SDCI Progress Report

2 April 2008

Period: 1/1/2008 – 31/3/2008

Introduction

The initial milestone for this project was to accomplish the following goal.

- Goal 1: libdap++/libnc-dap + netcdf 3.6.x to be delivered in pre-beta snapshot release by 3/2008. The idea here is to take the libnc-dap C++ software and move it into Unidata's code base, get it integrated into the build and get something usable to people right away. The libnc-dap software is effectively the glue code which connects the DAP implementation to the netCDF library.

This goal has been substantially accomplished, although the result has not been released to any external group yet for experimental use.

Process

The integration process involved the following steps.

1. Add *libdap* and *libnc-dap* to the netCDF code base. The *libdap* library implements the client-side of the DAP protocol. The libnc-dap library implements a translation between the netCDF3 API and the libdap API.
2. Integrate the *libdap* and *libnc-dap configure.ac* files into the existing netCDF *configure.ac*. Recall that there are currently two versions of *configure.ac*: one for netCDF version 3 and one for netCDF version 4. As part of this process, these two existing *configure.ac* files were merged into a single *configure.ac* and the *libdap* and *libnc-dap configure.ac* were merged into that to produce a unified *configure.ac*.
3. The *Makefile.am* files for *libdap* and *libnc-da*p were modified to utilize the new *configure.ac*.
4. The *ncdump* program was modified to accept OPeNDAP urls as file arguments.
5. Test the integrated system.
6. Place all of the relevant modifications under conditional flags. The flag in the Makefiles was called BUILD_DAP and the flag in *config.h* was called USE_DAP.

Configure Options

As a result of the *configure.ac* merge, a number of new options have been added for the configure command. These include the following.

--enable-dap	This enables the inclusion of *libdap* and *libnc-dap* functionality into the netCDF build.
--with-curl	The dap libraries require the curl library for processing URLs obtaining data across the internet.
--with-xml2	The dap libraries require the xml2 library for processing XML data that comes from DAP servers.

--enable-renamev3	Rename the netCDF3 API for use with netCDF4.

Known Problems

The integration did not proceed entirely smoothly. A number of problems were encountered. At this point, solutions (work-arounds really) exist for those problems.

Libtool Library Linking

It turns out that libtool (or at least the version(s) used by Unidata) is not capable of recognizing if a library was produced by C versus C++. Consider the following case. *Ncdump* is C code, but it must link to the appropriate netCDF library to access the netCDF API. Normally, this library is produced from C code. But when *libnc-dap* and *libdap* are used, then *ncdump* will be linked against some combination of C and C++ libraries. Libtool is not capable of telling if a library was produced using C++ code, so it does not link in the required C++ libraries (e.g. stdc++).

The Automake manual provides a rather ugly but usable work-around by including the following for each executable (including test programs).

```
if BUILD_OPENDAP
nodist_EXTRA_ncdump_SOURCES=dummy.cc
endif
```

The idea is to “trick” libtool into assuming that the executable (e.g. *ncdump*) is being compiled with a C++ program called *dummy.cc*. This causes libtool to properly link with the necessary C++ libraries. Note that dummy.cc does not in fact exist nor does it have to exist. A perusal of the various new netCDF *Makefile.am* files will show a number of instances of the above work-around. Note that this Makefile code is conditional on using DAP, so it has no effect when DAP is not enabled.

Fortran Issues

Currently, enabling DAP support disables all FORTRAN support. This is because the FORTRAN libraries do not work with the libnc-dap library. This should be a temporary condition and is being addressed.

Windows Issues

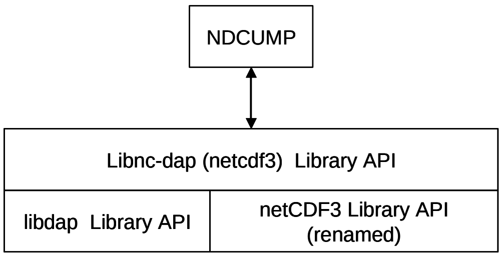
The OPeNDAP integration currently does not support windows.

