# Merging the NetCDF and HDF5 Libraries to Achieve Gains in Performance

Ed Hartnett
Unidata/UCAR
P.O. Box 3000
Boulder, CO, 80307-3000

*Abstract* - **The overall goal of this collaborative development project is to create and deploy software that will preserve the desirable common characteristics of netCDF and HDF5 data models and data access libraries, while taking advantage of their separate strengths: the widespread use and simplicity of netCDF and the generality and performance of HDF5. We have recently completed a prototype implementation of the netCDF-3 C interface using HDF5 as a storage layer, and have tested the performance of this combination to verify that read/write times and file sizes are satisfactory. We will present the latest status of this software and the benefits we see from its use for numerical models, large datasets, parallel I/O, and analysis and visualization applications.**

## I. Introduction

Unidata's netCDF data format is widely used in the Earth Science community. NetCDF consists of a data model, a platform-independent storage format, and a set of software libraries which allow users to create, write, and read data in netCDF files.

The netCDF-4 project seeks to merge the netCDF interface with the HDF5 data format, while preserving backward compatibility for existing netCDF users, and adding new features to the netCDF interface.

## II. What is netCDF?

NetCDF (Network Common Data Form) is a data model and an accompanying API. It was designed to store scientific data, and it's simplicity and portability, along with the wide applicability of its data model, have ensured its success.

NetCDF was developed at Unidata, by Russ Rew, Glenn Davis, and Steve Emerson starting in 1988. The most recent version of the netCDF API, 3.5.1, was released in 2004. As will be described in this paper, major new features are being added to netCDF, and will be released in 2005.

### *A. Supported Languages and Platforms*

The netCDF library is available in several programming languages. The original interface was written in C. This interface is made available to Fortran 77 programmers as a Fortran interface. Additional C++ and Fortran 90 interfaces have been written (both of which use the C interface).

These four language interfaces (C, C++, F90, F77) have been ported to many different Unix platforms (including the Macintosh), and have also been ported to Windows.

A completely seperate Java interface exists, which has interesting additional features, like the capability to subset remotely stored data over the internet. A Java implementation is beyond the scope of the netCDF-4 project. Such an implementation may be developed independently at Unidata.

Over the years, netCDF users have contributed some very valuable interfaces in other languages. Perl, Python and Ruby interfaces exist, and are in wide use. A VB.NET interface was recently introduced.

### *B. A Note About Terminology*

Although their data models are similar, netCDF and HDF5 use different terminology for the most important element in the data model - the data arrays. In HDF5 these are called "datasets," in netCDF they are called "variables." To add to the confusion, in netCDF documentation, a "dataset" means an entire data file, with all it's variables.

### *C. The NetCDF Data Model*

The netCDF data model is a simple one, well-suited to arrays of scientific data. In the netCDF model, an array of data, or variable, has zero or more dimensions and accompanying metdata.

The dimensions are often used by more than one variable in a netCDF file.

To store ancillary metadata, attributes can be attached to any variable, or to the file as a whole.

Variables and attributes must be arrays or scalars of one of six defined netCDF types, character, 1-byte integer, 2-byte integer, 4-byte integer, 4-byte floating point, or 8-byte floating point.

Fig. 1 shows the netCDF data model, with two variables, sharing three dimensions. One variable attribute is shown, and one file-level attribute is shown.

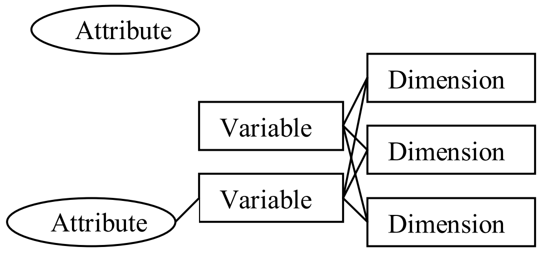

Fig. 1 NetCDF Data Model

*E. A NetCDF Example*

Consider a typical application of netCDF. A user wishes to store some surface pressure and temperature on a latitude/logitude grid, with several timesteps of data.

To store such data in netCDF, the user would define three dimensions, two variables, and as many attributes as seemed significant.

(In the example, one attribute is attached to the temp variable, showing its units; another applies to the file as a whole, storing the name of the researcher).

Note that the three dimensions are shared by both variables. Sharing dimensions is a common netCDF practice. Many geoscience data sets consist of many variables, all gridded on the same gridpoints, e.g. atmospheric model output.

Fig. 2 shows a diagram of this data file.

