDAL's abstraction layer, users do not need to know HDF4 or HDF5.

Both NCL and GrADS could recognize HDF4 and netCDF-4 attributes that are necessary for NCL and GrADS to easily draw plots. For HDF-EOS2 files, NCL can calculate longitude and latitude even when longitude and latitude are two-dimensional. Handling HDF-EOS2 files that have two-dimensional longitude and latitude is more difficult for GrADS because GrADS requires an additional description file, which generally requires more efforts.

9 References

1. *HDF4.* [Online] The HDF Group. http://www.hdfgroup.org/products/hdf4/.

2. *HDF5.* [Online] The HDF Group. http://www.hdfgroup.org/HDF5/.

3. HDF-EOS2. *HDF-EOS.* [Online] http://www.hdfeos.org/software.php#HDF-EOS2.

4. *NetCDF-4.* [Online] Unidata. http://www.unidata.ucar.edu/software/netcdf/netcdf-4/.

5. CF Conventions. [Online] http://cf-pcmdi.llnl.gov/.

6. *NCAR Common Language (NCL).* [Online] http://www.ncl.ucar.edu/.

7. *GrADS.* [Online] http://www.iges.org/grads/grads.html.

8. *PyHDF.* [Online] http://pysclint.sourceforge.net/pyhdf/.

9. *GNU Data Language (GDL).* [Online] http://gnudatalanguage.sourceforge.net/.

10. *Geospatial Data Abstraction Library (GDAL).* [Online] http://www.gdal.org/.

11. Usage of NCL, GrADS, PyHDF, GDL and GDAL to Access HDF Files. [Online]

12. *NetCDF.* [Online] http://www.unidata.ucar.edu/software/netcdf/.

13. The NetCDF-4 Classic Model Format. [Online] http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/NetCDF_002d4-Classic-Model-Format.html#NetCDF_002d4-Classic-Model-Format.

14. *NumPy.* [Online] http://numpy.scipy.org/.

15. *Interactive Data Language (IDL).* [Online] http://www.ittvis.com/idl.

16. Szip Compression in HDF Products. [Online] http://hdf.ncsa.uiuc.edu/doc_resource/SZIP/.

17. *PLplot.* [Online] http://plplot.sourceforge.net/.

18. *GNU Scientific Library (GSL).* [Online] http://www.gnu.org/software/gsl/.

19. Ticket #2296. *GDAL.* [Online] http://trac.osgeo.org/gdal/attachment/ticket/2296/gdal-hdf4-UNKNOWN.diff.