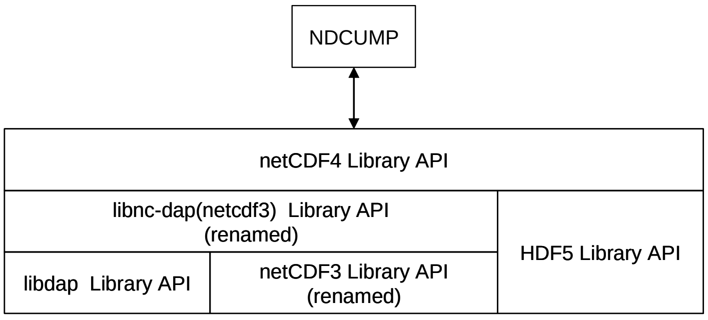
Library Layering and API Renaming

The layering of the various libraries is shown in Figures 1 and 2. Figure 1 shows the layering when using *libnc-dap* to replace the standard netCDF3 library. Figure 2 shows the layering when libnc-dap is used in the context of the netCDF4 library.

The difference has to do with API renaming. In the original netCDF4 distribution, the netCDF3 API is renamed consistently (using #define statements) so that the original names can be used in the netCDF4 API. Thus *nc_inq_var* is implemented in the netCDF4 API and the netCDF3 version is renamed to be *nc3_inq_var*.

When *libnc-dap* is added, this same kind of renaming is still necessary, but there is an added complication when it is used with netCDF4. In this latter case, the *libnc-dap* API must be


NDCUMP
Libnc-dap (netcdf3) Library API
libdap Library API
netCDF3 Library API (renamed)
Figure 1. netCDF3 Library Structure


NDCUMP
netCDF4 Library API
libnc-dap(netcdf3) Library API (renamed)
HDF5 Library API
libdap Library API
netCDF3 Library API (renamed)
Figure 2. netCDF4 Library Structure

renamed as required by netCDF4. But this also means that the original netCDF3 API must be renamed to some second format so that there is no confusion. This second rename converts, for example, *nc_inq_var* to *lnc3_inq_var*.

The original *libnc-dap* did a similar rename but was apparently implemented before rename was added to netCDF for version 4. As part of the integration, the original rename was modified to utilize the existing rename mechanism. Additionally, the secondary rename mechanism was introduced to support the situation shown in Figure 2.

Testing

There were two testing goals. First, support all existing netCDF3 and netCDF4 tests against the *libnc-dap* library replacement for the netCDF3 library. Second, support the *libnc-dap* tests that come with the standard *libnc-dap* distribution. With respect to the latter goal, this essentially required that *ncdump* support passing urls to the library because all of the existing *libnc-dap* tests used *ncdump* as the testing vehicle. No attempt was made to support the existing *libdap* tests.

The tests to be supported were those in the directories *nctest, nc_test, nc_test4,* and *nf_test*. In addition, a new test directory was added called *ncdap_test* to hold modified versions of the original *libnc-dap* tests.

There was one small issue with testing of the netCDF3 library in the *libsrc* directory. That directory contains a single test. However, since the netCDF3 library in *libsrc* must be built before the *libnc-dap* library, it is not possible for that test program to run against the *libnc-dap* library. So that test case was modified to continue to test the library produced in *libsrc* by using renaming when *–enable-dap* is true.

Testing Status

Currently, all testing goals have been met with some exceptions.

1. The FORTRAN tests are not working correctly. This is being addressed.
2. Two of the original *libnc-dap* tests are causing illegal memory failures. Since all of the other tests are working, fixing these two has low priority.
3. Occasionally, the *libnc-dap* tests in *ncdap_test* fail for no discernible reason. Re-running the tests usually results in successful execution. One possible cause is slow response by the OPeNDAP test servers that cause a timeout. Again, solving this problem is deemed low priority.

One consequence of #3 is that failures in the *ncdap_test* tests do not halt the build. This means that the output should be manually inspected to look for any anomalies.