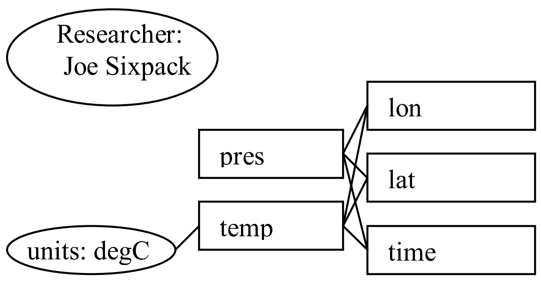

Fig. 2 Example of NetCDF Data Model

This data file can be created with the C program in Table 1. Fortran code to create this file would look very similar.

TABLE 1

C Code to Produce Example NetCDF File

```
/* This netCDF sample program stores one day
   of phony hourly surface pressure and temp.

   Ed Hartnett, 5/10/4
*/

#include "netcdf.h"

#define MAX_LON 10
#define MAX_LAT 20
#define MAX_TIME 24

int
main()
{
   int ncid, tempid, presid;
   int dimids[3];
   float *fp;
   int i, j, k;
   float temp[MAX_LON][MAX_LAT][MAX_TIME];
   float pres[MAX_LON][MAX_LAT][MAX_TIME];

   /* Create phoney data. */
   for (i=0; i<MAX_LON; i++)
      for (j=0; j<MAX_LAT; j++)
         for (k=0; k<MAX_TIME; k++)
         {
            temp[i][j][k] = 10. + j;
            pres[i][j][k] = 1000. + k;
         }

   /* Create the file. */
   nc_create("test.nc", NC_CLOBBER, &ncid);

   /* Define three dimensions. */
   nc_def_dim(ncid, "lon", MAX_LON, &dimids[0]);
   nc_def_dim(ncid, "lat", MAX_LON, &dimids[1]);
   nc_def_dim(ncid, "time", MAX_TIME, &dimids[2]);

   /* Define two data variables. */
   nc_def_var(ncid, "temp", NC_FLOAT, 3,
              dimids, &tempid);
   nc_def_var(ncid, "pres", NC_FLOAT, 3,
              dimids, &presid);

   /* Define attributes. */
   nc_put_att_text(ncid, NC_GLOBAL, "Researcher",
                   12, "Joe Sixpack");
   nc_put_att_text(ncid, tempid, "units", 5,
   "degC");

   /* Tell library that metadata is complete. */
   nc_enddef(ncid);

   /* Write the pressure and temperature data. */
   nc_put_var_float(ncid, tempid, &temp[0][0][0]);
   nc_put_var_float(ncid, presid, &pres[0][0][0]);

   /* Close the netCDF file. */
   nc_close(ncid);
}
```

*F. NetCDF Features Not Demonstrated in the Example*

In netCDF users may also define an “unlimited” dimension, which allows them to grow the data along one dimension. For example if the time dimension were to be declared unlimited, then new data records could be added to the file along the time dimension.

Its a convention in netCDF files to store coordinate data in a variable with the same name as the dimension to which it applies. In the example, the user might create a variable called "lon" and store in it the values of longitute that apply along that axis.

NetCDF also allows sophisticated subsetting and type-conversion of the data during reads and writes.

*F. NetCDF Limitations*

Several significant limitations exist in the original netCDF data model and file format. Due to an internal 31-bit offset field, datasets greater than 2 gigabytes can only be written with difficulty.

(A new version of netCDF, version 3.6.0, will introduce a modified netCDF file format that allows much larger files. This new format, 64-bit offset netCDF, addresses the needs of users who need very large files. NetCDF 3.6.0 is scheduled for release in the Summer of 2004.)

The original file format also restricts the number of unlimited dimensions to one. In the example above, if we made time an unlimited dimension, we could add more timesteps to the file. However, we couldn't ever add any more latitude or longitude points.

In netCDF-4 all of these limitations will be removed.

## III. MERGING THE FORMATS

*A. Architecture*

In Fig. 3, the architecture of the netCDF-4 prototype is shown. The netCDF-4 library acts as a layer between the end user and the netCDF-3 and HDF5 libraries. The user can choose to create data files in netCDF-3 format, or in HDF5 format.

When reading the data, the netCDF-4 layer transparently adjusts to the file format. NetCDF-3 format files are read using the netCDF-3 library. HDF5 files are read with the HDF5 library, and presented to the user by the netCDF-4 layer as a netCDF style file.

HDF5 users may access HDF5 files produced by the netCDF-4 library directly, without using the netCDF-4 interface.

