NetCDF-4/HDF5 Libraries Interoperability Issues

Larry Knox, Elena Pourmal

September 22, 2010

There is an increasing demand from the Earth Sciences community to achieve better interoperability between netCDF-4 and HDF5 libraries. This document describes a few common use cases that demonstrate the existing interoperability problems and discusses the possible solutions.

Introduction

In the past year The HDF Group Help Desk and developers received several questions and problem reports from the users of the netCDF-4 applications, who wanted to read and modify "non-netCDF-4" HDF5 files using the netCDF-4 library, or wanted to create applications that access netCDF-4 and HDF5 files using both netCDF-4 and HDF5 libraries. Unfortunately the current design and implementation of the HDF5 and netCDF-4 libraries make it difficult to achieve this kind of interoperability.

In our preliminary investigation we identified at least two sources of the interoperability problems:

1. The HDF5 and netCDF-4 libraries (and an application program) use different default settings and conflicting HDF5 programming models.
   This type of interoperability failure is discussed in the use cases 1 and 2 below.

2. The netCDF-4 library aborts operations on an HDF5 file that lacks certain properties, or when it encounters unsupported properties found in the file.

   NetCDF-4 files have certain characteristics or properties such as indexing objects according to creation order, presence of dimension scales, and a limited set of supported datatypes.[1] When the netCDF-4 library operates on an HDF5 file and the objects stored in it, it checks those characteristics and operations may fail if certain properties are not used or unsupported properties are encountered. This type of interoperability is discussed in use cases 3 and 4.

Four major use cases described below were derived from the questions and problems reported to the HDF Group. While these use cases definitely do not cover all possible interoperability problems, they do represent the major problems the users of the NPOESS files will encounter, and therefore provide a good foundation for investigating library interoperability[2].

---

[1] For more information see discussion of the netCDF-4 format http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/NetCDF_002d4-Format.html

[2] We used netCDF-4 version 4.1.2-beta1-snapshot2010091920 of Sep 20 2010 08:21:22 and HDF5 version 1.8.6

Use cases and interoperability issues

Use Case 1: Simultaneous access of an HDF5 file with the HDF5 and netCDF-4 libraries

An application opens a file using the netCDF-4 library and then opens the same file using the HDF5 library.

Issue: The call to the HDF5 library H5Fopen fails.

Note: The order of the open calls is irrelevant: if HDF5 opens the file first, then netCDF-4 open will fail. This is a problem for *any* application using **both** libraries to access a file **simultaneously.**

The problem occurs because the netCDF-4 and HDF5 libraries use different default file access properties (specifically, a close degree mode set by the H5Pset_fclose_degree function, which is "H5F_CLOSE_WEAK" for HDF5 and "H5F_CLOSE_STRONG" for netCDF-4).

Currently there is no way for an application to find the properties used by the netCDF-4 library. If the application opens an HDF5 file via the netCDF-4 library first, one can only use a trial and error approach (or use the intimate knowledge of netCDF-4 internals) to find and open the file with the correct properties in HDF5. If application opens the file via HDF5, there is no way to pass the access property used by HDF5 to netCDF-4.

Proposed solutions:

0. Document the current behavior of netCDF-4 and recommend that applications use the same close degree for the HDF5 library.

   This approach requires extra work on opening the file via HDF5. With recent enhancements to HDF5, it is probably possible to relax netCDF-4 setting of the close degree H5F_CLOSE_STRONG property, and use the default one. One should note that in general this approach is not flexible and doesn't give an application the ability to tune performance by specifying appropriate access properties in both netCDF-4 and HDF5.

1. The application could check to see what access properties and specifically close degree modes are already in use for a file it is about to open and set those properties to match. In this case both HDF5 and netCDF-4 library should provide appropriate functions to retrieve such information. NetCDF-4 will need to provide a function to change the access properties to a file.

   This approach will provide necessary tuning mechanisms to access files via both HDF5 and netCDF-4 libraries.

2. Opening a file with HDF5 or netCDF-4 (and ultimately HDF5 underneath) could conceivably check for access properties in use for the file and automatically set the same mode for the current opening of the file.

This action will be transparent to an application, but may be not desired since application will not be aware of the access property changed on the fly.

Use Case 2: Usage of HDF5 wrapper libraries and netCDF-4 library in the same application

A C++ application opens and closes a netCDF-4 file using netCDF-4 library calls, then opens a different HDF5 file using HDF5 C++ API calls.

Issue: C++ H5Fopen call fails.

The failure occurs because netCDF-4 explicitly shuts down the HDF5 library. NetCDF-4 calls H5close() when closing an HDF5 file, attempting to avoid memory leaks in application and HDF5. Shutting down the library with H5close() invalidates property list identifiers held by the C++ H5File class created at application start-up.

Proposed solutions:

1. Remove the H5close() call from netCDF-4.

   This has been done beginning with the current netCDF-4 snapshot (9/11/2010), but could also be done for older netCDF-4 code that can be rebuilt.

2. Document effect of H5close for applications built on top of HDF5.

Use Case 3: Modifying an HDF5 file opened with the netCDF-4 library

A netCDF-4 application attempts to open an HDF5 file for modification (i.e. with the NC_WRITE flag).

Issue: Open operation fails.

The failure occurs if netDCF-4 opens an HDF5 file for writing that doesn't have creation order tracking enabled. NetCDF-4 uses a call to H5Oget_info_by_idx with H5_INDEX_CRT_ORDER to determine if creation ordering is available. If the call fails (return value < 0) and NC_WRITE was specified, nc_open will fail.

Proposed solutions:

1. TBD. If netCDF-4 can read the file, i.e., there are no unsupported types as in use case 4 it may or may not be possible to write to the file, depending on netCDF-4 requirements.

2. Provide a special flag to open such files and allow the application to modify data of one of the supported datatypes.

   Having a special flag, which an application can use when opening a non-netCDF-4 file, will allow preserving the default behavior of netCDF-4 while providing necessary functionality to the application.

Use Case 4: Opening an arbitrary HDF5 file with the netCDF-4 library

A netCDF-4 application opens an NPOESS file to read a dataset (e.g. Longitude - 2-D array of floats)

Issue: Open operation fails.

The failure occurs because netCDF-4 library traverses the file and encounters a dataset of type H5T_REFERENCE that is not supported in netCDF-4 code. Datasets with object and region references is one of unique characteristics of the NPOESS files.

Proposed solutions:

1. One option would be that netCDF-4 would ignore or report as unknown the unsupported types while opening the file with access to the supported types. The feasibility and amount of work for netCDF-4 code to do this is unknown.
2. As in use case 3, netCDF-4 will provide a flag that skips datatype checking during the open operation, providing access to the data of the supported type.

These examples are not intended as a complete list of problems that may be encountered by applications using both HDF5 and netCDF-4. Hopefully illustrating these basic problems will aid in considering the options for removing obstacles to using them simultaneously or interchangeably.