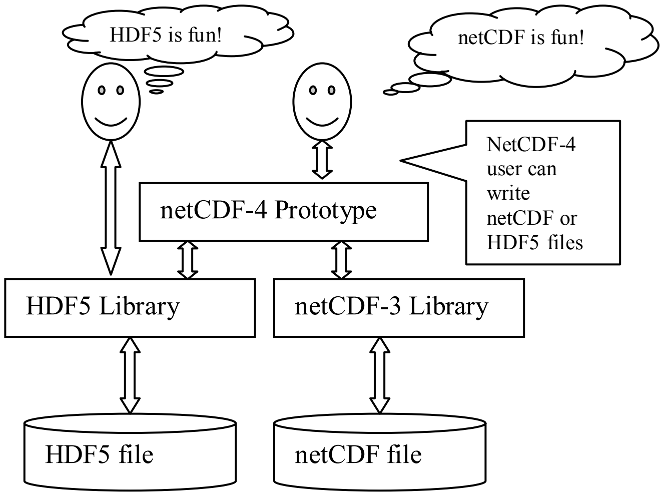

Fig.3 NetCDF-4 Prototype accesses either the netCDF-3 or HDF5 library to read/write data file.

*B. Compatibility*

Current users of the netCDF library won't suddenly lose access to old data by upgrading netCDF. NetCDF-4 can read netCDF-3 files transparently, and can create them.

Since the netCDF-4 API is a super-set of the netCDF-3 API, existing netCDF programs will be able to upgrade to HDF5 with a simple recompile.

*C. Performance*

Performance is always an important issue with scientific data access, and netCDF-4 must not degrade performance in any way. As Table 2 shows, the netCDF-4 prototype performs reasonably well.

The table shows the total CPU use and the wall clock time. The times are averaged over thirty repetitions of the same task. A 2000 x 300 x 500 array of ints is written, then read, by netCDF-3, then by netCDF-4 writing netCDF-3 files, then netCDF-4 writing HDF5 files.

The netCDF-3 format is always big-endian, and netCDF-3 suffers in performance on little-endian machines. NetCDF-4 tells HDF5 to use the native endianess of any machine; it outperforms netCDF-3 on little-endian machines.

TABLE 2

Prototype Performance Tests on Little Endian Machine

| Test | CPU Time (s) | User Time (s) |
|---|---|---|
| netCDF-3 write (BE) | 17.15 | 45 |
| netCDF-4 writing netCDF (BE) | 17.26 | 45 |
| netCDF-4 writing HDF5 (LE) | 15.87 | 36 |
| netCDF-3 read | 9.18 | 29 |
| netCDF-4 reading netCDF | 11.64 | 20 |
| netCDF-4 reading HDF5 | 12.49 | 20 |

### *D. Future Plans*

The following new features are currently being added to HDF5, to support netCDF-4:

- NetCDF style type conversion with fill-value substitution for out-of-range values.
- Shared dimension scales, and dimension coordinate data handling.
- Object access by creation order index.
- Zero-length attributes.

At the same time, netCDF is moving towards the merger with the release of accumulated changes and bug fixes, and a revision of the documentation to prepare for the addition of netCDF-4 features.

In the coming six months the prototype will be combined with the existing netCDF distribution. This will allow the use of HDF5 as a storage layer for the netCDF-3 API.

Once the software has been stabilized, new netCDF-4 features will be added. All new features will be released by the summer of 2005.

The new features of netCDF-4 will include:

- Better support for parallel I/O.
- Support for new data types.
- Multiple unlimited dimensions.
- Support for data compression and packing.
- Use of HDF5 group feature for better data organization.

## IV. CONCLUSION

The merger of netCDF and HDF5 will deliver a number of important benefits for users. NetCDF users will gain access to the new features that HDF5 makes possible. HDF5 users will be able to take advantage of the large body of working netCDF code.

## ACKNOWLEDGMENTS

This work is funded by the NASA Earth Science Technology Office under NASA award AIST-02-0071. The author would also like to extend thanks to Mike Folk from NCSA, and Russ Rew from Unidata, for the opportunity to work on this exciting and important project.

## REFERENCES

NCSA HDF5 web site: http://hdf.ncsa.uiuc.edu/HDF5

Unidata NetCDF web site:
http://www.unidata.ucar.edu/packages/netcdf

Unidata NetCDF-4 web site:
http://my.unidata.ucar.edu/content/software/netcdf/netcdf-4/index.html